ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»

(освітньо-кваліфікаційний рівень)

на тему

Інформаційна технологія підбору та рекомендації кандидатів на вакансію на основі компетентнісної оцінки

Information technology for recruiting and recommending candidates for a vacancy based on competency assessment

Виконав: студент денної форми навчання

спеціальності 126 – Інформаційні системи та технології .

(шифр і назва напряму підготовки, спеціальності)

Освітня програма «Інформаційні системи та технології»

(назва освітньої програми)

Джигов Дмитро Юрійович

(прізвище, ім'я, по-батькові)

Керівник д.т.н., проф. Малахов. Є.В.

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент канд. т. н, доц. Пенко В.Г

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2024 р.

Завідувач кафедри

_____ _Євгеній МАЛАХОВ_

(підпис) (ім'я, прізвище)

Захищено на засіданні ЕК № ___

протокол № __ від «_ _» ____ 2024 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Володимир ВИЧУЖАНІН_

(підпис) (ім'я, прізвище)

Одеса - 2024

АНОТАЦІЯ

В даній кваліфікаційній роботі розглянуто процес підбору та рекомендації кандидатів на вакансії шляхом розробки інформаційної технології на основі компетентнісної оцінки за допомогою методів машинного навчання.

Метою даного дослідження є підвищення ефективності процесу підбору кадрів за рахунок зниження суб’єктивності вибору шляхом розробки інформаційної технології компетентністної оцінки персоналу на основі комбінованої рекомендаційної моделі, яка поєднує контентну та колаборативну фільтрацію.

Об'єктом дослідження є процес підбору та рекомендації кандидатів на вакансії у галузі крюїнг-бізнесу.

Предметом дослідження є методи та алгоритми класифікації та ранжування груп кандидатів (моряків) в ІС «Crewisor» крюїнг-сервісу СТАФФ ЦЕНТР.

Розроблено комбіновану систему рекомендацій, яка поєднує контентну модель на основі XGBoost та колаборативну модель на основі KNN. Реалізовано механізм динамічного налаштування вагового коефіцієнта для оптимального поєднання результатів обох моделей. В якості прикладної предметної області обрано процеси найму крюїнг-компанії.

Ефективність розробленої технології підтверджено за двома ключовими критеріями: швидкість відбору кандидатів збільшилась утричі, що характеризує покращення продуктивності системи, а точність відбору зросла в півтора рази за рахунок автоматизації процесу рекомендацій та зниження впливу суб'єктивних факторів.

ABSTRACT

This thesis examines the process of selecting and recommending candidates for vacancies through the development of an information technology based on competency assessment using machine learning methods.

The object of the study is the process of selecting and recommending candidates for vacancies in the crewing business.

The subject of the study is the methods and algorithms for classifying and ranking groups of candidates (seafarers) in the «Crewisor» IS of the STAFF CENTER crewing service.

A hybrid recommendation system has been developed that combines a content-based model based on XGBoost and a collaborative model based on KNN. A mechanism for dynamically adjusting the weight coefficient has been implemented to optimally combine the results of both models. The hiring processes of a crewing company were chosen as the applied subject area.

The effectiveness of the developed technology has been confirmed by two key criteria: the speed of candidate selection has tripled, which indicates an improvement in system performance, and the accuracy of selection has increased by one and a half times due to the automation of the recommendation process and the reduction of the influence of subjective factors.

ЗМІСТ

Стор.

СПИСОК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ 7

ВСТУП 8

1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПІДБОРУ КАНДИДАТІВ НА ВАКАНСІЇ 10

1.1 Введення в предметну область 10

1.1.1 Традиційний підхід 10

1.1.2 Бази даних та системи управління кандидатами 10

1.1.3 ML та штучний інтелект 11

1.1.4 Важливість компетентнісної оцінки 11

1.2 Огляд методів оцінки та відбору кандидатів 11

1.2.1 Проблеми організації процесу відбору 12

1.2.2 Методи, засновані на експертних та психологічних оцінках 13

1.2.3 Методи засновані на даних 14

1.2.4 Комбіновані методи 15

1.2.5 Висновки за методами відбору 16

1.3 Постановка задачі 17

2 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВІДБОРУ ТА РЕКОМЕНДАЦІЇ КАНДИДАТІВ НА ОСНОВІ КОМБІНОВАНОЇ МОДЕЛІ 19

2.1 Опис технології комбінованої рекомендації 19

2.2 Кроки вирішення задач 20

3 ТЕОРЕТИЧНІ ОСНОВИ МЕТОДІВ ТА МОДЕЛЕЙ У ОСНОВІ РЕКОМЕНДАЦІЙНОЇ ТЕХНОЛОГІЇ 22

3.1 Контентна фільтрація 22

3.1.1 Принципи роботи контентних моделей 22

3.1.2 Використання атрибутів об'єктів для рекомендацій 23

3.2 Алгоритми та моделі контентної фільтрації 23

3.2.1 Огляд моделей Random Forest та XGBoost 24

3.2.2 Вибір XGBoostRanker для вирішення поставленої задачі 25
3.3 Аналітична основа моделі XGBoostRanker 26
3.3.1 Математичне обґрунтування алгоритму градієнтного бустингу 26
3.3.2 Особливості ранжування та функції втрат 27
3.3.3 Параметри моделі та їх вплив на результат 28
3.4. Колаборативна фільтрація 29
3.4.1. Основи колаборативної фільтрації 29
3.4.2 Використання взаємодій між моряками та вакансіями 30
3.4.3. Алгоритми колаборативної фільтрації (KNN) 32
3.5. Аналітична основа алгоритму KNN 33
4 КОМБІНОВАНА МОДЕЛЬ РЕКОМЕНДАЦІЙ 34
4.1 Обмеження контентних та колаборативних моделей 34
4.2 Проблема «холодного старту» та її вплив на рекомендації 34
4.3. Методи об'єднання моделей 35
4.4. Математичне обґрунтування комбінованої моделі 36
5 РЕАЛІЗАЦІЯ ТЕХНОЛОГІЇ ВІДБОРУ ТА РЕКОМЕНДАЦІЇ КАНДИДАТІВ НА ОСНОВІ КОМБІНОВАНОЇ МОДЕЛІ 38
5.1 Підготовка та обробка даних датасетів 38
5.1.1 Попередня обробка даних датасету 39
5.1.2 Формування ознак для моделей 40
5.1.3 Розподіл даних на тренувальні та тестові набори 41
5.2 Реалізація контентної моделі рекомендацій 42
5.2.1 Підготовка до навчання. Розрахунок оптимальних гіперпараметрів.. 42
5.2.2 Навчання контентної моделі 43
5.3 Реалізація колаборативної моделі рекомендацій. Навчання моделі KNN 45
5.4. Комбінування рекомендацій. Зважене об'єднання балів 47
5.5. Інтеграція комбінованої моделі рекомендацій. API 48
5.5.1 Загальна архітектура API 48
5.5.2 Модулі рекомендаційної системи 50

5.5.3 Взаємодія з системою 50
5.5.4 Масштабування та підтримка 51
6 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ТЕХНОЛОГІЇ РЕКОМЕНДАЦІЙ 52
6.1 Тестування моделей та оцінка результатів 52
6.1.1 Оцінка якості контентної моделі 52
6.1.2 Оцінка якості колаборативної моделі 55
6.1.3 Оцінка якості комбінованої моделі 58
6.2 Впровадження та тестування технології комбінованих рекомендацій .. 62
ВИСНОВКИ 64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66
ДОДАТОК А Підготовка та обробка даних 72
ДОДАТОК Б Підбір гіперпараметрів для контентої моделі 77
ДОДАТОК В Навчання моделі XGBoost 79
ДОДАТОК Г Навчання моделі KNN 82
ДОДАТОК Д Збереження та відновлення моделей та трансформерів 84
ДОДАТОК Е Отримання рекомендацій за домогою комбінованої моделі ... 85
ДОДАТОК Ж Бекенд рекомендаційної технології (FastAPI on Python) 88
ДОДАТОК К Формування аналітики для контентої моделі (XGBoost) 92
ДОДАТОК Л Формування аналітики для колаборативної моделі (KNN) 97
ДОДАТОК М Формування аналітики для комбінованої моделі 102
ДОДАТОК Н Запит на виконання кваліфікаційної роботи 106
ДОДАТОК О Довідка про впровадження 108

СПИСОК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ATS – Applicant Tracking Systems

KNN – K-Nearest Neighbors

ML – Machine Learning

MAP – Mean Average Precision

NLP – Natural Language Processing

NDCG – Normalized Discounted Cumulative Gain

ВСТУП

У сучасному світі морські перевезення відіграють важливу роль у міжнародній торгівлі. Ефективна робота суден залежить від злагодженої команди моряків, тому процес найму персоналу є ключовим фактором для успіху будь-якої крюїнгової компанії [1].

Кваліфікований екіпаж гарантує безпечне плавання та запобігає аваріям, що зберігає життя людей, майно та довкілля, при цьому досвідчені моряки забезпечують високу продуктивність судна, скорочуючи час простою та оптимізуючи витрати.

Інформаційні технології відкривають нові можливості для оптимізації цього процесу, оскільки дозволяють автоматизувати багато етапів, зокрема відбір та групування кандидатів за певними критеріями, розраховувати оцінку їхніх компетенцій та виконувати вибір найкращих з них.

Процес підбору кадрів у крюїнгу має свої труднощі та проблеми:

1) трудомісткість перевірки кандидатів – перевірка документів, досвіду роботи, кваліфікацій та рекомендацій займає багато часу та ресурсів;

2) суб'єктивність вибору кандидатів – відсутність єдиних стандартів і критеріїв оцінки робить процес менш прозорим і об'єктивним;

3) похибки через людський фактор – люди можуть допускати помилки під час оцінки документів, інтерв'ю або аналізу інформації про кандидата.

Глобалізація та міжнародний характер судноплавства роблять процес підбору екіпажу складнішим та вимогливішим. Компанії змушені працювати з кандидатами з різних країн, враховувати особливості їхньої освіти, кваліфікації та досвіду роботи. Постійні зміни у законодавстві різних країн також впливають на процес підбору кадрів. Крім того, нестача кваліфікованих моряків є серйозною проблемою, що вимагає нових підходів і рішень для швидкого і ефективного знаходження та рекомендації найкращих кандидатів на вакантні позиції.

Метою даного дослідження є підвищення ефективності процесу підбору кадрів за рахунок зниження суб’єктивності вибору шляхом розробки інформаційної технології компетентністної оцінки персоналу на основі комбінованої рекомендаційної моделі, яка поєднує контентну та колаборативну фільтрацію.

Об'єктом дослідження є процес підбору та рекомендації кандидатів на вакансії у галузі крюїнг-бізнесу.

Предметом дослідження є методи та алгоритми класифікації та ранжування груп кандидатів в ІС «Crewisor» крюїнг-сервісу СТАФФ ЦЕНТР.

Дослідження виконано у відповідності до запиту (див. додаток Н), що також підтверджує його актуальність.

Результати дослідження будуть впроваджені у ІС «Crewisor» компанії СТАФФ ЦЕНТР (див. додаток О).

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) дослідити існуючі методи компетентнісної оцінки моряків;

2) дослідити методи машинного навчання для аналізу даних моряків та виявлення кореляцій між різноманітними параметрами;

3) розробити модель компетентнісної оцінки моряків;

4) навчити розроблену модель на доступних даних, використовуючи різні комбінації корельованих даних;

5) розробити алгоритм порівняння навчених моделей для знаходження найкращих комбінацій параметрів;

6) розробити технологію підбору персоналу на основі навченої моделі та інтегрувати її в існуючу ІС;

7) провести аналіз ефективності розробленої інформаційної технології з урахуванням критеріїв ефективності у порівнянні з існуючими аналогами та вживаними підходами.

1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПІДБОРУ КАНДИДАТІВ НА ВАКАНСІЇ

1.1 Введення в предметну область

Крюїнг, або крю-менеджмент, є комплексом послуг, спрямованих на набір та управління екіпажами для суден. У контексті найму персоналу, крюїнг займає ключову роль у забезпеченні судноплавних компаній кваліфікованими фахівцями, від капітанів до рядових моряків. Цей процес є надзвичайно важливим, оскільки від якості екіпажу залежить безпека, ефективність та загальний успіх морських операцій [2].

1.1.1 Традиційний підхід

Традиційний підхід до найму екіпажу включає ручний процес відбору кандидатів на основі резюме, інтерв'ю та рекомендацій. Крюїнг-агентства або відділи кадрів судноплавних компаній аналізують резюме кандидатів, проводять інтерв'ю та перевіряють рекомендації, щоб оцінити компетенції та відповідність кандидатів вимогам вакансії [3]. Цей метод є трудомістким і вимагає значних людських ресурсів, що може призводити до помилок через суб'єктивність оцінки.

1.1.2 Бази даних та системи управління кандидатами

Використання баз даних та систем управління кандидатами (ATS) значно спрощує процес найму. ATS дозволяють зберігати, організовувати та аналізувати інформацію про кандидатів, автоматизуючи багато етапів процесу відбору. Це включає автоматичне відсіювання нерелевантних резюме, відстеження статусу кандидатів та організацію інтерв'ю. Системи ATS допомагають знизити ризик суб'єктивності та підвищити ефективність процесу найму.

1.1.3 ML та штучний інтелект

Інформаційні технології, зокрема машинне навчання та штучний інтелект, відкривають нові можливості у крюїнгу. Використання алгоритмів машинного навчання дозволяє аналізувати великі обсяги даних про кандидатів, їхній досвід, кваліфікації та результати попередніх робіт. На основі цього аналізу створюються моделі, які можуть прогнозувати успішність кандидатів у певних ролях, забезпечуючи більш об'єктивний і точний відбір.

1.1.4 Важливість компетентнісної оцінки

Компетентнісна оцінка кандидатів є ключовим елементом у процесі крюїнгу. Вона базується на аналізі професійних навичок, досвіду, освіти та інших важливих параметрів кандидатів. Компетентнісна оцінка дозволяє отримати більш об'єктивну картину про кожного кандидата, що допомагає у прийнятті обґрунтованих рішень під час найму.

Традиційні методи оцінки часто залежать від суб'єктивної думки рекрутерів, що може призводити до помилок. Використання інформаційних технологій для компетентнісної оцінки дозволяє автоматизувати цей процес, знижуючи ризик суб'єктивності та підвищуючи точність оцінки.

1.2 Огляд методів оцінки та відбору кандидатів

Точне і надійне оцінювання компетенцій і навичок моряків допомагає гарантувати, що екіпажі мають необхідні знання та вміння для виконання своїх обов'язків у складних умовах морського середовища. У даній секції розглянемо основні проблеми організації процесу оцінки, приклади сучасних методів відбору та оцінки, які використовуються в галузі крюїнг-менеджменту та порівняємо їх ефективність і застосування.

1.2.1 Проблеми організації процесу відбору

Організація процесу оцінки моряків стикається з низкою проблем, які можна поділити на кілька ключових аспектів:

1) суб'єктивність інтерпретації оцінки – традиційні методи оцінки, такі як інтерв'ю та практичні випробування, часто залежать від суб'єктивних оцінок інструкторів чи екзаменаторів. Це може призводити до упереджень і неточностей у визначенні рівня компетентності моряків. Подібні проблеми спостерігаються і в інших галузях, де набір персоналу здійснюється на основі суб'єктивних інтерв'ю, що часто призводить до неправильної оцінки здібностей кандидатів;

2) різноманітність компетенцій – моряки повинні мати широкий спектр навичок, від технічних до соціальних, включаючи навички управління стресом та здатність працювати в команді. Оцінка такої різноманітності компетенцій вимагає комплексного підходу. У сфері IT, наприклад, для оцінки кандидатів використовуються багаторівневі тести, що охоплюють як технічні знання, так і соціальні навички;

3) стандартизація – відсутність стандартизованих методів оцінки може призводити до неоднозначних результатів. Це особливо важливо для міжнародних компаній, де стандарти можуть значно відрізнятися між країнами. У медичній галузі стандартизація оцінок є критично важливою для забезпечення єдиних критеріїв якості послуг незалежно від регіону;

4) адаптивність методів – методи оцінки повинні бути гнучкими та адаптуватися до швидко змінюваних вимог і технологій. Наприклад, використання симуляторів та ВР може значно покращити точність оцінки навичок моряків. Інтерпретація результатів такого способу контролю може змінюватись залежно від потреб бізнесу.

Для зручності диференціювання методів організації процесу оцінки моряків, можна згрупувати їх за критерієм способу оцінювання:

1) методи, засновані на експертних оцінках та психологічних аспектах;
2) методи, засновані на даних;
3) комбіновані методи.

1.2.2 Методи, засновані на експертних та психологічних оцінках

Дані методи включають безпосередню оцінку людей за допомогою опитувань, тестів або експертних оцінок. Вони можуть враховувати як технічні навички, так і психологічні характеристики та поведінкові аспекти.

1) психометричні тести – оцінюють когнітивні здібності, особистісні характеристики та психологічні аспекти;
2) метод Дельфі – використовує експертні оцінки для прогнозування майбутніх подій або оцінки поточних компетенцій.

Перейдемо докладніше до кожного з методів.

Метод Дельфі – це структурований підхід для отримання експертних оцінок і прогнозів, який застосовується для оцінки компетенцій моряків:

1) формулювання питань та вибір експертів – визначення конкретних питань (наприклад, оцінка компетенцій моряків) і вибір незалежних експертів;
2) анонімне опитування – експерти надають свої оцінки анонімно через анкети, що запобігає впливу особистих думок;
3) аналіз і повторні раунди – аналіз відповідей і проведення кількох раундів опитування для досягнення консенсусу;
4) заключення – остаточний аналіз і підготовка звіту з рекомендаціями.

Переваги методу у тому, що зберігається анонімність на незалежність думок, об'єднання знань багатьох експертів підвищує надійність та гнучкість результатів у застосуванні до різних питань. Нажаль, метод потребує значну кількість часу для організації та доволі високі вимоги до якості експертів.

Психометричні тести – оцінюють когнітивні здібності, особистісні характеристики та інші психологічні аспекти, що можуть впливати на продуктивність моряків стресостійкості, здатності до роботи в команді, лідерських якостей та інших важливих характеристик.

Імплементація у процесі оцінки моряків включає оцінку особистісних характеристик для визначення придатності до роботи в морських умовах.

Переваги методу у тому, що забезпечується висока точність об’єктивної оцінки психологічних та когнітивних аспектів. До недоліків належить необхідність спеціалізованих знань для розробки та інтерпретації тестів та потенційна суб’єктивність у відповідях кандидатів.

1.2.3 Методи засновані на даних

Методи, засновані на даних, використовують великі обсяги інформації для аналізу та оцінки кандидатів. Вони базуються на алгоритмах машинного навчання та статистичних моделях.

Потужним елементом цих методів є класифікатори, які дозволяють автоматично категоризувати кандидатів на основі їхніх характеристик.

Розглянемо основні типи класифікаторів:

1) дерева рішень та ансамблеві методи – використовують ієрархічну структуру правил для класифікації кандидатів;

2) методи опорних векторів (SVM) – знаходять оптимальну гіперплощину для розділення класів кандидатів [5];

3) нейронні мережі – моделюють складні нелінійні залежності між характеристиками кандидатів та їхньою придатністю до посади.

Ці класифікатори можуть працювати з різними типами даних, включаючи числові показники, категоріальні змінні та текстову інформацію.

Вони здатні виявляти приховані закономірності у даних та автоматично адаптуватися до нових прикладів.

Перевагами методів заснованих на даних є:

- здатність обробляти великі обсяги інформації;
- виявлення неочевидних зв'язків між характеристиками кандидатів;
- можливість автоматизації процесу оцінки та відбору.

Однак, ці методи також мають певні обмеження:

- потреба у великій кількості якісних даних для навчання;
- складність інтерпретації результатів деяких моделей;
- ризик перенавчання на особливостях навчальної вибірки.

1.2.4 Комбіновані методи

Комбіновані методи поєднують різні підходи до оцінки та відбору кандидатів, дозволяючи отримати більш повну та об'єктивну картину їхніх компетенцій.

Дані методи зазвичай використовують механізми, які дозволяють відслідковувати та інтерпретувати результати оцінювання за допомогою кількох рівнів оцінювачів, наприклад, об'єктивного (машини) та суб'єктивного (екзаменатора або рекрутера).

До основних комбінованих методів належать:

1) Симуляції та VR:

а) створюють реалістичні сценарії для тренування та оцінки в умовах, наближених до реальних;

б) дозволяють перевіряти навички моряків в управлінні судном, навігації, виконанні аварійних процедур та інших критичних завданнях у безпечних умовах;

в) VR-сценарії можуть включати екстремальні погодні умови, аварійні ситуації або інші стресові фактори;

г) надають можливість збирати об'єктивні дані про дії кандидата під час симуляції;

д) потребують участі експерта для якісної оцінки результатів.

2) Багатофакторний аналіз:

а) використовує кілька методів і критеріїв для отримання комплексної оцінки кандидатів;

б) включає використання різноманітних тестів: технічні знання, симульовані вправи, психологічні тести;

в) дозволяє оцінити загальну готовність моряків до роботи в різноманітних умовах;

г) забезпечує всебічну оцінку та високу точність результатів.

Переваги комбінованих методів:

- забезпечують більш повну та об'єктивну оцінку кандидатів;
- дозволяють оцінити як практичні навички, так і психологічні риси;
- знижують ризики обрання недосвідченого кандидата.

Недоліки та складнощі комбінованих методів:

- потребують значних ресурсів для реалізації (обладнання, експерти);
- складність в інтерпретації результатів різних методів оцінки;
- високі вимоги до кваліфікації експертів, які проводять оцінку.

Комбіновані методи є потужним інструментом для комплексної оцінки кандидатів, особливо в таких складних та відповідальних галузях, як морські перевезення. Вони дозволяють поєднати переваги різних підходів, забезпечуючи більш точний та надійний відбір персоналу.

1.2.5 Висновки за методами відбору

Огляд методів оцінки показав, що відбір кандидата – доволі складне завдання, яке можна вирішити різними шляхами.

Кожен з розглянутих методів має свої переваги та недоліки, а також специфічні області застосування.

Для забезпечення комплексної оцінки моряків доцільно використовувати комбінацію різних методів, які доповнюють один одного, створюючі доволі потужні рішення, наприклад:

1) використання моделей класифікації може бути ефективним для аналізу показників, що описують моряків, та групування моряків за подібними характеристиками, що дозволяє виявляти ключові патерни;

2) використання симуляцій та психометричних тестів допомагає оцінити як технічні навички, так і психологічні характеристики, що впливають на ефективність роботи;

3) імплементація багатофакторного аналізу та нейронних мереж дозволяє отримати всебічну картину компетенцій моряків, використовуючи дані з різних джерел та прогнозуючи майбутню продуктивність на основі складних патернів.

Найбільш перспективним підходом є розробка автономної технології, що навчається на історичних даних, пов'язаних з досвідом моряків, їхніми оцінками та іншими доступнми параметрами.

Як зазначено раніше, методи, засновані на даних, здатні забезпечувати високий рівень точності та об'єктивності при оцінці кандидатів, що дозволяє уникнути суб'єктивних помилок і упередженостей, які відбуваються при ручній оцінці екзаменаторами.

1.3 Постановка задачі

Предметна область, що розглядається в цій роботі – процес підбору та рекомендації кандидатів на вакансії у крюїнг-компаніях. Основною метою є розробка інформаційної технології, яка може бути інтегрована у існуючу ІС «Crewisor» компанії СТАФФ ЦЕНТР.

При аналізі предметної області сформульовані основні задачі, які має вирішувати дана технологія:

1) обробка та підготовка даних – застосування методів обробки для автоматичного вилучення ключових ознак з даних, що описують компетенції кандидатів та вимоги вакансій;

2) розробка контентної моделі – створення моделі, яка аналізує профілі кандидатів та вимоги вакансій для оцінки їх відповідності [6];

3) розробка колаборативної моделі – створення моделі, яка аналізує історичні дані про успішні призначення для виявлення прихованих патернів та залежностей [7];

4) розробка механізму зваженого об’єднання результатів моделей – розробка алгоритму, який комбінує результати контентної та колаборативної моделей для формування фінальних рекомендацій методом;

5) формування рейтингових рекомендацій – створення алгоритму, який на основі результатів комбінованої моделі надає рекомендації щодо найбільш підходящих кандидатів для конкретних вакансій;

6) інтеграція з ІС – забезпечення сумісності розробленої технології з існуючою ІС «Crewisor», включаючи API для обміну даними та синхронізації процесів.

В результаті аналізу та детального огляду поставлених задач виявлено 3 основні групи користувачів, які будуть взаємодіяти з розробленою технологією:

1) менеджери екіпажів – вводять дані про нових кандидатів, редагують інформацію та переглядають результати оцінки та рекомендації, надані технологією;

2) рекрутери – здійснюють відбір кандидатів на основі рекомендацій технології та проводять інтерв'ю з відібраними кандидатами;

3) аналітики – аналізують результати роботи технології, налаштовують алгоритми та моделі для покращення точності оцінок;

2 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВІДБОРУ ТА РЕКОМЕНДАЦІЇ КАНДИДАТІВ НА ОСНОВІ КОМБІНОВАНОЇ МОДЕЛІ

2.1 Опис технології комбінованої рекомендації

Для реалізації інформаційної технології відбору та рекомендації кандидатів необхідно обрати методи, що можуть надати можливість трансформації описових даних у формат, який дозволить легше виділяти групи кандидатів та проводити компетентнісну оцінку кожного з них.

Після ретельного аналізу різних підходів, обрано метод комбінування моделей як найбільш перспективний та ефективний.

Технологія включає наступні основні компоненти (див. рис. 2.1):

1) контентна модель – аналізує профілі кандидатів та вимоги вакансій та надає оцінки за рахунок власних характеристик;

2) колаборативна модель – аналізує історичні дані про успішні призначення та надає оцінки за рахунок історії взаємодій;

3) механізм зваженого об'єднання – комбінує оцінки контентної та колаборативної моделей методом динамічного призначення ваги кожній оцінці моделей.

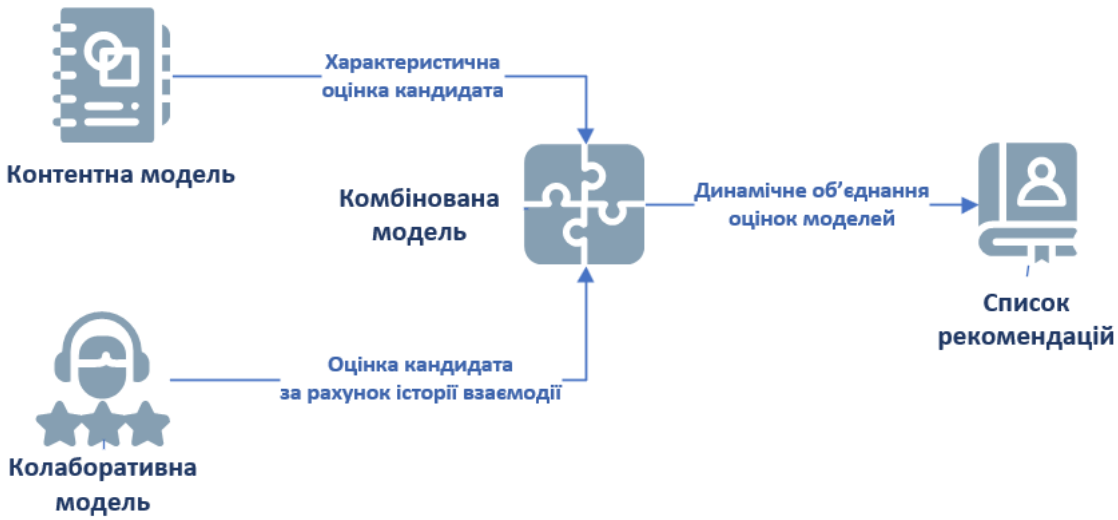

Рисунок 2.1 – Рекомендаційна технологія

Даний підхід дозволяє отримати більш збалансовані та надійні рекомендації, враховуючи як об'єктивні характеристики кандидатів, так і патерни успішних призначень у минулому. Він також забезпечує гнучкість та масштабованість системи, дозволяючи легко адаптувати її до змін у вимогах або появи нових джерел даних.

2.2 Кроки вирішення задач

Для досягнення мети роботи та реалізації технології рекомендації необхідно виконати наступні кроки:

1) Аналіз та обґрунтування методологічної бази:
   а) дослідження існуючих підходів до рекомендаційних систем;
   б) аналіз методів машинного навчання для задач ранжування;
   в) формування вимог до компонентів технології;
   г) аналіз метрик оцінки якості рекомендацій;
   д) визначення критеріїв оцінки ефективності.
2) Підготовка та обробка даних:
   а) збір та систематизація структурованих даних;
   б) обробка та валідація вхідної інформації;
   в) трансформація якісних характеристик у кількісні показники;
   г) нормалізація та стандартизація числових значень;
   д) формування збалансованих наборів даних для навчання;
   е) створення процедур валідації та тестування.
3) Розробка контентної моделі:
   а) формалізація задачі ранжування кандидатів;
   б) вибір та обґрунтування архітектури моделі;
   в) визначення ключових параметрів та гіперпараметрів;
   г) реалізація механізмів навчання та валідації контентної моделі;
   д) оптимізація параметрів моделі;

е) оцінка якості прогнозування;

ж) аналіз важливості характеристик.

4) Розробка колаборативної моделі:

а) формування матриці взаємодій об'єктів;

б) вибір метрик схожості об'єктів;

в) реалізація алгоритмів пошуку найближчих сусідів;

г) налаштування параметрів колаборативної фільтрації;

д) оцінка точності рекомендацій;

е) аналіз результатів та обмежень методу.

5) Створення механізму комбінування моделей:

а) розробка алгоритму динамічної агрегації результатів;

б) визначення правил розрахунку вагових коефіцієнтів;

в) реалізація механізмів нормалізації оцінок;

г) розробка механізмів адаптації ваг;

д) оцінка ефективності комбінованого підходу;

е) аналіз стійкості комбінованих рекомендацій;

ж) створення API для інтеграції.

6) Тестування та впровадження:

а) розробка методології тестування;

б) оцінка продуктивності та масштабованості;

в) аналіз відгуків користувачів;

г) документування результатів впровадження.

У наступних розділах детально розглянуто теоретичні основи використаних методів та моделейь а також практичну реалізацію кожного з перерахованих кроків для досягнення оптимальних результатів у відборі та оцінці кандидатів.

3 ТЕОРЕТИЧНІ ОСНОВИ МЕТОДІВ ТА МОДЕЛЕЙ У ОСНОВІ РЕКОМЕНДАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Контентна фільтрація

Контентна фільтрація є одним із ключових підходів у розробці рекомендаційних систем, який базується на аналізі властивостей об'єктів та користувачів для надання персоналізованих рекомендацій [8].

Принципи роботи контентних моделей полягають у використанні інформації про характеристики об'єктів та уподобання користувачів для прогнозування їхньої взаємодії.

3.1.1 Принципи роботи контентних моделей

Основна ідея контентної фільтрації в контексті підбору кандидатів полягає в тому, щоб рекомендувати вакансії кандидатам (або кандидатів на вакансії), базуючись на схожості їхніх характеристик [9]. Даний підхід досягається шляхом аналізу атрибутів вакансій та кандидатів, створення профілів вакансій на основі їхніх вимог та профілів кандидатів на основі їхніх компетенцій та досвіду.

Контентні моделі працюють за наступними принципами:

1) аналіз властивостей об'єктів – кожен об'єкт описується набором атрибутів або ознак, які можуть бути використані для визначення його характеристик;

2) створення профілю користувача – профіль користувача формується на основі атрибутів об'єктів, з якими він взаємодіяв у минулому;

3) обчислення схожості – використовуючи відповідні метрики, визначається схожість між профілем користувача та потенційними об'єктами для рекомендації;

4) формування рекомендацій – об'єкти з найвищою схожістю рекомендуються користувачу.

Даний підхід дозволяє враховувати індивідуальні вподобання користувачів та надавати релевантні рекомендації навіть у разі відсутності даних про інших користувачів [10].

3.1.2 Використання атрибутів об'єктів для рекомендацій

Атрибути об'єктів є центральним елементом контентної фільтрації. Вони можуть бути як числовими, так і категоріальними, включати текстові описання та інші метадані [11].

У нашому випадку, при підборі моряків на вакансії, атрибутами об'єктів можуть бути:

1) професійні навички та компетенції;
2) досвід роботи на суднах та стаж;
3) наявність сертифікацій та ліцензій;
4) знання мов та додаткові кваліфікації.

Використання цих атрибутів дозволяє будувати моделі, які враховують специфічні вимоги вакансій та особливості кандидатів, що підвищує точність рекомендацій [12].

3.2 Алгоритми та моделі контентної фільтрації

Для реалізації контентної фільтрації використовуються різноманітні алгоритми ML, які можуть ефективно працювати з високорозмірними та різнорідними даними [13]. Серед них популярними та доволі потужними є моделі на основі дерев рішень, такі як Random Forest та XGBoost.

3.2.1 Огляд моделей Random Forest та XGBoost

Random Forest – ансамблевий метод, який використовує множину дерев рішень для підвищення точності прогнозування [14].

Основні характеристики Random Forest:

1) бутстрепінг даних – кожне дерево навчається на випадковій підмножині даних;

2) випадковий вибір ознак – для розгалужень у деревах використовується випадкова підмножина ознак;

3) зменшення варіативності – усереднення результатів множини дерев для отримання меншої кількості гіпотез.

XGBoost (Extreme Gradient Boosting) – вдосконалений алгоритм градієнтного бустингу дерев рішень, який оптимізований для швидкості та продуктивності розрахунків [15].

Основні характеристики XGBoost:

1) покращена обробка пропущених значень;

2) регуляризація для запобігання перенавчанню;

3) паралельне обчислення та оптимізація розрахункових ресурсів.

Обидві моделі надають потужні механізми машинного навчання за допомогою дерев рішень (див. табл. 3.1):

Таблиця 3.1 – Порівняння методів на основі дерев рішень

	Random Forest	XGBoost
Переваги	Стійкість до перенавчання завдяки випадковості в побудові дерев;	Висока точність прогнозування завдяки бустингу;
	Може обробляти великі обсяги даних та працювати з різними типами ознак;	Гнучкість у налаштуванні та можливість роботи з різними функціями втрат

Продовження таблиці 3.1

	Random Forest	XGBoost
Переваги	Може надавати оцінку важливості ознак.	Ефективність та швидкість обчислень;
Недоліки	Може бути повільним при великій кількості дерев;	Складність налаштування гіперпараметрів;
	Менш ефективний у прогнозуванні ранжування порівняно з алгоритмами бустингу	Можливість перенавчання при неправильному налаштуванні регуляризації

3.2.2 Вибір XGBoostRanker для вирішення поставленої задачі

Для задачі підбору кандидатів на вакансії важливо не лише передбачити ймовірність успішного призначення, але й правильно ранжувати кандидатів за ступенем відповідності [16].

XGBoostRanker – спеціалізована версія XGBoost для задач ранжування є оптимальним вибором з наступних причин:

1) підтримка функцій втрат для ранжування, що дозволяє моделі оптимізувати порядковість кандидатів відповідно до їх відповідності вакансії;

2) XGBoostRanker може обробляти великі обсяги даних з високою швидкістю, що важливо для систем з великою кількістю кандидатів;

3) гнучкість у налаштуванні дозволяє встановити гіперпараметри для досягнення оптимальної точності та узагальнюваності;

4) покращена обробка нерівномірних даних, завдяки вбудованим механізмам регуляризації та обробки пропущених значень, вдаліше справляється з реальними даними, які часто є неповними та «зашумленими».

Використання XGBoostRanker у нашій задачі дозволяє ефективно реалізувати контентну фільтрацію з акцентом на точне ранжування кандидатів, що підвищує якість рекомендацій та задовольняє вимоги системи.

3.3 Аналітична основа моделі XGBoostRanker

3.3.1 Математичне обґрунтування алгоритму градієнтного бустингу

Градієнтний бустинг є потужним ансамблевим методом машинного навчання, який комбінує багато слабких моделей (зазвичай дерев рішень) для побудови сильної моделі [17].

Основна ідея полягає в тому, щоб послідовно навчати нові моделі на залишкові помилки попередніх, тим самим мінімізуючи функцію втрат.

Алгоритм градієнтного бустингу опишемо наступними кроками:

1) ініціалізація моделі – початкова модель $F_0(x)$ встановлюється як константа, яка мінімізує функцію втрат $L(y, F(x))$.

2) послідовне навчання – для кожного кроку $m = 1,2,\dots,M$ виконуються наступні дії:

а) обчислення залишків:

$$r_{im} = -\frac{\partial L\big(y_i, F(x_i)\big)}{\partial F(x_i)}\bigg|_{F(x)=F_{m-1}(x)} \tag{3.1}$$

б) навчання слабкого учня $h_m(x)$ на даних $\{(x_i, r_{im})\}$.

в) оновлення моделі:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{3.2}$$

де γ_m – коефіцієнт навчання, який визначається шляхом мінімізації функції втрат:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\big(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\big) \tag{3.3}$$

г) розрахунок кінцевої моделі: після M ітерацій кінцева модель має наступний вигляд:

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x) \tag{3.4}$$

Градієнтний бустинг використовує метод градієнтного спуску в просторові функцій для мінімізації функції втрат, послідовно додаючи нові моделі, що спрямовані на зменшення похибки попередніх моделей [18].

3.3.2 Особливості ранжування та функції втрат

У задачах ранжування метою є впорядкування елементів таким чином, щоб більш релевантні елементи були вище у списку. Даний підхід відрізняється від задач регресії чи класифікації, оскільки необхідно використовувати спеціальні функції втрат, які враховують порядковість об'єктів при відборі.

Існують три основні підходи до навчання ранжувальних моделей:

1) pointwise – розглядає кожен елемент окремо та мінімізує помилку прогнозу його релевантності. Використовуються функції втрат, подібні до регресії чи класифікації [19];

2) pairwise – фокусується на парних порівняннях елементів, намагаючись мінімізувати кількість неправильно впорядкованих пар [20];

3) listwise – працює з повними списками елементів та оптимізує метрики ранжування безпосередньо (наприклад, NDCG) [21]. У цьому підході функція втрат визначається для всього списку елементів, що дозволяє моделі враховувати взаємодії між ними одночасно.

Одним із популярних методів у цьому підході є ListNet, функція втрат якого базується на перехресній ентропії між двома ймовірнісними розподілами над перестановками елементів [22].

Функція втрат для ListNet визначається як:

$$L = -\sum_{i=1}^{n} P(y_i) \log P(f_i) \tag{3.5}$$

де:

- n – кількість елементів у списку;
- $P(y_i)$ – ймовірність релевантності елемента i у вихідних даних;
- $P(f_i)$ – прогнозована ймовірність моделі для елемента i.

Дана функція втрат спрямована на максимізацію ймовірності того, що більш релевантний елемент матиме вищий прогнозований бал, ніж менш релевантний елемент.

3.3.3 Параметри моделі та їх вплив на результат

XGBoostRanker має багато гіперпараметрів, які впливають на продуктивність та узагальнюваність моделі [23]:

1) Параметри, що відповідають за складність моделі:

а) max_depth – максимальна глибина дерев. Більші значення можуть призвести до перенавчання;

б) min_child_weight – мінімальна сума ваг спостережень у листі. Вищі значення роблять модель більш простою.

2) Параметри регуляризації – lambda та alpha – L_2 та L_1 регуляризації відповідно. Дані параметри допомагають запобігти перенавчанню, штрафуючи за складність моделі.

3) Параметри, що контроюють навчання:

а) learning_rate – коефіцієнт навчання, який зменшує вплив кожного додаткового дерева. Менші значення можуть покращити узагальнюваність, але вимагають більшої кількості ітерацій;

б) num_boost_round – кількість дерев у моделі (дуже важливо балансувати між точністю та перенавчанням)

4) Параметри, що контролюють випадковість:

а) subsample – частка вибірки, використаної для навчання кожного дерева. Значення менше 1.0 можуть запобігти перенавчанню;

б) colsample_bytree – частка ознак, використаних при побудові кожного дерева.

5) Параметри, специфічні для ранжування:

а) objective – функція втрат для ранжування, наприклад, «rank»;

б) eval_metric – метрика оцінки моделі, наприклад, NDCG, MAP.

Важливо розуміти вплив параметрів на результат. Перенавчання може виникнути при занадто великій глибині дерев або недостатній регуляризації, що призводить до високої точності на тренувальних даних, але при цьому низької на тестових.

Недонавчання може статися при занадто високих значеннях регуляризаційних параметрів або низькій складності моделі. Коефіцієнт навчання впливає на швидкість конвергенції, тому менші значення вимагають більшої кількості ітерацій, але можуть призвести до кращої узагальнюваності.

Параметри випадковості допомагають зменшити кореляцію між деревами, що покращує узагальнюваність. Вибір функції втрат визначає як модель оптимізує ранжування, тому її правильний вибір функції втрат та метрики оцінки є критичним для задач ранжування [24].

Налаштування цих параметрів за допомогою методів крос-валідації або байєсової оптимізації може значно покращити якість моделі та її здатність до узагальнення на нових даних [25].

3.4. Колаборативна фільтрація

3.4.1. Основи колаборативної фільтрації

Колаборативна фільтрація є одним із найпоширеніших підходів у рекомендаційних системах, який базується на аналізі історії взаємодій між користувачами та відповідними об'єктами [26].

Основна ідея колаборативної фільтрації полягає в тому, що користувачі, які мали схожі вподобання в минулому, ймовірно, матимуть схожі вподобання і в майбутньому. Цей підхід дозволяє системі виявляти неявні зв'язки та патерни в даних, які можуть бути неочевидними при простому аналізі характеристик користувачів або об'єктів.

Принципи роботи колаборативних моделей включають:

1) виявлення схожості між користувачами або об'єктами – моделі визначають, наскільки користувачі або об'єкти схожі один на одного на основі їхньої історії взаємодій;

2) прогнозування уподобань – на основі схожості моделі прогнозують, які об'єкти можуть бути цікавими для користувача;

3) постійне навчання та адаптація – моделі здатні постійно вдосконалюватися, враховуючи нові взаємодії користувачів, що дозволяє системі адаптуватися до оновлень у вподобаннях та появи нових об'єктів.

3.4.2 Використання взаємодій між моряками та вакансіями

Колаборативна фільтрація може стати потужним інструментом у світі крюїнгових агентств та систем найму моряків, використовуючи колективний досвід для надання персоналізованих рекомендацій щодо вакансій.

Даний метод базується на ефективній ідеї: моряки, які мали схожий досвід роботи та кар'єрні траєкторії в минулому, ймовірно, будуть зацікавлені у схожих вакансіях і в майбутньому.

В основі колаборативної фільтрації для крюїнгу лежить аналіз взаємодій між моряками та вакансіями на суднах. Наприклад, система може виявити, що моряки, які працювали на танкерах, часто успішно переходять на роботу на газовози, навіть якщо попередній досвід формально не відповідає вимогам.

Процес роботи колаборативної моделі в крюїнгу можна розділити на кілька ключових етапів:

1) Спочатку система аналізує історію працевлаштувань моряків, такі як посади на різних типах суден, тривалість контрактів, можливі оцінки від попередніх роботодавців.

2) На основі отриманих даних визначається міра схожості між моряками або вакансіями. Для цього можуть використовуватися різні

показчики, наприклад, схожість досвіду роботи на певних типах суден або схожість вимог до кваліфікації.

3) Наступним кроком є прогнозування відповідності моряка вакансії. Система використовує виявлені схожості для передбачення, які вакансії можуть зацікавити конкретного моряка або які моряки найкраще підходять для певної вакансії. Наприклад, якщо система виявила, що моряк А має схожий досвід з моряком Б, і моряк Б успішно працював на певному типі судна, ця вакансія може бути рекомендована моряку А.

Центральним елементом колаборативної фільтрації в крюїнгу є матриця взаємодій. Дана матриця являє собою структуровану форму даних, де кожен рядок представляє моряка, а кожен стовпець - вакансію на судні. На перетині рядків та стовпців знаходяться дані про взаємодію між конкретним моряком та вакансією.

Взаємодії в контексті крюїнгу можуть бути різних типів:

1) явні взаємодії – це ті, які моряк свідомо надає системі, наприклад, подання заявки на конкретну вакансію або оцінка задоволеності після завершення контракту [27];

2) неявні взаємодії є менш прямими, але часто не менш інформативними. Ними можуть бути факти перегляду деталей вакансії, час, проведений на сторінці з описом посади.

Особливістю матриці взаємодій у крюїнгу є її розрідженість. У більшості випадків моряки взаємодіють лише з частиною всіх доступних вакансій, що призводить до того, що більшість елементів матриці залишаються порожніми.

Аналіз даної матриці дозволяє системі виявляти приховані патерни та закономірності у кар'єрних траєкторіях моряків. На основі цього аналізу система може прогнозувати ймовірну відповідність для пар «моряк-вакансія», які ще не розглядалися, що і є основою для надання персоналізованих рекомендацій. Система пропонує моряку вакансії, на які він ще не подавався, але які, згідно з прогнозом моделі, можуть бути для нього цікавими та відповідними його кваліфікації [28].

Такий підхід дозволяє створювати дійсно персоналізовані рекомендації вакансій, які враховують не лише явні кваліфікації моряка, але й неявні патерни кар'єрного розвитку, виявлені на основі аналізу великих обсягів даних про працевлаштування всіх моряків у системі. Дана методика допомагає судноплавним компаніям знайти ідеальних кандидатів для своїх вакансій враховуючи багатокритеріальний аналіз заснований на досвіді.

3.4.3. Алгоритми колаборативної фільтрації (KNN)

Спільна фільтрація на основі пам'яті (memory-based) та на основі моделей (model-based) – два найвідоміші методи колаборативної фільтрації.

До методів спільної фільтрації на основі пам'яті (також називають фільтрацією на основі сусідства) належить метод К-ближніх сусідів (KNN), що визначає найбільш схожих користувачів або об'єкти та використовує їх уподобання для прогнозування [29]. У цьому методі рейтинги об'єктів обчислюються простим способом, шляхом врахування рейтингів людей або речей, що знаходяться поруч. Хоча методи на основі пам'яті є простими і зрозумілими, вони можуть мати проблеми з масштабуванням і запуском в «холодному» середовищі.

Особливості реалізації моделі KNN (на основі близькості):

1) простота реалізації та інтерпретації – алгоритм простий у розумінні та реалізації;

2) потреба в повній матриці взаємодій – ефективність падає при наявності великої кількості пропущених значень;

3) погана масштабованість – алгоритм повільний при великій кількості користувачів або об'єктів.

У реальних системах рекомендацій часто використовуються Комбіновані підходи, які знаходять компроміс між простотою і точністю, використовуючи плюси та можливості обох підходів.

3.5. Аналітична основа алгоритму KNN

Алгоритм KNN є одним із фундаментальних методів машинного навчання, який використовується для задач класифікації та регресії. Його сутність полягає в тому, що прогноз для нового зразка здійснюється на основі інформації про K найближчих зразків з навчального набору даних.

Аналітична основа алгоритму KNN базується на наступних принципах:

- просторова близькість – припускається, що зразки, які знаходяться близько один до одного в просторі ознак, мають схожі властивості або належать до одного класу [30];
- метричний підхід – для визначення близькості використовуються різні метрики відстані або схожості, такі як евклідова відстань, мангеттенська відстань, косинусна схожість тощо;
- локальність – прогноз базується виключно на локальному оточенні кожного зразка, що робить алгоритм не параметричним і гнучким до різних розподілів даних.

Алгоритм KNN працює за наступним алгоритмом:

1) обчислення відстаней – для нового зразка обчислюються відстані до всіх зразків у навчальному наборі даних;
2) вибір K найближчих сусідів – з усього навчального набору обираються K зразків з найменшими відстанями до нового зразка;
3) прогнозування – для задач класифікації: прогнозується клас, який є найбільш поширеним серед K сусідів (метод більшості голосів); для задач регресії прогнозується середнє значення цільової змінної серед K сусідів.

Прогноз для користувача u щодо об'єкта i обчислюється як зважена сума оцінок сусідів [31]:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_k(u)} s(u,v) \cdot r_{vi}}{\sum_{v \in N_k(u)} |s(u,v)|} \tag{3.6}$$

де $N_k(u)$– множина k найближчих сусідів користувача u, $s(u,v)$ – схожість між користувачами u та v, r_{vi} – рейтинг користувача v для об'єкта i.

4 КОМБІНОВАНА МОДЕЛЬ РЕКОМЕНДАЦІЙ

4.1 Обмеження контентних та колаборативних моделей

Контентні та колаборативні фільтраційні моделі мають свої переваги, але при використанні окремо стикаються з певними обмеженнями, що впливає на якість рекомендацій [32]:

1) обмежена перспектива даних – контентні моделі покладаються лише на атрибути об'єктів, ігноруючи вподобання користувачів, тоді як колаборативні моделі використовують лише історію взаємодій, не враховуючи характеристики об'єктів;

2) колаборативні моделі можуть страждати від впливу популярності, рекомендувати лише популярні об'єкти, ігноруючи менш відомі, але потенційно релевантні;

3) вразливість до розрідженості даних – колаборативні моделі погано працюють при недостатній кількості даних про взаємодії, особливо для нових користувачів або об'єктів.

4.2 Проблема «холодного старту» та її вплив на рекомендації

«Холодний старт» – це ситуація, коли система не має достатньої інформації про нових користувачів або об'єкти, що ускладнює надання точних рекомендацій [33]. Вплив даної проблеми на рекомендації наступний:

1) нові користувачі – колаборативні моделі не можуть надати релевантні рекомендації, оскільки немає історії взаємодій;

2) нові об'єкти – об'єкти не будуть рекомендовані, оскільки користувачі ще не взаємодіяли з ними;

3) зниження якості рекомендацій – без достатніх даних система може надавати нерелевантні або занадто загальні рекомендації, що знижує задоволеність користувачів.

Комбіновані моделі допомагають вирішити ці проблеми, комбінуючи переваги контентних та колаборативних підходів для покращення точності та релевантності рекомендацій [34].

4.3. Методи об'єднання моделей

Коли необхідно поєднати результати моделей, можна задіяти методи до комбінування рекомендацій.

Існує кілька методів об'єднання результатів, які підходять у контексті даної задачі для об’єднання результатів контентних та колаборативних моделей у комбінованій системі [35]:

1) зважене об'єднання (weighted-hybridization) – комбінування результатів різних моделей шляхом зваженого сумування їхніх балів або рейтингів; ваги можуть бути фіксованими або адаптивними [36];

2) каскадні моделі (cascade-hybridization) – моделі працюють послідовно; результати однієї моделі подаються як вхідні дані до наступної; наприклад, контентна фільтрує кандидатів, а колаборативна їх ранжує [37];

3) мета-навчання (meta-learning) – створення моделі вищого рівня, яка навчається комбінувати результати декількох базових моделей; використовується машинне навчання для оптимального поєднання [38].

Для нашої системи підбору кандидатів обрано метод зваженого об'єднання. Даний метод забезпечує простоту реалізації та інтерпретації; метод не вимагає складних архітектур або додаткового навчання мета-моделей, а також не потребує значних додаткових ресурсів, що важливо для оперативної роботи системи хоста.

Ваговий коефіцієнт α можна адаптувати для досягнення оптимального балансу між контентною та колаборативною моделями [39]. Даний підхід дозволяє налаштовувати технологію під специфічні потреби бізнесу (надання переваги «перевіреним» клієнтам, чи навпаки, різноманітності та динамічному підбору за рахунок характеристичних параметрів моряків).

4.4. Математичне обґрунтування комбінованої моделі

Для отримання рекомендації комбінована модель обчислює комбінований бал S_{combined} для кожного кандидата шляхом зваженого сумування нормалізованих балів контентної S_{content} та колаборативної $S_{\text{colaborative}}$ моделей:

$$S_{\text{combined}} = \alpha \cdot S_{\text{colaborative}} + (1 - \alpha) \cdot S_{\text{content}} \quad (4.1)$$

де $\alpha \in [0,1]$ – ваговий коефіцієнт, що визначає вклад кожної моделі [40].

Ваговий коефіцієнт α визначає баланс між моделями:

1) $\alpha = 1$: система покладається лише на колаборативну модель; ефективно, коли є багато даних про взаємодії користувачів з об'єктами;

2) $\alpha = 0$: система покладається лише на контентну модель; використовується, коли є багато атрибутів об'єктів, але мало історії взаємодій;

3) $0 < \alpha < 1$: система комбінує обидві моделі; дозволяє компенсувати недоліки однієї моделі за рахунок іншої [41].

Для підвищення адаптивності системи реалізовано механізм динамічного визначення «ступеня довіри» до конкретної моделі через ваговий коефіцієнт α на основі накопиченого досвіду взаємодій.

Коефіцієнт розраховується для кожної вакансії окремо за формулою:

$$\alpha = \frac{n}{n + k} \quad (4.2)$$

де:

- n – кількість накопичених взаємодій для даної вакансії (досвід призначень), що визначає ступінь довіри до колаборативної складової;
- k – параметр регуляризації, що контролює швидкість зміни ступеня довіри між моделями.

Такий підхід забезпечує автоматичну адаптацію ступеня довіри до обсягу накопиченого досвіду та плавну зміну між режимами роботи завдяки параметру k:

- при малому досвіді взаємодій ($n \to 0$) система має більший ступінь довіри до контентної моделі ($\alpha \to 0$);
- при зростанні досвіду взаємодій ($n \to \infty$) поступово підвищується ступінь довіри до колаборативної моделі ($\alpha \to 1$);

При повній відсутності досвіду взаємодій ($n = 0$) система автоматично встановлює максимальний ступінь довіри до контентної моделі, що дозволяє ефективно вирішувати проблему «холодного старту».

Для оцінки якості комбінованої моделі використовуються метрики, що враховують порядок та релевантність рекомендацій [42]:

1) NDCG – вимірює якість ранжування, враховуючи позицію та релевантність кандидатів; значення метрики знаходиться в діапазоні від 0 до 1, де 1 – ідеальне ранжування;
2) MAP – середня точність рекомендацій; використовується для оцінки бінарної релевантності;
3) Precision – показують, яка частка рекомендацій є релевантними.

У нашому випадку релевантність кандидата визначається через фактичний результат призначення на вакансію: кандидат вважається релевантним, якщо він успішно пройшов всі етапи відбору та був призначений на посаду, що забезпечує об'єктивний критерій для оцінки якості рекомендаційної системи.

Використання даних метрик дозволяє об'єктивно оцінити ефективність комбінованої моделі та порівняти її з використанням контентної та колаборативної моделей окремо [43].

5 РЕАЛІЗАЦІЯ ТЕХНОЛОГІЇ ВІДБОРУ ТА РЕКОМЕНДАЦІЇ КАНДИДАТІВ НА ОСНОВІ КОМБІНОВАНОЇ МОДЕЛІ

5.1 Підготовка та обробка даних датасетів

Для розробки інформаційної технології підбору кандидатів сформовано наступні датасети:

1) профілі моряків – містять інформацію про персональні дані, кваліфікації, досвід роботи, сертифікати та інші релевантні атрибути кандидатів. Ці дані є ключовими для контентної моделі, оскільки дозволяють оцінювати відповідність кандидатів вимогам вакансій;

2) профілі вакансій – включають опис посад, вимоги до кандидатів, типи суден, маршрути плавання та інші специфікації. Ці дані використовуються для формування профілю вакансій, з яким порівнюються профілі кандидатів;

3) номінація моряків на вакансії – містять історичну інформацію про призначення моряків на вакансії (проміжні), включаючи факти успішності призначень. Дані датасету є основою для навчання колаборативної моделі, оскільки відображають взаємодії між кандидатами та вакансіями та допомагають прослідкувати патерни вибору кандидатів рекрутерами.

Під час аналізу даних виявлено кілька особливостей:

1) наявність пустих значень – деякі записи містять пропущені значення в ключових полях, таких як досвід роботи або сертифікати, що може призвести до неточностей у моделях та потребує спеціальної обробки;

2) категоріальні ознаки – багато атрибутів, наприклад, посада, тип судна, національність, є категоріальними. Для використання в моделях машинного навчання їх закодовано відповідним чином;

3) дисбаланс класів – у даних про призначення спостерігається дисбаланс між успішними та неуспішними призначеннями, що може вплинути на якість моделей та вимагає застосування методів для роботи з дисбалансованими даними.

Усі вказані особливості коректним чином оброблені перед використанням даних.

5.1.1 Попередня обробка даних датасету

Увесь код підготовки та обробки даних надано у додатку А

Для обробки відсутніх значень застосовано наступні підходи:

1) видалення записів з критичними пропусками – якщо запис має відсутні значення в ключових полях, які неможливо заповнити, він видаляється з набору даних.

2) заповнення нульовими значеннями для числових ознак (наприклад, досвід на кораблі), а для категоріальних – додається спеціальна категорія, наприклад, «None», яка інтерпретується системою як відсутність даних.

Категоріальні ознаки закодовано за допомогою наступних методів

1) кодування лінійних міток (Label Encoding) – для порядкових категорій, де існує природний порядок, використовується цілочисельне кодування (наприклад, доступні типи суден);

2) кодування багатозначних міток (MultiLabelBinarizer) – перетворення багатозначних категоріальних ознак у бінарний формат (наприклад, знання мов, які можуть бути множинами для однієї людини).

Для числових ознак, таких як вік або досвід роботи, застосовано масштабування числових ознак для рівномірного внеску в модель за допомогою StandardScaler.

Зазвичай над даними виконуються 2 типи операцій масштабування:

1) стандартизація – зведення даних до нульового середнього та одиничного стандартного відхилення;

2) нормалізація – приведення значень до діапазону від 0 до 1, що є важливим для алгоритмів, чутливих до масштабу даних;

У нашому випадку масштабування допомагає покращити конвергенцію моделей та підвищує точність прогнозів.

5.1.2 Формування ознак для моделей

Для контентної моделі відібрано ознаки, що мають найбільший вплив на відповідність кандидата вакансії (див. рис. 5.1):

1) наявність необхідних кваліфікацій та сертифікатів;
2) досвід роботи на певних типах суден, кількість рейсів (або років роботи на відповідних суднах);
3) володіння мовами, які вимагаються вакансією;
4) додаткові кваліфікації, наявність спеціальних навичок або тренінгів;
5) відповідність посаді, що вимагаються вакансіями.

```
In [106]: list(importance.keys())

Out[106]: ['sailor_position_encoded',
           'vacancy_position_encoded',
           'experience_years_tanker',
           'experience_years_passenger_ship',
           'experience_years_bulk_carrier',
           'experience_years_container_ship',
           'performance_rating_tanker',
           'performance_rating_passenger_ship',
           'performance_rating_bulk_carrier',
           'performance_rating_container_ship',
           'vacancy_ship_type_encoded',
           'sailor_cert_brm',
           'sailor_cert_ecdis',
           'sailor_cert_erm',
           'sailor_cert_gmdss',
           'sailor_cert_stcw_ii/2',
           'vacancy_cert_ecdis',
           'vacancy_cert_erm',
           'vacancy_cert_gmdss',
           'vacancy_cert_stcw_ii/2',
           'sailor_skill_engine_maintenance',
           'sailor_skill_crew_management',
           'sailor_skill_communication',
           'sailor_skill_safety_procedures',
           'sailor_skill_navigation',
           'vacancy_skill_req_engine_maintenance',
           'vacancy_skill_req_crew_management',
           'vacancy_skill_req_communication',
           'vacancy_skill_req_safety_procedures',
           'vacancy_skill_req_navigation',
           'sailor_language_ukrainian',
           'sailor_language_english',
           'sailor_language_russian',
           'vacancy_language_req_ukrainian',
           'vacancy_language_req_english',
           'vacancy_language_req_russian']
```

Рисунок 5.1 – Ознаки для навчання контентної моделі

Вибір ознак базується на експертному аналізі та кореляційному аналізі з успішністю призначень.

Для колаборативної моделі сформовано матрицю взаємодій:

1) рядки-кандидати, стовпці-вакансії – матриця відображає взаємодії між кандидатами та вакансіями;

2) значення матриці – використовуються бінарні значення (1 – успішне номінування на вакансію, 0 – відсутність номінації) або рейтинги, якщо такі є;

3) розрідженість матриці – матриця є розрідженою, оскільки більшість кандидатів взаємодіяли лише з невеликою кількістю вакансій.

5.1.3 Розподіл даних на тренувальні та тестові набори

При розподілі даних враховано необхідність уникнення перетину даних одного кандидата або вакансії між тренувальним та тестовим наборами:

1) GroupShuffleSplit використовується для розподілу даних з урахуванням груп, де групами є кандидати або вакансії;

2) запобігання витоку даних дозволяє уникнути ситуації, коли модель «бачить» дані про одного й того ж кандидата в обох наборах, що може призвести до переоцінки її точності. Масштабування та кодування ознак виконуються одночасно для тренувального та тестового наборів, щоб забезпечити цілісність параметрів.

Після розподілу даних сформовано матриці ознак та цільових змінних, також збережено механізми масштабування та кодування тегів для досягнення аналогічних результатів при наступних етапах донавчання чи трансформування нових даних у необхідний формат.

5.2 Реалізація контентної моделі рекомендацій

5.2.1 Підготовка до навчання. Розрахунок оптимальних гіперпараметрів.

Для реалізації контентної моделі ранжування обрано алгоритм XGBoostRanker, спеціалізовану версію алгоритму XGBoost, призначену для задач ранжування [44].

Підбір оптимальних значень гіперпараметрів здійснюється шляхом експериментів з використанням крос-валідації та аналізу метрик якості на валідаційному наборі даних. Для пошуку найкращих параметрів реалізовано процес сіткового пошуку (grid-search) з крос-валідацією по групах (див. додаток Б).

Розглядаються наступні ключові гіперпараметри моделі:

1) Параметри структури дерев:
   а) max_depth: (3, 4, 5, 6) – максимальна глибина дерев;
   б) min_child_weight: (1, 3, 5) – мінімальна сума вагів у листі;
   в) gamma: (0.5, 1, 1.5) – зменшення функції втрат для розділення.
2) Параметри формування вибірки:
   а) subsample: (0.6, 0.8, 1.0) – частка записів для навчання кожного дерева у ланцюзі;
   б) colsample_bytree: (0.6, 0.8, 1.0) – частка ознак для навчання кожного дерева у ланцюзі.
3) Параметр навчання – eta (learning rate): (0.01, 0.1, 0.3) – модифікатор швидкості навчання.

Для оцінки якості моделі з кожним набором параметрів використовується метрика NDCG.

В результаті пошуку оптимальних параметрів (з ~1000 комбінацій) отримано наступну конфігурацію:

```
best_params = {
    'objective': 'rank:ndcg',
    'eval_metric': 'ndcg',
    'max_depth': 5,
    'min_child_weight': 1,
    'gamma': 0.5,
    'subsample': 0.8,
    'colsample_bytree': 0.6,
    'eta': 0.3
}
```

Дана конфігурація забезпечила найкраще значення NDCG = 0.7 при валідації, що свідчить про високу якість ранжування кандидатів моделлю. Використання ранньої зупинки (early-stopping) після 10 ітерацій без покращення метрики на валідаційній вибірці дозволяє уникнути перенавчання контентної моделі.

5.2.2 Навчання контентної моделі

Для навчання моделі XGBoostRanker використовуються підготовлені дані, що включають ознаки кандидатів та інформацію про відповідність певним вакансіям [45].

Процес навчання контентної моделі складається з наступних кроків:

1) підготовка даних – сформовано матрицю ознак та цільову змінну; дані розподілено на тренувальний та валідаційний набори з урахуванням уникнення витоку даних;

2) створення груп даних – для задачі ранжування необхідно вказати групи, які визначають набори об'єктів для ранжування; у цьому випадку групами є вакансії, для яких ранжується список кандидатів;

3) конвертація даних у формат DMatrix – використовується спеціальний формат даних DMatrix, оптимізований для роботи з XGBoost; цей формат дозволяє ефективно зберігати дані та пов'язані з ними групи.

DMatrix є спеціалізованим класом даних у XGBoost, який забезпечує швидкий доступ до даних під час навчання. Для задач ранжування необхідно передати інформацію про групи об'єктів за допомогою параметра «group», який містить розміри кожної групи.

Приклад створення DMatrix з групами:

```
import xgboost as xgb
dtrain = xgb.DMatrix(x_train, label=y_train)
dtrain.set_group(group_train)
```

де:

- x_train – матриця ознак тренувального набору;
- y_train – цільова змінна для тренувального набору;
- $group\_train$ – список розмірів груп у тренувальному наборі.

Під час навчання моделі виконується валідація на валідаційному наборі даних для контролю узагальнюваності моделі та запобігання перенавчанню.

Для цього використовується параметр «eval_set», який приймає список з тренувального та валідаційного наборів:

```
eval_set = [(X_train, y_train), (X_valid, y_valid)]
model.fit(X_train, y_train, eval_set=eval_set,
eval_metric='ndcg', verbose=True)
```

Моніторинг метрики NDCG на кожній ітерації дозволяє відстежувати прогрес навчання та використовувати ранню зупинку для припинення навчання, коли метрика на валідаційному наборі перестає покращуватися.

Фінальна модель навчається на повному наборі даних з використанням знайдених оптимальних параметрів протягом 1000 ітерацій або до спрацювання ранньої зупинки.

Повний код навчання моделі XGBoost надано у додатку В.

5.3 Реалізація колаборативної моделі рекомендацій. Навчання моделі KNN

Для реалізації колаборативної моделі рекомендацій розглянуто деякі популярні бібліотеки та алгоритми, які підтримують модель KNN:

1) implicit – спеціалізована бібліотека для неявних взаємодій, оптимізована для великих розріджених матриць;

2) scikit-learn – загальна бібліотека машинного навчання, яка надає інструменти для реалізації алгоритму KNN та прогнозування;

3) surprise – бібліотека, спеціально розроблена для побудови та аналізу систем рекомендацій; підтримує різні алгоритми, включаючи KNN.

При виборі бібліотек враховувалися наступні критерії:

1) підтримка реалізації алгоритму KNN;

2) простота використання та наявність якісних прикладів;

3) продуктивність та масштабованість – здатність ефективно обробляти розріджені матриці взаємодій великого розміру.

Бібліотека scikit-learn надає вичерпну імплементацию алгоритму KNN для виконання у рамках поставленої задачі.

Для побудови колаборативної складової рекомендаційної системи було обрано алгоритм k найближчих сусідів (KNN). Реалізація базується на класі NearestNeighbors з бібліотеки scikit-learn, який забезпечує ефективний механізм пошуку схожих об'єктів у багатовимірному просторі ознак [46].

Ключовим параметром моделі є кількість сусідів (n_neighbors), що визначає скільки найбільш схожих об'єктів буде враховано при рекомендації.

В результаті експериментів встановлено, що оптимальне значення знаходиться в діапазоні 10-15 сусідів. При меншій кількості встановлених сусідів результати стають нестабільними, а при більшій – можуть розмиватися і втрачати релевантність.

Важливим аспектом реалізації є вибір метрики схожості. Для нашої задачі обрано косинусну метрику, яка особливо ефективна при роботі з розрідженими даними, що характерно для матриць взаємодій у рекомендаційних системах. Косинусна відстань обчислює косинус кута між векторами ознак, надаючи нормалізовані значення в діапазоні від -1 до 1, де більші значення відповідають більшій схожості об'єктів.

Процес навчання моделі починається з підготовки даних. Формується матриця взаємодій, де рядки відповідають вакансіям, а стовпці - морякам.

Матриця заповнюється бінарними значеннями на основі історії призначень: 1 – якщо призначення було успішним; 0 – в іншому випадку.

Для забезпечення коректної роботи метрики схожості проводиться нормалізація даних.

Дані розділяються на навчальну (70%) та тестову (30%) вибірки з використанням стратифікації для збереження розподілу класів. Важливим етапом є валідація якості розбиття через перевірку відсутності перетину множин, що забезпечує об'єктивність оцінки якості моделі.

При формуванні рекомендацій для конкретної вакансії спочатку знаходяться K найбільш схожих вакансій з навчального набору. Для цього використовується метод kneighbors(), який повертає як індекси схожих вакансій, так і відповідні їм відстані. Ці відстані потім використовуються як ваги при агрегації результатів.

На основі знайдених схожих вакансій аналізуються успішні призначення та формується зважений список потенційних кандидатів. При цьому враховується як міра схожості між вакансіями, так і успішність попередніх призначень. Фінальним етапом є ранжування кандидатів за спаданням релевантності та формування впорядкованого списку рекомендацій.

Важливо враховувати, що модель потребує значних обчислювальних ресурсів, особливо при великій кількості об'єктів. Використання пам'яті та час пошуку зростають зі збільшенням розміру датасету, що вимагає постійного моніторингу навантаження та можливих оптимізацій.

Для підтримки актуальності рекомендацій необхідне регулярне перенавчання моделі на оновлених даних.

Реалізована KNN модель забезпечує ефективний механізм колаборативної фільтрації, який вдало доповнює контентну складову у фінальній комбінованій системі рекомендацій. Комбінування цих підходів дозволяє отримати більш точні та релевантні рекомендації, враховуючи як схожість профілів, так і успішний досвід попередніх призначень.

Повний код навчання моделі KNN надано у додатку Г. Збереження та завантаження навчених моделей ML надано у додатку Д.

5.4. Комбінування рекомендацій. Зважене об'єднання балів

Функція «combined_recommendations_dynamic_alpha» реалізує алгоритм зваженого об'єднання балів та забезпечує інтеграцію контентної та колаборативної моделей (див. додаток Е).

Інтеграція моделей здійснюється шляхом виклику прогнозуючих функцій обох моделей всередині функції «combined_recommendations».

Функція розрахунку рекомендацій забезпечує:

- уніфікований інтерфейс – обидві моделі приймають однакові вхідні дані та повертають бали у спільному форматі для об’єднання пулів кандидатів;
- синхронізація даних – дані про кандидатів та вакансії актуальні та узгоджені між моделями;
- оптимізація продуктивності – обчислення ефективно використовують потужності хоста, що забезпечує швидке формування рекомендацій кандидатів.

Після комбінування балів результати кандидатів ранжуються за зменшенням комбінованого балу.

Для покращення функціоналу, рекомендації можуть додатково фільтруватись на стороні серверу, наприклад, застосовуватися відсікання кандидатів з балом нижче певного порогу, або недоступних кандидатів для

призначення на момент вибірки. Відсортований список кандидатів може передаватись для подальшої обробки або виводитись користувачу.

5.5. Інтеграція комбінованої моделі рекомендацій. API

5.5.1 Загальна архітектура API

Для інтеграції розробленої комбінованої моделі рекомендацій у існуючі системи обрано архітектурний підхід REST API.

В якості базового фреймворку обрано FastAPI (мова програмування Python), що надає сучасний набір інструментів для створення швидких та надійних веб-сервісів [47].

Основними перевагами даного підходу є:

- асинхронна обробка запитів, що збільшує продуктивність виконання запитів бекендом;
- вбудована валідація даних;
- автоматична OpenAPI документація;
- зручність тестування та інтеграції.

Архітектура API реалізована у вигляді окремих модулів (див. рис. 5.2):

1) моделі даних – описують структури вхідних та вихідних даних;
2) модуль рекомендацій – містить основну бізнес-логіку;
3) контролери – обробляють HTTP запити;
4) обробники помилок – забезпечують коректну обробку виключень;
5) допоміжні модулі – містять утиліти та конфігурації.

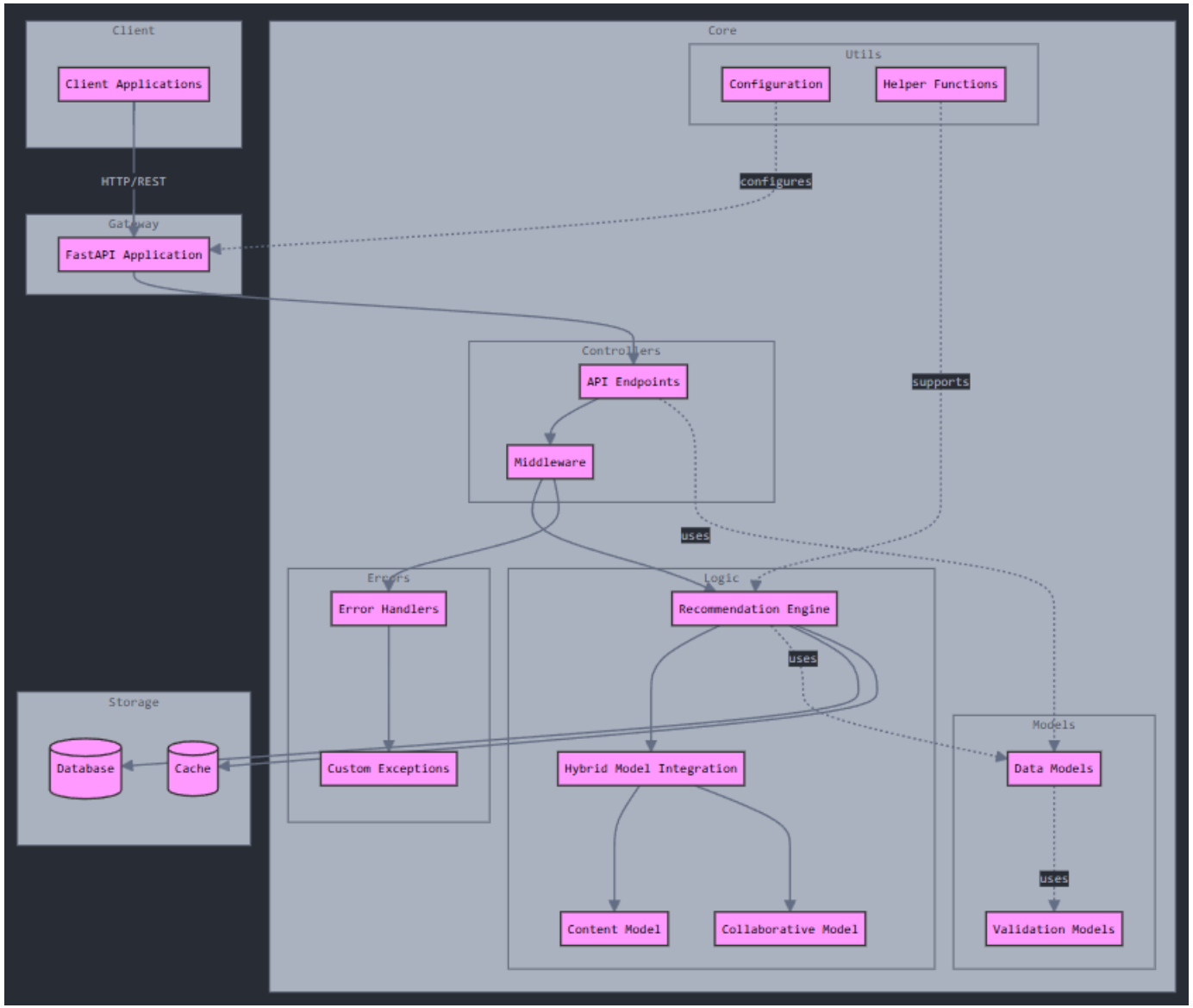

Рисунок 5.2 – Архітектура API

5.5.2 Модулі рекомендаційної системи

Визначено базові структури даних для роботи API:

- Candidate – модель кандидата з полями id, оцінка та ранг;
- RecommendationResponse – структура відповіді з списком рекомендацій;
- ValidationModels – моделі для валідації вхідних параметрів.

Основна логіка роботи з моделями інкапсульована у клас RecommendationSystem, який забезпечує завантаження та ініціалізацію моделей, обробку запитів на рекомендації, балансування між контентною та колаборативною моделями та форматування результатів (див. додаток Ж).

Реалізовано основні ендпоінти API:

- /health – перевірка стану сервісу;
- /recommendations/{vacancy_id} – отримання рекомендацій для вакансії;
- /metrics – статистика роботи системи;

5.5.3 Взаємодія з системою

API підтримує стандартні HTTP методи та використовує JSON для обміну даними.

Основний процес отримання рекомендацій включає наступні кроки:

1) Клієнт надсилає GET запит з ідентифікатором вакансії.
2) Система валідує параметри запиту.
3) Запит передається до рекомендаційного модуля.
4) Формуються рекомендації з використанням комбінованої моделі.
5) Результат форматується та повертається клієнту.

Для взаємодії та тестування на даному етапі API рекомендовано використовувати утиліту Postman або подібні інструменти, поки не буде налаштовано повноцінний потік взаємодії на фронтенді.

5.5.4 Масштабування та підтримка

Система спроектована з урахуванням можливості горизонтального масштабування:

- можливість розгортання у контейнерах Docker;
- підтримка балансування навантаження;
- моніторинг стану та метрик навантаження;
- автоматичне оновлення моделей, що містяться на бекенді.

Під час проектування системи рекомендацій особлива увага приділена можливостям її масштабування та підтримки в промислових умовах. В основу архітектури закладено принцип горизонтального масштабування, що дозволяє системі легко адаптуватися до зростаючих навантажень без втрати продуктивності.

Для забезпеченні надійності системи надано механізм балансування навантаження. Реалізований балансувальник навантаження розподіляє запити між декількома екземплярами API, що дозволяє рівномірно розподіляти навантаження та забезпечувати безперервність роботи навіть при виході з ладу окремих серверів.

Для забезпечення стабільної роботи впроваджено систему моніторингу. Вона збирає широкий спектр метрик: від часу відповіді окремих ендпоінтів до загального використання ресурсів серверів, що дозволяє оперативно виявляти потенційні проблеми та реагувати на них завчасно.

Розроблений API забезпечує надійний та зручний спосіб інтеграції рекомендаційної системи з існуючими бізнес-процесами ІС «Crewisor» компанії СТАФФ ЦЕНТР. Система може бути легко розширена додатковими функціями та адаптована під конкретні потреби користувачів.

6 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ТЕХНОЛОГІЇ РЕКОМЕНДАЦІЙ

6.1 Тестування моделей та оцінка результатів

Повний код аналітичних операцій з отримання метрик та порівняння результатів надано у додатках К, Л та М.

6.1.1 Оцінка якості контентної моделі

Модель XGBoost надає можливість оцінити важливість ознак у моделі. Важливість ознак може бути обчислена різними способами, наприклад, за кількістю використань ознаки в розгалуженнях дерев або за середнім приростом інформації.

Візуалізація важливості ознак надає можлвість демонструвати які ознаки найбільше впливають на прогнозування моделі. Для цього можна використати бібліотеку matplotlib або спеціалізовані функції XGBoost:

Виведення графіків важливості ознак здійснюється за допомогою наступних методів:

```
importance =
     model.get_booster().get_score(importance_type='weight')
###
xgb.plot_importance(model, max_num_features=20)
```

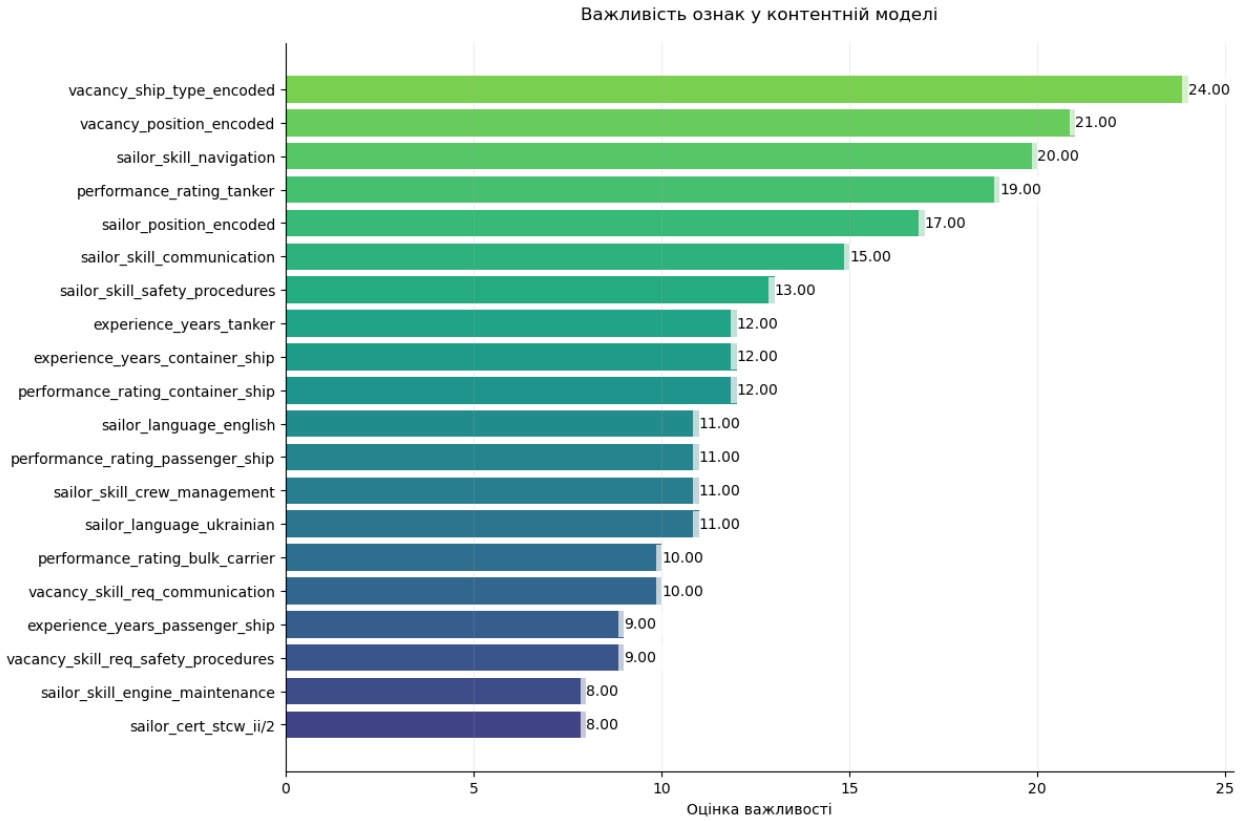

Рисунок 6.1 – Показники важливості ознак при навчання XGBoostRanker

Результати показують, що при відборі кандидатів найбільшу вагу мають практичний досвід роботи та спеціалізація на конкретних типах суден, кваліфікаційні вимоги та мовні навички, а також професійні компетенції:

- досвід роботи на певних типах суден – підтверджує, що досвід є ключовим фактором при відборі кандидатів;
- наявність необхідних сертифікатів – підкреслює важливість кваліфікаці, хоча відбуваються випадки, коли наявність сертифікацій ігнорується (наприклад, до моменту оформлення контракту);
- рівень володіння мовами – вказує на ключому роль у комунікативних навичках при визначенні позиції.

Для аналізу кожної групи (вакансії) здійснюється обчислення метрик NCDG@K, MAP@K та Precision@K (див. рис. 6.2-6.4).

Наведемо порівняльну таблицю метрик при різних значеннях кількості кандидатів K (див. таблицю 6.1):

Таблиця 6.1 – Порівняння значень метрик при різних значеннях K

Метрика	K=5	K=10	K=20	Висновки
NDCG	0.8688	0.8652	0.8589	Відмінне ранжування релевантних кандидатів
MAP	0.5588	0.5155	0.4383	Хороша точність перших рекомендацій
Precision	0.6800	0.6600	0.6033	Стабільна точність навіть при збільшенні списку
Std (NDCG)	0.1311	0.1070	0.0909	Зростання стабільності з розміром списку
Std (MAP)	0.2627	0.2141	0.1619	Висока варіативність на малих списках
Std (Prec.)	0.1973	0.1645	0.1278	Помірна стабільність точності

При K=5 маємо найкращі показники всіх метрик:

- NDCG = 0.8688 (найкраще ранжування).
- Precision = 0.6800 (найвища точність).
- MAP = 0.5588 (найкраща середня точність).

Дані показники означають, що модель найефективніше працює з короткими списками кандидатів, що ідеально підходить для початкового скринінгу на вакансію.

Результати за метриками також можуть говорити про надійність рекомендацій. Std (Standard Deviation) – це стандартне відхилення, статистичний показник, що вимірює розсіювання значень відносно середнього значення.

Розглянемо детальніше в контексті нашої моделі:

- висока стабільність NDCG (std: 0.13 → 0.09) свідчить про надійне ранжування кандидатів;
- зниження варіативності з збільшенням K показує, що модель стає більш передбачуваною при більших списках;
- мінімальні значення Precision не падають нижче 0.6, що гарантує базову якість рекомендацій;

Для покращення якості підбору контенттної моделі рекомендується:

- посилити вагу технічних навичок та сертифікацій;
- додати нові ознаки для покращення точності при K>10;
- впровадити механізми врахування актуальності досвіду.

Модель демонструє найкращі результати при формуванні коротких списків кандидатів (K=5-10), що ідеально відповідає потребам рекрутерів при первинному відборі. Висока стабільність метрик свідчить про надійність рекомендацій та можливість використання моделі як основного інструменту для автоматизації процесу підбору персоналу у випадках, коли не наявна історія взаємодії моряків з вакансіями.

6.1.2 Оцінка якості колаборативної моделі

Модель KNN дозволяє оцінити схожість між вакансіями та моряками на основі їх минулих взаємодій. Аналіз розрідженості матриці взаємодій та розподілу успішних призначень надає можливість оцінити ефективність колаборативного підходу для рекомендацій.

Візуалізація розподілу взаємодій (див. рис. 6.5-6.6) демонструє характер даних, з якими працює модель:

- розподіл взаємодій для вакансій має відносно рівномірний характер з піком в діапазоні 225-275 взаємодій на вакансію;
- розподіл взаємодій для моряків має виражений правосторонній хвіст, що вказує на наявність досвідчених кандидатів з великою кількістю успішних призначень.

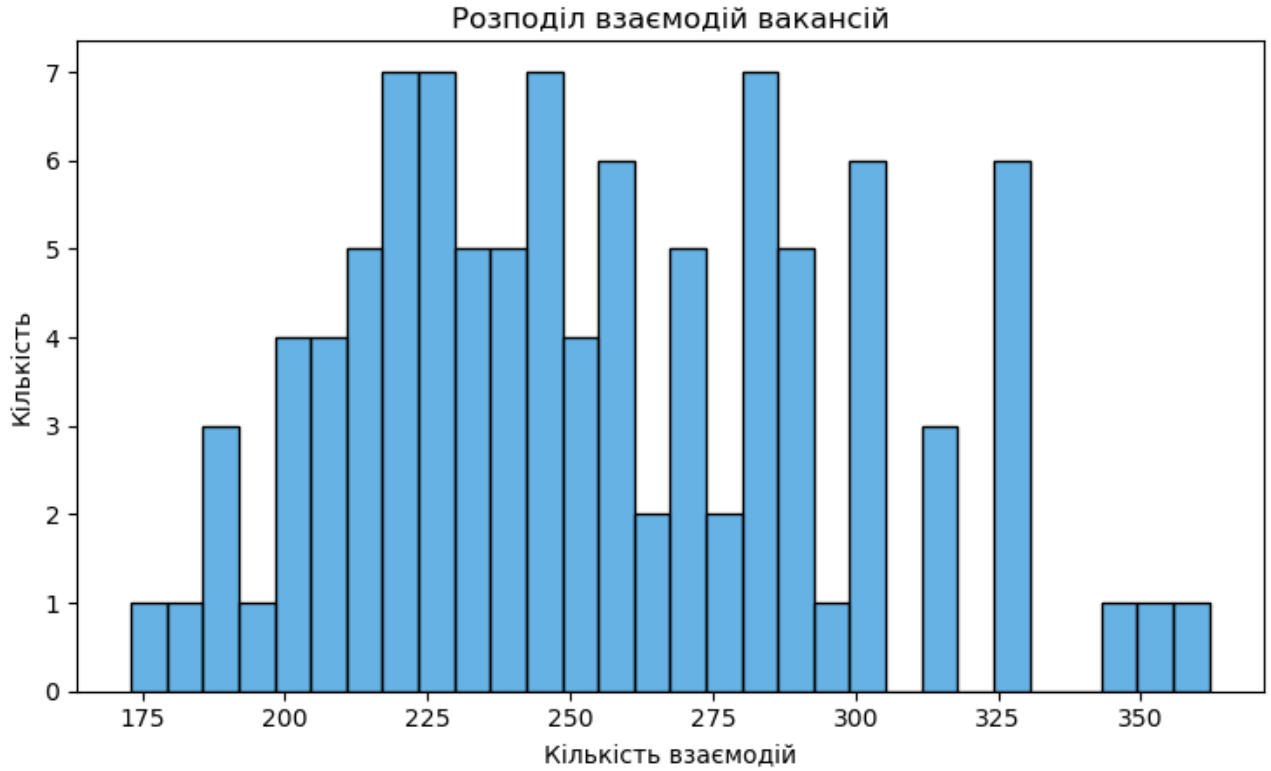

Рисунок 6.5 – Розподіл взаємодій для вакансій

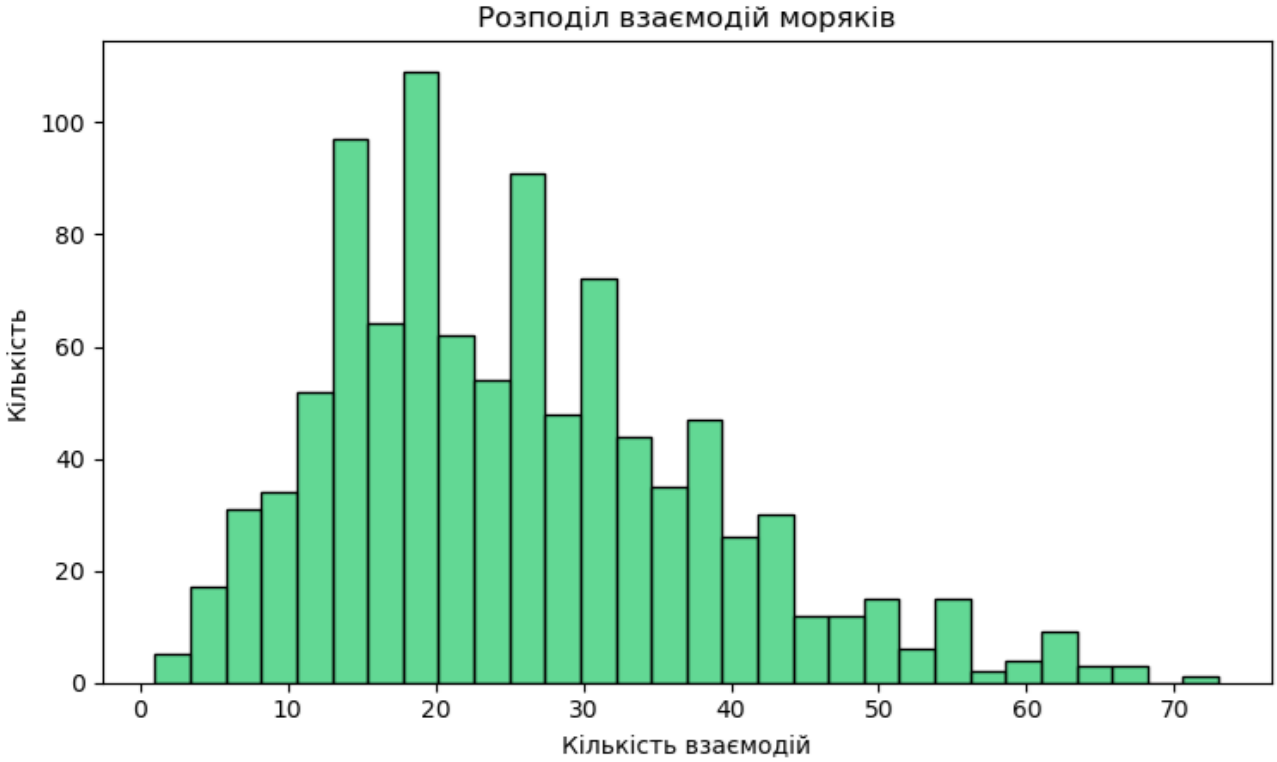

Рисунок 6.6 – Розподіл взаємодій для моряків

Аналіз розрідженості даних показує ключові характеристики матриць взаємодій:

- загальна розрідженість: 74.48% (більше 2/3 можливих комбінацій не мають взаємодій);
- в середньому 255.19 взаємодій на вакансію (демонструє активне використання вакансій);
- в середньому 25.52 взаємодій на моряка (вказує на помірну активність кандидатів).

Наведемо порівняльну таблицю метрик колаборативної моделі при різних значеннях K (див. таблицю 6.2):

Таблиця 6.2 - Порівняння метрик колаборативної моделі

Метрика	K=5	K=10	K=20	Висновки
Precision	0.732	0.809	0.807	Найкраща точність при K=10
Recall	0.015	0.032	0.064	Низьке охоплення релевантних кандидатів
Coverage	0.005	0.010	0.020	Мінімальне охоплення загальної бази
Std (Precision)	0.212	0.146	0.104	Підвищення стабільності з ростом K
Std (Recall)	0.004	0.007	0.011	Стабільно низька варіативність
Std (Coverage.)	0.000	0.000	0.000	Відсутність варіацій у охопленні

При K=10 маємо оптимальні показники точності та охоплення:

- Precision = 0.809 (найвища точність рекомендацій);
- Recall = 0.032 (помірне охоплення релевантних кандидатів);
- Coverage = 0.010 (обмежене охоплення загального пулу).

Дані показники означають, що модель ефективно знаходить найбільш релевантних кандидатів, але має обмежене охоплення бази.

Результати за метриками демонструють стабільність рекомендацій:

- зменшення Std для Precision (0.212 → 0.104) вказує на зростання стабільності при збільшенні K;
- низька варіативність Recall (std < 0.011) свідчить про стабільне охоплення релевантних кандидатів;
- нульова варіативність Coverage показує передбачуване охоплення бази знань.

Для покращення якості колаборативної моделі рекомендується:

- впровадити механізми роботи з «холодним стартом» для нових кандидатів та вакансій (у випадку нашої комбінованої моделі дане рішення надає контентна модель);
- додати часові ваги для надання більшого значення недавнім взаємодіям кандидатів (актуальний досвід).

Модель демонструє високу точність (>80%) при формуванні списків рекомендацій з K=10, що робить її ефективним інструментом для пошуку найбільш надійних кандидатів серед тих, хто має історію успішних призначень. Однак низькі показники метрик Recall та Coverage вказують на необхідність комбінування з іншими підходами для забезпечення повноцінного процесу рекомендацій.

Модель найкраще підходить для формування коротких списків кандидатів з успішною історією призначень, що робить її цінним доповненням до контентної моделі, особливо для вакансій з багатою історією взаємодій.

6.1.3 Оцінка якості комбінованої моделі

Комбінована модель з динамічним параметром $alpha$ дозволяє автоматично адаптувати вагу колаборативної та контентної складових залежно від наявної історії взаємодій.

Аналіз статистики розподілу динамічного параметру довіри показує, що модель значно спирається на колаборативну складову, що вказує на багату історію взаємодій у більшості вакансій (див. рис. 6.7).

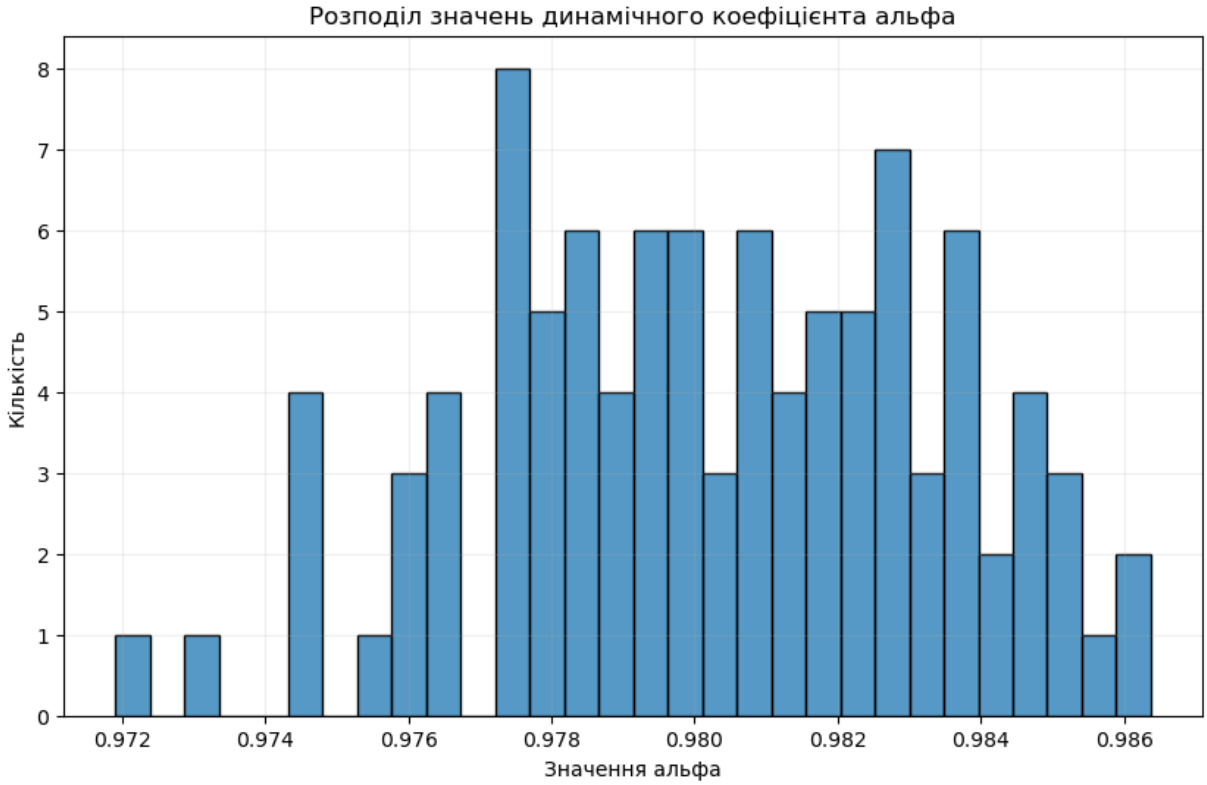

Рисунок 6.7 – Розподіл динамічного коефіцієнта альфа

Наведемо порівняльну таблицю метрик для комбінованої моделі при різних значеннях K (див. табл. 6.3):

Таблиця 6.3 - Порівняння метрик комбінованої моделі

Метрика	K=5	K=10	K=20	Висновки
Precision	0.626 ±0.209	0.592 ±0.171	0.633 ±0.131	Стабільна точність ~60%
Recall	0.012 ±0.004	0.023 ±0.006	0.050 ±0.009	Поступове зростання охоплення
NDCG	0.836 ±0.166	0.835 ±0.132	0.838 ±0.096	Відмінне стабільне ранжування
Coverage	0.005 ±0.000	0.010 ±0.000	0.020 ±0.000	Передбачуване лінійне охоплення

Інтерпретація отриманих результатів для комбінованої моделі:

1) Високе середнє значення $alpha$ (0.95-0.985):

а) показує, що модель сильно покладається на колаборативну складову технології;

б) свідчить про наявність багатої історії взаємодій;

в) більшість вакансій мають >>5 історичних призначень.

2) Низьке стандартне відхилення (~0.003):

а) вказує на стабільність у кількості взаємодій між вакансіями;

б) демонструє однорідність даних у навчальній вибірці; свідчить про усталені патерни взаємодій.

3) Вплив на рекомендації:

а) при alpha ≈ 0.98 рекомендації на 98% базуються на колаборативній складовій;

б) контентна складова виступає як «страховка» при нестачі історичних даних; такий розподіл ваг пояснює стабільність прогнозів.

4) Обмеження поточної реалізації:

а) можливе недостатнє врахування контентної інформації;

б) потенційна проблема при появі нових типів вакансій;

в) ризик надмірної довіри до історичних патернів.

5) Можливості покращення:

а) збільшення параметра k_alpha для встановлення більшої ваги контентної складової;

б) впровадження часових вагів для історичних взаємодій;

в) адаптація формули під специфіку різних типів вакансій з власними ваговими коефіцієнтами для разних моделей.

Розроблена комбінована модель демонструє збалансований підхід до вирішення задачі рекомендації кандидатів на вакансії у крюїнговій компанії. Аналіз результатів показує, що модель успішно поєднує переваги контентного та колаборативного підходів, забезпечуючи при цьому стабільну якість рекомендацій.

Високі показники NDCG (>0.83 для всіх K) свідчать про відмінну здатність моделі ранжувати кандидатів, що є критично важливим для практичного застосування в процесі рекрутингу.

Особливо важливо відзначити стабільність цього показника при різних значеннях K, що демонструє надійність моделі при формуванні списків рекомендацій різного розміру.

Показники Precision демонструють стабільний рівень у діапазоні 59-63%, що є прийнятним для автоматизованої системи рекомендацій, особливо враховуючи складність предметної області та високі вимоги до якості підбору морського персоналу. Важливо відзначити тенденцію до зменшення варіативності цього показника з збільшенням K (std: 0.20 → 0.13), що свідчить про зростання надійності рекомендацій при збільшенні кількості кандидатів.

Однак, модель має певні обмеження, які необхідно враховувати при її практичному застосуванні. Низькі показники Recall (<5%) та Coverage (<2%) вказують на консервативність моделі у виборі кандидатів. Це можна розглядати як компроміс між точністю та різноманітністю рекомендацій, де модель віддає перевагу якості над кількістю.

При цьому оптимальним є використання K=20, що забезпечує найкращий баланс між точністю (63.3%), стабільністю (std=0.131) та охопленням кандидатів.

З точки зору подальшого розвитку, система має значний потенціал для вдосконалення через:

- оптимізацію формули розрахунку динамічного коефіцієнта alpha;
- впровадження механізмів диверсифікації рекомендацій;
- розширення охоплення бази кандидатів без втрати точності.

Загалом, розроблена комбінована модель успішно вирішує поставлену задачу автоматизації підбору морського персоналу, забезпечуючи стабільно високу якість рекомендацій та адаптивність до різних сценаріїв використання.

Хоча існують певні обмеження, вони не є критичними для практичного застосування системи, особливо в контексті її використання як допоміжного інструменту в процесі рекрутингу.

6.2 Впровадження та тестування технології комбінованих рекомендацій

Для оцінки практичної ефективності розробленої системи рекомендацій проведено тестування на реальних даних крюїнгової компанії СТАФФ ЦЕНТР. Використано актуальні записи про кандидатів, вакансії та історію призначень, що дозволило оцінити роботу технології в умовах, максимально наближених до реальних.

Тестування здійснено за наступним сценарієм:

1) Зібрано та підготовлено актуальні дані про кандидатів та вакансії, включаючи нові записи, які не використовувалися при навчанні моделей.

2) Розгорнуто технологію рекомендацій на тестовому сервері компанії у вигляді REST-API, де забезпечено доступ до необхідних баз даних та копій навчених моделей.

3) Формування рекомендацій – система генерувала списки рекомендованих кандидатів для поточних вакансій, використовуючи комбіновану модель.

4) Збір зворотного зв'язку – результати роботи системи передавалися рекрутерам, які оцінювали якість рекомендацій та надавали зворотний зв'язок.

За результатами тестування надано наступні висновки:

- підвищення релевантності рекомендацій – рекрутери відзначили, що запропоновані системою кандидати в більшості випадків відповідали вимогам вакансій, що зменшило час на пошук та оцінку кандидатів;
- зменшення часу на відбір – автоматизація процесу рекомендацій дозволила скоротити час, необхідний для первинного відбору кандидатів, що підвищило ефективність роботи рекрутерів;
- виявлення потенційних кандидатів – система допомогла виявити кандидатів, які раніше могли бути пропущені при ручному відборі, що розширило пул можливих претендентів.

Для оцінки ефективності системи проведено порівняння показників часу «до» та «після» її впровадження за двома основними критеріями. За експертною оцінкою представника відділу рекрутингу А. Жужи отримано наступні результати:

1) швидкість відбору кандидатів – оцінювався середній час, необхідний рекрутеру для формування списку потенційних кандидатів на вакансію. За даними експертної оцінки, цей показник зменшився майже утричі завдяки впровадженню автоматизованих рекомендацій;

2) точність відбору – вимірювалось співвідношення успішних призначень до загальної кількості унікальних пропозицій кандидатів. Згідно з оцінкою експерта, після впровадження технології цей показник збільшився в півтора рази, що демонструє суттєве підвищення якості відбору та ефективності рекомендацій.

Отримані результати експертної оцінки підтверджують значне покращення як швидкості, так і якості процесу підбору кандидатів після впровадження розробленої технології. Інтеграція технології збільшила кількість оброблених вакансій та кандидатів за той самий час, а зниження впливу людського фактора при автоматизації відбору зменшила ризик помилок та суб'єктивності в оцінці кандидатів, що свідчить про підвищення загальної продуктивності відділу рекрутингу.

Застосування зазначених підходів дозволило:

- покращити якість рекомендацій для нових об'єктів, оскільки навіть без історичних даних система надає рекомендації на основі аналізу атрибутів;
- релізувати можливість адаптації системи – швидке врахування нових даних, що забезпечує актуальність рекомендацій та підвищує довіру користувачів до системи;
- підвищити загальну ефективність – вирішення проблеми «холодного старту» сприяло стабільній роботі системи в умовах динамічних змін бази кандидатів та вакансій;

ВИСНОВКИ

В результаті аналізу предметної області сформульовано задачі, що необхідні для досягнення мети проекту, визначено основні особливості функціонування технології підбору та оцінки кандидатів у межах інформаційної системи Crewisor компанії СТАФФ ЦЕНТР, визначено список груп користувачів технології (менеджери екіпажів, рекрутери та аналітики).

Дослідження виконано у відповідності до запиту (див. додаток Н), що також підтверджує його актуальність.

Для інтеграції магістерського дослідження обрано технології машинного навчання, зокрема методи обробки природної мови (NLP), кластеризації та регресії. При аналізі предметної області сформульовані основні задачі, які має вирішувати дана технологія.

Створена технологія успішно реалізує усі функціональні можливості відповідно до задач, визначених на етапі постановки задачі:

1) обробка та підготовка даних про кандидатів;
2) групування кандидатів за схожими характеристиками;
3) оцінка та прогнозування успішності кандидатів;
4) надання рекомендацій щодо відбору кандидатів.

Ефективність розробленої технології підтверджено за двома ключовими критеріями: швидкість відбору кандидатів збільшилась утричі, що характеризує покращення продуктивності системи, а точність відбору зросла в півтора рази за рахунок автоматизації процесу рекомендацій та зниження впливу суб'єктивних факторів.

Реалізація технології має ряд перспектив розвитку, які можуть бути вирішені у майбутніх версіях:

1) вдосконалення методів обробки даних та виявлення ознак за допомогою моделей LLM (аналіз резюме та особистих даних моряків);

2) розширення особистих ознак моряків при контентній та колаборативній фільтрації;

Створена технологія надає міцний фундамент для розвитку нових модулів та підсистем, забезпечуючи ефективний та об'єктивний процес підбору кандидатів у крюїнг-компаніях.

У ході виконання роботи розроблено комбіновану систему рекомендацій для підбору кандидатів на вакансії у крюїнговій індустрії. Технологія поєднує контентну та колаборативну фільтрацію, що дозволяє враховувати як характеристики кандидатів і вимоги вакансій, так і історію взаємодій між кандидатами суміжних вакансій. В рамках технології реалізовано контентну модель на основі XGBoostRanker та колаборативну моделі на основі KNN, які інтегровано у комбіновану рекомендаційну технологію за допомогою зваженого об'єднання балів.

Отримані результати відповідають поставленій меті та задачам. Проведено аналіз алгоритмів рекомендаційних систем, вибрано оптимальні моделі для контентної та колаборативної фільтрації, підготовлено та оброблено дані для навчання, реалізовано та налаштовано моделі, інтегровано їх у комбіновану технологію та проведено її оцінку на реальних даних.

Ефективність розробленої комбінованої технології підтверджено результатами тестування та оцінки на актуальних даних компанії. Технологія продемонструвала високу точність рекомендацій, перевершуючи окремі моделі за метрикою NDCG та Precision@K. Впровадження технології дозволило скоротити час на відбір кандидатів, підвищити продуктивність роботи рекрутерів та зменшити вплив суб’єктивного фактора на процес прийняття рішень.

За результатами даної роботи опубліковано тези на XXI всеукраїнській конференції та виконано тестове впровадження в технологічний процес існуючої ІС «Crewisor» компанії СТАФФ ЦЕНТР (див. додаток О).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джигов Д.Ю., Інформаційна технологія підбору та рекомендації кандидатів на вакансію на основі компетентнісної оцінки / Д.Ю. Джигов, Є.В. Малахов // Інформатика, інформаційні системи та технології: тези доповідей двадцять першої всеукраїнської конференції студентів і молодих науковців. Одеса, 26 квітня 2024 р. - Одеса, 2024.– С. 120-121.
2. Gavalas, D., Syriopoulos, T., RoumIpis, E. (2022). Digital adoption and efficiency in the maritime industry. Journal of Shipping and Trade [Електронний ресурс] – Режим доступу: http://surl.li/ugpbw
3. Wang, P., Mileski, J. (2018). Strategic maritime management as a new emerging field in maritime studies. Maritime Business Review [Електронний ресурс] –. http://surl.li/ugpca
4. Ian Goodfellow, Yoshua Bengio, Aaron Courville (2016). Deep Learning, The MIT Press.
5. Koren, Y., Bell, R., Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. Computer, 42(8), 30-37. [Електронний ресурс] – Режим доступу: https://doi.org/10.1109/MC.2009.263
6. Aggarwal, C.C. (2016). Content-Based Recommender Systems. У Recommender Systems. Springer. [Електронний ресурс] – Режим доступу: https://doi.org/10.1007/978-3-319-29659-3_3 ↩
7. Shi, Y., Larson, M., Hanjalic, A. (2014). Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. ACM Computing Surveys, 47(1), 3. [Електронний ресурс] – Режим доступу: https://doi.org/10.1145/2556270 ↩
8. Ricci, F., Rokach, L., Shapira, B. (2015). Recommender Systems: Introduction and Challenges. In: Recommender Systems Handbook. Springer. [Електронний ресурс] – Режим доступу: https://link.springer.com/chapter/10.1007/978-1-4899-7637-6_1 ↩

9. Lops, P., Gemmis, M. De, Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends. In: Recommender Systems Handbook. Springer. [Електронний ресурс] – Режим доступу: https://link.springer.com/chapter/10.1007/978-0-387-85820-3_3 ↩
10. Pazzani, M.J., Billsus, D. (2007). Content-based recommendation systems. The Adaptive Web. Springer. [Електронний ресурс] – Режим доступу: https://link.springer.com/chapter/10.1007/978-3-540-72079-9_10 ↩
11. Aggarwal, C.C. (2016). Content-Based Recommender Systems. In: Recommender Systems. Springer. [Електронний ресурс] – Режим доступу: https://link.springer.com/chapter/10.1007/978-3-319-29659-3_3 ↩
12. Middleton, S.E., Shadbolt, N.R., De Roure, D.C. (2004). Ontological user profiling in recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 54-88. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/963770.963773
13. Zhang, S., Yao, L., Sun, A., Tay, Y. (2019). Deep Learning based Recommender System: A Survey and New Perspectives. ACM Computing Surveys (CSUR), 52(1), 1-38. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/3285029 ↩
14. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. [Електронний ресурс] – Режим доступу: https://link.springer.com/article/10.1023/A:1010933404324 ↩
15. Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/2939672.2939785 ↩
16. Liu, T.Y. (2009). Learning to Rank for Information Retrieval. Foundations and Trends® in Information Retrieval, 3(3), 225-331. [Електронний ресурс] – Режим доступу: https://www.nowpublishers.com/article/Details/INR-016 ↩

17. Friedman, J.H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189-1232. [Електронний ресурс] – Режим доступу: https://statweb.stanford.edu/~jhf/ftp/trebst.pdf ↩
18. Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning. Springer Series in Statistics. [Електронний ресурс] – Режим доступу: https://web.stanford.edu/~hastie/ElemStatLearn/ ↩
19. Li, H. (2011). A Short Introduction to Learning to Rank. IEICE Transactions on Information and Systems, 94(10), 1854-1862. [Електронний ресурс] – Режим доступу: https://www.ieice.org/transactions/e90-d/6/1116.pdf ↩
20. Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G. (2005). Learning to Rank Using Gradient Descent. Proceedings of the 22nd International Conference on Machine Learning (ICML). [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/1102351.1102363 ↩
21. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H. (2007). Learning to Rank: From Pairwise Approach to Listwise Approach. Proceedings of the 24th International Conference on Machine Learning (ICML). [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/1273496.1273513 ↩
22. Xu, J., Li, H. (2007). Adarank: a boosting algorithm for information retrieval. Proceedings of the 30th Annual International ACM SIGIR Conference. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/1277741.1277809 ↩
23. Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/2939672.2939785 ↩
24. Wang, L., Lin, Z., Luo, J. (2014). Learning to Rank Using Pairwise Documents with Multi-modal Features for Multimedia Information Retrieval. IEEE Transactions on Multimedia, 16(6), 1692-1705. [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/document/6809183 ↩

25. Bergstra, J., Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13(Feb), 281-305. [Електронний ресурс] – Режим доступу: http://jmlr.org/papers/v13/bergstra12a.html ↩
26. Su, X., Khoshgoftaar, T.M. (2009). A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence. [Електронний ресурс] – Режим доступу: https://www.hindawi.com/journals/aai/2009/421425/ ↩
27. Schafer, J.B., Konstan, J.A., Riedl, J. (2007). Recommender Systems in E-Commerce. Proceedings of the 1st ACM Conference on Electronic Commerce. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/336992.337035 ↩
28. Koren, Y., Bell, R., Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. IEEE Computer, 42(8), 30-37. [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/document/5197422 ↩
29. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. Proceedings of the 22nd Annual International ACM SIGIR Conference. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/312624.312682 ↩
30. Breese, J.S., Heckerman, D., Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1301.7363 ↩
31. Aggarwal, C.C. (2016). Recommender Systems: The Textbook. Springer. [Електронний ресурс] – Режим доступу: https://link.springer.com/book/10.1007/978-3-319-29659-3 ↩
32. Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370. [Електронний ресурс] – Режим доступу: https://link.springer.com/article/10.1023/A:1021240730564 ↩

33. Lam, C.P., Frankowski, D., Riedl, J. (2008). Addressing Cold-Start in Recommender Systems. Proceedings of the 2008 ACM Conference on Recommender Systems. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/1454008.1454014 ↩
34. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M. (2002). Methods and Metrics for Cold-Start Recommendations. Proceedings of the 25th Annual International ACM SIGIR Conference. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/564376.564421 ↩
35. Adomavicius, G., Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749. [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/document/1423975 ↩
36. Son, L.H. (2016). Dealing with the New User Cold-Start Problem in Recommender Systems: A Comparative Review. Information Systems, 58, 87-104. [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/pii/S0306437916000542 ↩
37. Burke, R. (2007). Hybrid Web Recommender Systems. In: The Adaptive Web. Springer. [Електронний ресурс] – Режим доступу: https://link.springer.com/chapter/10.1007/978-3-540-72079-9_12 ↩
38. Zhang, S., Yao, L., Sun, A., Tay, Y. (2019). Deep Learning based Recommender System: A Survey and New Perspectives. ACM Computing Surveys, 52(1), 1-38. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/3285029 ↩
39. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G. (2010). Recommender Systems: An Introduction. Cambridge University Press. [Електронний ресурс] – Режим доступу: https://www.cambridge.org/core/books/recommender-systems/09C76C8C5C9E8A9B2E0E7D3B01FE0A66 ↩

40. Lin, C.J., Juan, Y., Wei, D. (2016). A Survey on Learning to Rank for Recommender Systems. Workshop on Learning from User Interactions. [Електронний ресурс] – Режим доступу: https://www.csie.ntu.edu.tw/~cjlin/papers/LTRSurvey.pdf ↩
41. Cremonesi, P., Koren, Y., Turrin, R. (2010). Performance of Recommender Algorithms on Top-N Recommendation Tasks. Proceedings of the 4th ACM Conference on Recommender Systems. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/1864708.1864721 ↩
42. Gunawardana, A., Shani, G. (2009). A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. Journal of Machine Learning Research, 10(Dec), 2935-2962. [Електронний ресурс] – Режим доступу: http://jmlr.org/papers/v10/gunawardana09a.html ↩
43. Yang, S., Steck, H., McAuley, J. (2012). Circle-Based Recommendation in Online Social Networks. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/10.1145/2339530.2339658
44. Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. У Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794, ACM. [Електронний ресурс] – Режим доступу: https://doi.org/10.1145/2939672.2939785 ↩
45. XGBoost Documentation – Learning to Rank. [Електронний ресурс] – Режим доступу: https://xgboost.readthedocs.io/en/latest/tutorials/learning_to_rank.html ↩
46. Scikit-learn documentation – Nearest Neighbors. [Електронний ресурс] – Режим доступу: https://scikit-learn.org/stable/modules/neighbors.html ↩
47. FastAPI: framework, high performance, easy to learn, fast to code, ready for production. [Електронний ресурс] – Режим доступу: https://fastapi.tiangolo.com/ ↩

ДОДАТОК А

Підготовка та обробка даних

```
# Імпорт бібліотек та завантаження даних
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.preprocessing import LabelEncoder,
MultiLabelBinarizer, StandardScaler
from sklearn.model_selection import GroupShuffleSplit
from sklearn.metrics import ndcg_score
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
import pickle

# Завантаження даних з файлів .pkl
sailors_df = pd.read_pickle('./Data/sailors_data__improved.pkl')
vacancies_df =
pd.read_pickle('./Data/vacancies_data__improved.pkl')
assignments_df =
pd.read_pickle('./Data/assignments_data__improved.pkl')

# Попередня обробка даних
# Об'єднання даних про моряків та призначення
merged_df = assignments_df.merge(sailors_df, on='sailor_id',
how='left')
merged_df = merged_df.merge(vacancies_df, on='vacancy_id',
how='left')

# Перейменування колонок у snake_case
merged_df.columns = [col.lower().replace(' ', '_') for col in
merged_df.columns]

# Вивід полів датасету
merged_df

# Кодування посади моряка та вакансії
le_position = LabelEncoder()
positions = pd.concat([merged_df['position_x'],
merged_df['position_y']]).unique()
le_position.fit(positions)

merged_df['sailor_position_encoded'] =
le_position.transform(merged_df['position_x'])
merged_df['vacancy_position_encoded'] =
le_position.transform(merged_df['position_y'])

# Створення ознак для досвіду на типах суден
def extract_ship_experience(ship_experience_dict, ship_types):
```

```
    experience_years = {}
    performance_ratings = {}
    for ship_type in ship_types:
        if ship_type in ship_experience_dict:
            experience_years[ship_type] =
ship_experience_dict[ship_type]['years']
            performance_ratings[ship_type] =
ship_experience_dict[ship_type]['rating']
        else:
            experience_years[ship_type] = 0
            performance_ratings[ship_type] = 0
    return experience_years, performance_ratings

# Отримуємо список усіх типів суден
all_ship_types = list(set(merged_df['ship_type'].unique()) |
set(ship_type for d in merged_df['ship_type_experience'] for
ship_type in d.keys()))

# Створення ознак для досвіду на суднах
sailor_experience_years = []
sailor_performance_ratings = []

for index, row in merged_df.iterrows():
    exp_years, perf_ratings =
extract_ship_experience(row['ship_type_experience'],
all_ship_types)
    sailor_experience_years.append(exp_years)
    sailor_performance_ratings.append(perf_ratings)

experience_years_df = pd.DataFrame(sailor_experience_years)
experience_years_df.columns = ['experience_years_' +
col.lower().replace(' ', '_') for col in
experience_years_df.columns]

performance_ratings_df =
pd.DataFrame(sailor_performance_ratings)
performance_ratings_df.columns = ['performance_rating_' +
col.lower().replace(' ', '_') for col in
performance_ratings_df.columns]

# Додаємо до основного датафрейму
merged_df = pd.concat([merged_df.reset_index(drop=True),
experience_years_df, performance_ratings_df], axis=1)

# merged_df['vacancy_position_encoded'].unique()
# merged_df['sailor_position_encoded'].unique()
merged_df['ship_type_experience']

# Кодування типу судна вакансії
le_ship_type = LabelEncoder()
all_ship_types = merged_df['ship_type'].unique()
le_ship_type.fit(all_ship_types)
```

```
merged_df['vacancy_ship_type_encoded'] =
le_ship_type.transform(merged_df['ship_type'])

# Кодування сертифікатів моряка
mlb_certifications = MultiLabelBinarizer()
sailor_certs = merged_df['certifications']
mlb_certifications.fit(sailor_certs)

sailor_certs_encoded =
mlb_certifications.transform(sailor_certs)
sailor_cert_columns = ['sailor_cert_' + c.lower().replace(' ',
'_') for c in mlb_certifications.classes_]
sailor_cert_df = pd.DataFrame(sailor_certs_encoded,
columns=sailor_cert_columns)

merged_df = pd.concat([merged_df.reset_index(drop=True),
sailor_cert_df], axis=1)

# Кодування необхідних сертифікатів вакансії
vacancy_certs = merged_df['required_certifications']
vacancy_certs_encoded =
mlb_certifications.transform(vacancy_certs)
vacancy_cert_columns = ['vacancy_cert_' + c.lower().replace(' ',
'_') for c in mlb_certifications.classes_]
vacancy_cert_df = pd.DataFrame(vacancy_certs_encoded,
columns=vacancy_cert_columns)

merged_df = pd.concat([merged_df.reset_index(drop=True),
vacancy_cert_df], axis=1)

# Отримуємо список усіх навичок
all_skills = list(set(skill for d in
merged_df['skill_competencies'] for skill in d.keys()) |
set(skill for d in merged_df['required_skill_competencies'] for
skill in d.keys()))

# Створення ознак для компетенцій моряка
sailor_skill_competencies = []
for index, row in merged_df.iterrows():
    competencies = {}
    for skill in all_skills:
        competencies[skill] =
row['skill_competencies'].get(skill, 0)
    sailor_skill_competencies.append(competencies)

sailor_competencies_df = pd.DataFrame(sailor_skill_competencies)
sailor_competencies_df.columns = ['sailor_skill_' +
col.lower().replace(' ', '_') for col in
sailor_competencies_df.columns]

merged_df = pd.concat([merged_df.reset_index(drop=True),
sailor_competencies_df], axis=1)
```

```
# Створення ознак для вимог до компетенцій вакансії
vacancy_skill_requirements = []
for index, row in merged_df.iterrows():
    requirements = {}
    for skill in all_skills:
        requirements[skill] =
row['required_skill_competencies'].get(skill, 0)
    vacancy_skill_requirements.append(requirements)

vacancy_requirements_df =
pd.DataFrame(vacancy_skill_requirements)
vacancy_requirements_df.columns = ['vacancy_skill_req_' +
col.lower().replace(' ', '_') for col in
vacancy_requirements_df.columns]

merged_df = pd.concat([merged_df.reset_index(drop=True),
vacancy_requirements_df], axis=1)

# Отримуємо список усіх мов
all_languages = list(set(lang for d in merged_df['languages']
for lang in d.keys()) | set(lang for d in
merged_df['language_requirements'] for lang in d.keys()))

# Кодування рівня володіння мовами моряка
language_levels_dict = {'Basic': 1, 'Intermediate': 2,
'Advanced': 3, 'Fluent': 4}

sailor_language_levels = []
for index, row in merged_df.iterrows():
    lang_levels = {}
    for lang in all_languages:
        level = row['languages'].get(lang, 'Basic')
        lang_levels[lang] = language_levels_dict.get(level, 0)
    sailor_language_levels.append(lang_levels)

sailor_languages_df = pd.DataFrame(sailor_language_levels)
sailor_languages_df.columns = ['sailor_language_' +
col.lower().replace(' ', '_') for col in
sailor_languages_df.columns]

merged_df = pd.concat([merged_df.reset_index(drop=True),
sailor_languages_df], axis=1)

# Кодування вимог до мов вакансії
vacancy_language_levels = []
for index, row in merged_df.iterrows():
    lang_levels = {}
    for lang in all_languages:
        level = row['language_requirements'].get(lang, 'Basic')
        lang_levels[lang] = language_levels_dict.get(level, 0)
    vacancy_language_levels.append(lang_levels)

vacancy_languages_df = pd.DataFrame(vacancy_language_levels)
```

```
vacancy_languages_df.columns = ['vacancy_language_req_' +
col.lower().replace(' ', '_') for col in
vacancy_languages_df.columns]

merged_df = pd.concat([merged_df.reset_index(drop=True),
vacancy_languages_df], axis=1)

# Формування списку ознак
feature_columns = []

# Додаємо закодовані посади
feature_columns.extend(['sailor_position_encoded',
'vacancy_position_encoded'])

# Додаємо досвід та рейтинг на типах суден
feature_columns.extend(experience_years_df.columns.tolist())
feature_columns.extend(performance_ratings_df.columns.tolist())

# Додаємо закодований тип судна вакансії
feature_columns.append('vacancy_ship_type_encoded')

# Додаємо закодовані сертифікати
feature_columns.extend(sailor_cert_df.columns.tolist())
feature_columns.extend(vacancy_cert_df.columns.tolist())

# Додаємо компетенції
feature_columns.extend(sailor_competencies_df.columns.tolist())
feature_columns.extend(vacancy_requirements_df.columns.tolist())

# Додаємо мовні навички
feature_columns.extend(sailor_languages_df.columns.tolist())
feature_columns.extend(vacancy_languages_df.columns.tolist())

# Масштабування числових ознак
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Вибираємо числові ознаки для масштабування
numeric_features = experience_years_df.columns.tolist() +
performance_ratings_df.columns.tolist() +
sailor_competencies_df.columns.tolist() +
vacancy_requirements_df.columns.tolist() +
sailor_languages_df.columns.tolist() +
vacancy_languages_df.columns.tolist()

merged_df[numeric_features] =
scaler.fit_transform(merged_df[numeric_features])
```

ДОДАТОК Б

Підбір гіперпараметрів для контентої моделі

```
# Пошук кращих гіперпараметрів для навчання

import xgboost as xgb
import numpy as np
from sklearn.model_selection import GroupKFold
from sklearn.preprocessing import StandardScaler

# Assuming X is your feature matrix, y is target variable,
# and groups is an array specifying the group (e.g., query ID)
for each sample

# 1. Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. Prepare the DMatrix with group information
dmatrix = xgb.DMatrix(X_scaled, label=y)
group_sizes = np.bincount(groups)
dmatrix.set_group(group_sizes)

# 3. Set up GroupKFold for cross-validation
group_kfold = GroupKFold(n_splits=5)

# 4. Define parameter grid for hyperparameter tuning
param_grid = {
    'max_depth': [3, 4, 5, 6],
    'min_child_weight': [1, 3, 5],
    'gamma': [0.5, 1, 1.5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'eta': [0.01, 0.1, 0.3]
}

# 5. Function to perform cross-validation
def xgb_ranking_cv(params):
    cv_scores = []
    for train_idx, val_idx in group_kfold.split(X_scaled, y,
groups):
        dtrain = xgb.DMatrix(X_scaled[train_idx],
label=y[train_idx])
        dtrain.set_group(np.bincount(groups[train_idx]))
        dval = xgb.DMatrix(X_scaled[val_idx], label=y[val_idx])
        dval.set_group(np.bincount(groups[val_idx]))

        model = xgb.train(
            params,
            dtrain,
            num_boost_round=100,
            evals=[(dval, 'eval')],
```

```
            early_stopping_rounds=10,
            verbose_eval=True
        )
        cv_scores.append(model.best_score)
    return np.mean(cv_scores)

# 6. Perform grid search
best_params = None
best_score = float('-inf')

for max_depth in param_grid['max_depth']:
    for min_child_weight in param_grid['min_child_weight']:
        for gamma in param_grid['gamma']:
            for subsample in param_grid['subsample']:
                for colsample_bytree in
param_grid['colsample_bytree']:
                    for eta in param_grid['eta']:
                        params = {
                            'objective': 'rank:ndcg',
                            'eval_metric': 'ndcg',
                            'max_depth': max_depth,
                            'min_child_weight':
min_child_weight,
                            'gamma': gamma,
                            'subsample': subsample,
                            'colsample_bytree':
colsample_bytree,
                            'eta': eta
                        }
                        score = xgb_ranking_cv(params)
                        if score > best_score:
                            best_score = score
                            best_params = params

print(«Best parameters:», best_params)
print(«Best NDCG score:», best_score)

# 7. Train the final model with the best parameters
final_model = xgb.train(
    best_params,
    dmatrix,
    num_boost_round=1000,
    early_stopping_rounds=10,
    verbose_eval=True
)

# 8. Feature importance
importance = final_model.get_score(importance_type='gain')
sorted_importance = sorted(importance.items(), key=lambda x:
x[1], reverse=True)
print(«Top 10 important features:»)
for feature, score in sorted_importance[:10]:
    print(f»{feature}: {score}»)
```

ДОДАТОК В

Навчання моделі XGBoost

```
# Підготовка даних для моделі ранжування

# Цільова змінна
y = merged_df['assignment_success']  # 1 - успішне призначення,
0 - неуспішне
# Ознаки
X = merged_df[feature_columns]

# Переконуємося, що немає пропущених значень
X = X.fillna(0)

# Групи (vacancy_id)
groups = merged_df['vacancy_id']

# 3.2. Розділення даних з урахуванням груп
# Ініціалізація сплітера
gss = GroupShuffleSplit(n_splits=1, test_size=0.3,
random_state=42)

# Розділення даних
train_idx, test_idx = next(gss.split(X, y, groups))

# Формування тренувального та тестового наборів
X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]
groups_train = groups.iloc[train_idx]
groups_test = groups.iloc[test_idx]

# 3.3. Підготовка груп для DMatrix

# Тренувальний набір
train_data = X_train.copy()
train_data['label'] = y_train
train_data['group'] = groups_train

# Сортуємо за групами
train_data =
train_data.sort_values('group').reset_index(drop=True)

# Отримуємо розміри груп
group_train_sizes =
train_data.groupby('group').size().to_numpy()

# Тестовий набір
test_data = X_test.copy()
test_data['label'] = y_test
test_data['group'] = groups_test
```

```
# Сортуємо за групами
test_data =
test_data.sort_values('group').reset_index(drop=True)

# Отримуємо розміри груп
group_test_sizes = test_data.groupby('group').size().to_numpy()

print(«Розмір X_train:», X_train.shape)
print(«Розмір X_test:», X_test.shape)
print(«Кількість унікальних груп у тренувальному наборі:»,
groups_train.nunique())
print(«Кількість унікальних груп у тестовому наборі:»,
groups_test.nunique())

# Переконаюсь, що групи не перетинаються між наборами:
common_groups = np.intersect1d(groups_train.unique(),
groups_test.unique())
print(«Кількість спільних груп між тренувальним та тестовим
наборами:», len(common_groups))

# 3.4. Створення DMatrix

# Тренувальний набір
dtrain = xgb.DMatrix(train_data[feature_columns],
label=train_data['label'])
dtrain.set_group(group_train_sizes)

# Тестовий набір
dtest = xgb.DMatrix(test_data[feature_columns],
label=test_data['label'])
dtest.set_group(group_test_sizes)

print(«Сума розмірів груп у тренувальному наборі:»,
group_train_sizes.sum())
print(«Кількість рядків у train_data:», train_data.shape[0])
print(«Сума розмірів груп у тестовому наборі:»,
group_test_sizes.sum())
print(«Кількість рядків у test_data:», test_data.shape[0])

# Best parameters: {'objective': 'rank:ndcg', 'eval_metric':
'ndcg', 'max_depth': 5, 'min_child_weight': 1, 'gamma': 0.5,
'subsample': 0.8, 'colsample_bytree': 0.6, 'eta': 0.3}
# Best NDCG score: 0.7304121728467919

# 3.5. Навчання контентної моделі

params = {
    'objective': 'rank:ndcg',
    'eval_metric': 'ndcg',
    'max_depth': 5,
    'min_child_weight': 1,
    'gamma': 0.5,
    'subsample': 0.8,
```

```
    'colsample_bytree': 0.6,
    'eta': 0.3
}

# Список даних для валідації
eval_list = [(dtrain, 'train'), (dtest, 'eval')]

# Навчання контентної моделі
model_content = xgb.train(
    params,
    dtrain,
    num_boost_round=1000,
    evals=eval_list,
    early_stopping_rounds=10,
#     verbose_eval=False
)
```

ДОДАТОК Г

Навчання моделі KNN

```
# Навчання та тестування колаборативної моделі
# 4.1. Підготовка матриці взаємодій

# Створення матриці взаємодій
interaction_df = assignments_df.pivot_table(
    index='vacancy_id',
    columns='sailor_id',
    values='assignment_success',
    fill_value=0
)

# 4.2. Навчання моделі KNN

# Перетворення матриці у формат, придатний для KNN
interaction_matrix = interaction_df.values

# Ініціалізація моделі KNN
model_cf = NearestNeighbors(metric='cosine', algorithm='brute')
model_cf.fit(interaction_matrix)

interaction_df
interaction_df.values

# 4.3. Функція для отримання рекомендацій від колаборативної
моделі

# Створення словників відповідності індексів та ідентифікаторів
vacancy_id_mapping = {id: idx for idx, id in
enumerate(interaction_df.index)}
vacancy_id_reverse_mapping = {idx: id for id, idx in
vacancy_id_mapping.items()}
sailor_id_list = interaction_df.columns.tolist()

def recommend_sailors_for_vacancy_cf(vacancy_id, N=10):
    # Отримуємо внутрішній індекс вакансії
    vacancy_idx = vacancy_id_mapping.get(vacancy_id, None)
    if vacancy_idx is None:
        return pd.DataFrame()

    # Отримуємо вектор взаємодій для вакансії
    vacancy_vector = interaction_matrix[vacancy_idx].reshape(1,
-1)

    # Знаходимо найбільш схожі вакансії
    n_neighbors = min(N+1, model_cf.n_samples_fit_)
    distances, indices = model_cf.kneighbors(vacancy_vector,
n_neighbors=n_neighbors)
```

```
    # Отримуємо список схожих вакансій (пропускаємо першу,
оскільки це сама вакансія)
    similar_vacancy_indices = indices.flatten()[1:]

    # Отримуємо моряків, які успішно працювали на схожих
вакансіях
    similar_vacancies =
interaction_df.index[similar_vacancy_indices]
    sailors_scores =
interaction_df.loc[similar_vacancies].sum().sort_values(ascendin
g=False)

    # Отримуємо топ N моряків
    top_sailors = sailors_scores.head(N).reset_index()
    top_sailors.columns = ['sailor_id', 'cf_score']

    return top_sailors.head(N)
```

ДОДАТОК Д

Збереження та відновлення моделей та трансформерів

```
# Збереження контентної моделі
model_content.save_model('./Models/xgboost_ranker__improved.mode
l')

# Збереження колаборативної моделі
with open('./Models/model_cf__improved.pkl', 'wb') as f:
    pickle.dump(model_cf, f)

# Збереження енкодерів та масштабувальників
joblib.dump(le_position,
'./Transformers/le_position__improved.pkl')
joblib.dump(le_ship_type,
'./Transformers/le_ship_type__improved.pkl')
joblib.dump(mlb_certifications,
'./Transformers/mlb_certifications__improved.pkl')
joblib.dump(scaler, './Transformers/scaler__improved.pkl')

# Використання системи для отримання рекомендацій

# Завантаження контентної моделі
model_content = xgb.Booster()
model_content.load_model('./Models/xgboost_ranker__improved.mode
l')

# Завантаження колаборативної моделі
with open('./Models/model_cf__improved.pkl', 'rb') as f:
    model_cf = pickle.load(f)

# Завантаження енкодерів
le_position =
joblib.load('./Transformers/le_position__improved.pkl')
le_ship_type =
joblib.load('./Transformers/le_ship_type__improved.pkl')
mlb_certifications =
joblib.load('./Transformers/mlb_certifications__improved.pkl')
scaler = joblib.load('./Transformers/scaler__improved.pkl')
```

ДОДАТОК Е

Отримання рекомендацій за домогою комбінованої моделі

```
def calculate_dynamic_k(vacancy_id, interaction_df):
    """
Розрахунок динамічного параметра k на основі характеристик даних
з покращеним масштабуванням
    """
    # Get vacancy's ship type and position
    vacancy = vacancies_df[vacancies_df['vacancy_id'] ==
vacancy_id].iloc[0]
    vacancy_type = vacancy['ship_type']
    vacancy_position = vacancy['position']

    # Find similar vacancies
    similar_vacancies = vacancies_df[
        (vacancies_df['ship_type'] == vacancy_type) &
        (vacancies_df['position'] == vacancy_position)
    ].vacancy_id

    # Calculate metrics
    similar_interactions =
interaction_df.loc[similar_vacancies].sum(axis=1)

    if len(similar_interactions) > 0:
        avg_similar = similar_interactions.mean()
        std_similar = similar_interactions.std()

        # Calculate coefficient of variation for similar
vacancies
        cv = std_similar / avg_similar if avg_similar > 0 else
1.0

        # Scale k based on coefficient of variation and number
of similar vacancies
        base_k = 5.0  # base value for k
        similarity_factor = np.exp(-len(similar_vacancies) / 10)
# decreases with more similar vacancies
        variability_factor = np.clip(cv, 0.1, 2.0)  # increases
with more variability

        k = base_k * (1 + similarity_factor) *
variability_factor
    else:
        # If no similar vacancies, use higher k to be more
conservative
        k = 10.0

    # Clip to reasonable bounds
    return np.clip(k, 3.0, 15.0)
```

```
def combined_recommendations_dynamic_alpha(vacancy, N=10,
is_new=False):
    """Get recommendations with dynamic alpha calculation"""
    if is_new:
        # For new vacancies, use only content-based approach
        temp_df = create_vacancy_sailor_pairs(vacancy,
sailors_df)
        temp_df = prepare_features(temp_df)
        X_candidates = temp_df[feature_columns]
        X_candidates = X_candidates.fillna(0)
        dmatrix_candidates = xgb.DMatrix(X_candidates)
        content_scores =
model_content.predict(dmatrix_candidates)
        temp_df['combined_score'] = (content_scores -
content_scores.min()) / (content_scores.max() -
content_scores.min() + 1e-8)
        return temp_df[['sailor_id',
'combined_score']].sort_values(by='combined_score',
ascending=False).head(N)

    vacancy_id = vacancy if isinstance(vacancy, (int,
np.integer)) else vacancy['vacancy_id']
    cf_recs = recommend_sailors_for_vacancy_cf(vacancy_id,
N=100)

    # Calculate dynamic k and alpha
    k = calculate_dynamic_k(vacancy_id, interaction_df)
    num_interactions = interaction_df.loc[vacancy_id].sum()
    alpha = num_interactions / (num_interactions + k)

    if cf_recs.empty or alpha < 0.1:  # Use threshold to ensure
minimum confidence
        candidates_df = merged_df[merged_df['vacancy_id'] ==
vacancy_id]
        X_candidates = candidates_df[feature_columns]
        X_candidates = X_candidates.fillna(0)
        dmatrix_candidates = xgb.DMatrix(X_candidates)
        content_scores =
model_content.predict(dmatrix_candidates)
        candidates_df['combined_score'] = content_scores
        return candidates_df[['sailor_id',
'combined_score']].sort_values(by='combined_score',
ascending=False).head(N)

    # Rest of the function remains the same...
    sailors_ids = cf_recs['sailor_id'].values
    candidates_df = merged_df[
        (merged_df['sailor_id'].isin(sailors_ids)) &
        (merged_df['vacancy_id'] == vacancy_id)
    ]

    X_candidates = candidates_df[feature_columns]
    X_candidates = X_candidates.fillna(0)
```

```
    dmatrix_candidates = xgb.DMatrix(X_candidates)

    content_scores = model_content.predict(dmatrix_candidates)
    candidates_df = candidates_df.reset_index(drop=True)
    candidates_df['content_score'] = content_scores

    cf_scores = cf_recs.set_index('sailor_id')['cf_score']
    candidates_df['cf_score'] =
candidates_df['sailor_id'].map(cf_scores)

    candidates_df['content_score_norm'] = (
        (candidates_df['content_score'] -
candidates_df['content_score'].min()) /
        (candidates_df['content_score'].max() -
candidates_df['content_score'].min() + 1e-8)
    )
    candidates_df['cf_score_norm'] = (
        (candidates_df['cf_score'] -
candidates_df['cf_score'].min()) /
        (candidates_df['cf_score'].max() -
candidates_df['cf_score'].min() + 1e-8)
    )

    candidates_df['combined_score'] = (
        alpha * candidates_df['cf_score_norm'] +
        (1 - alpha) * candidates_df['content_score_norm']
    )

    return candidates_df[['sailor_id',
'combined_score']].sort_values(by='combined_score',
ascending=False).head(N)
```

ДОДАТОК Ж

Бекенд рекомендаційної технології (FastAPI on Python)

```
# Import required libraries
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import List, Optional
import xgboost as xgb
import numpy as np
import pandas as pd
from sklearn.neighbors import NearestNeighbors
import joblib
import pickle
from datetime import datetime

# Initialize FastAPI app
app = FastAPI(title=«Crew Recommendation API»)

# Define data models
class Candidate(BaseModel):
    sailor_id: int
    score: float
    rank: int

class RecommendationResponse(BaseModel):
    vacancy_id: int
    recommendations: List[Candidate]
    timestamp: str
    processing_time: float

class RecommendationSystem:
    def __init__(self):
        # Load models and encoders
        self.content_model = xgb.Booster()

self.content_model.load_model('./Models/xgboost_ranker__improved
.model')

        with open('./Models/model_cf__improved.pkl', 'rb') as f:
            self.collab_model = pickle.load(f)

        # Load encoders and scalers
        self.le_position =
joblib.load('./Transformers/le_position__improved.pkl')
        self.le_ship_type =
joblib.load('./Transformers/le_ship_type__improved.pkl')
        self.mlb_certifications =
joblib.load('./Transformers/mlb_certifications__improved.pkl')
        self.scaler =
joblib.load('./Transformers/scaler__improved.pkl')

        # Load interaction matrix for collaborative filtering
```

```
        self.interaction_df =
pd.read_pickle('./Data/interaction_matrix.pkl')

    def _calculate_dynamic_alpha(self, vacancy_id: int, k: float
= 5.0) -> float:
        «««Calculate dynamic weight based on interaction
history.»««
        num_interactions =
self.interaction_df.loc[vacancy_id].sum()
        return num_interactions / (num_interactions + k)

    def _prepare_features(self, vacancy_data: dict,
candidate_data: dict) -> np.ndarray:
        «««Prepare and encode features for content-based
model.»««
        # Implementation of feature preparation logic
        # This would include all the feature engineering steps
from your original code
        # Returns processed features ready for the model
        pass

    def get_recommendations(self, vacancy_id: int,
n_recommendations: int = 10) -> List[Candidate]:
        «««Get hybrid recommendations for a vacancy.»««
        start_time = datetime.now()

        try:
            # Get collaborative recommendations
            cf_recs =
self._get_collaborative_recommendations(vacancy_id, n=100)

            # Calculate dynamic alpha
            alpha = self._calculate_dynamic_alpha(vacancy_id)

            if cf_recs.empty:
                # Fall back to content-based only
                recommendations =
self._get_content_recommendations(vacancy_id, n_recommendations)
            else:
                # Get hybrid recommendations
                recommendations =
self._get_hybrid_recommendations(
                    vacancy_id,
                    cf_recs,
                    alpha,
                    n_recommendations
                )

            # Format response
            response = RecommendationResponse(
                vacancy_id=vacancy_id,
                recommendations=[
                    Candidate(
```

```
                        sailor_id=int(row['sailor_id']),
                        score=float(row['combined_score']),
                        rank=idx + 1
                    )
                    for idx, row in recommendations.iterrows()
                ],
                timestamp=datetime.now().isoformat(),
                processing_time=(datetime.now() -
start_time).total_seconds()
            )

            return response

        except Exception as e:
            raise HTTPException(
                status_code=500,
                detail=f»Error generating recommendations:
{str(e)}»
            )

# Initialize recommendation system
recommender = RecommendationSystem()

# Define API endpoints
@app.get(«/health»)
async def health_check():
    «««Health check endpoint.»««
    return {«status»: «healthy», «timestamp»:
datetime.now().isoformat()}

@app.get(«/recommendations/{vacancy_id}»)
async def get_recommendations(
    vacancy_id: int,
    n_recommendations: Optional[int] = 10
) -> RecommendationResponse:
    «««Get recommendations for a vacancy.»««
    return recommender.get_recommendations(vacancy_id,
n_recommendations)

# Error handlers
@app.exception_handler(HTTPException)
async def http_exception_handler(request, exc):
    return {
        «error»: exc.detail,
        «status_code»: exc.status_code,
        «timestamp»: datetime.now().isoformat()
    }

@app.exception_handler(Exception)
async def general_exception_handler(request, exc):
    return {
        «error»: «Internal server error»,
        «detail»: str(exc),
```

```
        «status_code»: 500,
        «timestamp»: datetime.now().isoformat()
    }

if __name__ == «__main__»:
    import uvicorn
    uvicorn.run(app, host=«0.0.0.0», port=8000)
```

ДОДАТОК К

Формування аналітики для контентої моделі (XGBoost)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import ndcg_score
import xgboost as xgb

def evaluate_content_model(model, test_data, feature_columns,
group_sizes, k_values=[5, 10, 20]):
    """
    Оцінює контентну модель використовуючи декілька метрик при
різних значеннях K
    """
    metrics = {k: {'ndcg': [], 'map': [], 'precision': []} for k
in k_values}

    start = 0
    for size in group_sizes:
        end = start + size

        # Отримуємо прогнози та справжні мітки для поточної
групи
        y_true = test_data['label'].values[start:end]
        X = test_data[feature_columns].values[start:end]

        # Створюємо DMatrix з явними іменами ознак
        dmatrix = xgb.DMatrix(X, feature_names=feature_columns)

        # Отримуємо прогнози
        y_pred = model.predict(dmatrix)

        # Обчислюємо метрики для кожного k
        for k in k_values:
            if len(y_true) >= k:
                # Сортуємо прогнози та беремо топ k
                indices = np.argsort(y_pred)[::-1][:k]
                y_true_k = y_true[indices]
                y_pred_k = y_pred[indices]

                # Обчислюємо метрики
                ndcg = ndcg_score([y_true_k], [y_pred_k])
                ap = average_precision_at_k(y_true_k, k)
                precision = precision_at_k(y_true_k, k)

                metrics[k]['ndcg'].append(ndcg)
                metrics[k]['map'].append(ap)
                metrics[k]['precision'].append(precision)
```

```
        start = end

    return metrics

def average_precision_at_k(y_true, k):
    """Обчислює середню точність при k"""
    if len(y_true) > k:
        y_true = y_true[:k]

    score = 0.0
    num_hits = 0.0

    for i, y in enumerate(y_true):
        if y == 1:
            num_hits += 1.0
            score += num_hits / (i + 1.0)

    return score / min(len(y_true), k) if num_hits > 0 else 0.0

def precision_at_k(y_true, k):
    """Обчислює точність при k"""
    if len(y_true) > k:
        y_true = y_true[:k]

    return np.mean(y_true)

def plot_metrics_comparison(metrics):
    """
    Створює візуалізацію порівняння різних метрик для різних
значень K
    """
    k_values = sorted(metrics.keys())
    metric_names = ['NDCG', 'MAP', 'Precision']
    colors = ['#2ecc71', '#3498db', '#e74c3c']

    # Створюємо фігуру з більшим простором зверху
    fig = plt.figure(figsize=(20, 7))
    fig.suptitle('Метрики ефективності контентної моделі при
різних K', fontsize=14, y=1.02)

    # Створюємо підграфіки
    axes = []
    for i in range(3):
        ax = fig.add_subplot(1, 3, i+1)
        axes.append(ax)

    for idx, (metric, color) in enumerate(zip(['ndcg', 'map',
'precision'], colors)):
        ax = axes[idx]

        data = [metrics[k][metric] for k in k_values]

        # Створюємо boxplot
```

```
        bp = ax.boxplot(data, labels=[f'K={k}' for k in
k_values],
                        patch_artist=True)

        # Налаштовуємо кольори
        for box in bp['boxes']:
            box.set(facecolor=color, alpha=0.6)
        plt.setp(bp['medians'], color='darkred', linewidth=1.5)

        # Додаємо точки розсіювання
        for i, d in enumerate(data):
            y = d
            x = np.random.normal(i+1, 0.04, size=len(y))
            ax.scatter(x, y, color=color, alpha=0.2, s=20)

        # Налаштовуємо вигляд
        ax.set_title(f'{metric_names[idx]}@K')
        ax.set_ylabel('Оцінка')
        ax.grid(True, alpha=0.2)
        ax.set_ylim(0, 1)

        # Додаємо статистику
        means = [np.mean(metrics[k][metric]) for k in k_values]
        stds = [np.std(metrics[k][metric]) for k in k_values]
        for i, (mean, std) in enumerate(zip(means, stds)):
            stats_text = f'μ={mean:.3f}\nσ={std:.3f}'
            ax.text(i+1, -0.15, stats_text, ha='center',
va='top', fontsize=8)

    plt.tight_layout()
    return fig

def plot_feature_importance(model, feature_columns, top_n=20):
    """
    Створює візуалізацію важливості ознак
    """
    # Отримуємо важливість ознак
    importance = model.get_score(importance_type='weight')

    # Створюємо DataFrame з усіма ознаками
    importance_df = pd.DataFrame({
        'Ознака': feature_columns,
        'Важливість': [importance.get(f, 0) for f in
feature_columns]
    })

    # Сортуємо та беремо топ ознаки
    importance_df = importance_df.sort_values('Важливість',
ascending=True).tail(top_n)

    # Створюємо фігуру
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_subplot(111)
```

```
    # Створюємо градієнтні кольори
    colors = plt.cm.viridis(np.linspace(0.2, 0.8,
len(importance_df)))

    # Створюємо стовпчики
    bars = ax.barh(importance_df['Ознака'],
importance_df['Важливість'],
                   color=colors)

    # Додаємо підписи значень
    for bar in bars:
        width = bar.get_width()
        ax.text(width, bar.get_y() + bar.get_height()/2,
                f'{width:.2f}',
                ha='left', va='center',
                bbox=dict(facecolor='white', edgecolor='none',
alpha=0.7))

    # Налаштовуємо вигляд
    ax.set_title('Важливість ознак у контентній моделі', y=1.02)
    ax.set_xlabel('Оцінка важливості')
    ax.grid(axis='x', alpha=0.2)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)

    plt.tight_layout()
    return fig

def analyze_content_model(model, test_data, group_sizes,
feature_columns):
    """
    Виконує повний аналіз контентної моделі та створює
візуалізації
    """
    print("Починаємо оцінку моделі...")
    metrics = evaluate_content_model(model, test_data,
feature_columns, group_sizes)

    print("Створюємо візуалізацію метрик...")
    metrics_fig = plot_metrics_comparison(metrics)

    print("Створюємо візуалізацію важливості ознак...")
    importance_fig = plot_feature_importance(model,
feature_columns)

    # Виводимо статистику
    print("\nЗведення метрик:")
    print("=" * 50)
    for k in sorted(metrics.keys()):
        print(f"\nПри K={k}:")
        for metric in ['ndcg', 'map', 'precision']:
            values = metrics[k][metric]
```

```
            metric_names = {'ndcg': 'NDCG', 'map': 'MAP',
'precision': 'Точність'}
            print(f"{metric_names[metric]}@{k}:")
            print(f"  Середнє: {np.mean(values):.4f}")
            print(f"  Станд. відхил.: {np.std(values):.4f}")
            print(f"  Мінімум: {np.min(values):.4f}")
            print(f"  Максимум: {np.max(values):.4f}")

    return metrics, metrics_fig, importance_fig

# Переконайтесь, що test_data містить всі необхідні колонки
X_test_columns = [col for col in feature_columns if col in
test_data.columns]
if len(X_test_columns) != len(feature_columns):
    print("Warning: Some features are missing in test data!")
    print("Missing features:", set(feature_columns) -
set(X_test_columns))

# Запуск аналізу
metrics, metrics_fig, importance_fig = analyze_content_model(
    model_content,
    test_data,
    group_test_sizes,
    X_test_columns
)

# Відображення графіків
plt.figure(1)
plt.show()
plt.figure(2)
plt.show()
```

ДОДАТОК Л

Формування аналітики для колаборативної моделі (KNN)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import precision_score, recall_score,
ndcg_score
from sklearn.metrics.pairwise import cosine_similarity
from collections import defaultdict

def evaluate_knn_model(model, interaction_df, assignments_df,
k_values=[5, 10, 20]):
    """
    Оцінює KNN модель використовуючи декілька метрик
    """
    metrics = defaultdict(list)

    # Отримуємо матрицю схожості
    interaction_matrix = interaction_df.values
    similarities = cosine_similarity(interaction_matrix)

    for vacancy_idx, vacancy_id in
enumerate(interaction_df.index):
        # Отримуємо фактичних успішних кандидатів
        actual_sailors = assignments_df[
            (assignments_df['vacancy_id'] == vacancy_id) &
            (assignments_df['assignment_success'] == 1)
        ]['sailor_id'].tolist()

        if not actual_sailors:
            continue

        # Отримуємо рекомендації для різних k
        # Знаходимо схожі вакансії
        similar_indices =
np.argsort(similarities[vacancy_idx])[::-1][1:max(k_values)+1]

        for k in k_values:
            # Беремо топ-k схожих вакансій
            top_k_similar = similar_indices[:k]

            # Отримуємо рекомендованих моряків зі схожих
вакансій
            recommended_sailors = []
            for idx in top_k_similar:
                vacancy_vector = interaction_matrix[idx]
                successful_sailors = np.where(vacancy_vector ==
1)[0]
```

```
recommended_sailors.extend(interaction_df.columns[successful_sai
lors])

            if not recommended_sailors:
                continue

            # Підраховуємо частоти та отримуємо топ рекомендації
            sailor_counts =
pd.Series(recommended_sailors).value_counts()
            recommended = sailor_counts.index.tolist()[:k]

            # Обчислюємо метрики
            if recommended and actual_sailors:
                precision = len(set(recommended) &
set(actual_sailors)) / len(recommended)
                recall = len(set(recommended) &
set(actual_sailors)) / len(actual_sailors)
                coverage = len(set(recommended)) /
len(interaction_df.columns)

                # Зберігаємо метрики
                metrics[f'precision_k{k}'].append(precision)
                metrics[f'recall_k{k}'].append(recall)
                metrics[f'coverage_k{k}'].append(coverage)

    return metrics

def analyze_sparsity(interaction_df):
    """
    Аналізує розрідженість матриці взаємодій
    """
    interaction_matrix = interaction_df.values
    total_cells = interaction_matrix.size
    filled_cells = np.count_nonzero(interaction_matrix)
    sparsity = 1 - (filled_cells / total_cells)

    vacancy_interactions = np.sum(interaction_matrix, axis=1)
    sailor_interactions = np.sum(interaction_matrix, axis=0)

    return {
        'sparsity': sparsity,
        'avg_vacancy_interactions':
np.mean(vacancy_interactions),
        'avg_sailor_interactions': np.mean(sailor_interactions),
        'vacancy_interactions': vacancy_interactions,
        'sailor_interactions': sailor_interactions
    }

def plot_knn_metrics(metrics, k_values=[5, 10, 20]):
    """
    Візуалізує метрики KNN моделі
    """
```

```
    fig, axes = plt.subplots(1, 3, figsize=(20, 6))
    fig.suptitle('Метрики ефективності KNN моделі', y=1.02,
fontsize=14)

    metric_names = ['Precision', 'Recall', 'Coverage']
    metric_keys = ['precision', 'recall', 'coverage']
    colors = ['#2ecc71', '#3498db', '#e74c3c']

    for idx, (metric_name, metric_key, color) in
enumerate(zip(metric_names, metric_keys, colors)):
        ax = axes[idx]

        # Підготовка даних для boxplot
        data = [metrics[f'{metric_key}_k{k}'] for k in k_values]
        if not all(data):  # Пропускаємо якщо немає даних
            ax.text(0.5, 0.5, 'Немає даних',
                    ha='center', va='center',
transform=ax.transAxes)
            continue

        # Створюємо boxplot
        bp = ax.boxplot(data, labels=[f'K={k}' for k in
k_values], patch_artist=True)

        # Налаштовуємо кольори
        for box in bp['boxes']:
            box.set(facecolor=color, alpha=0.6)
        plt.setp(bp['medians'], color='darkred', linewidth=1.5)

        # Додаємо точки розсіювання
        for i, d in enumerate(data):
            y = d
            x = np.random.normal(i+1, 0.04, size=len(y))
            ax.scatter(x, y, color=color, alpha=0.2, s=20)

        # Налаштовуємо вигляд
        ax.set_title(f'{metric_name}@K')
        ax.set_ylabel('Оцінка')
        ax.grid(True, alpha=0.2)
        ax.set_ylim(0, 1)

        # Додаємо статистику
        means = [np.mean(d) if d else 0 for d in data]
        stds = [np.std(d) if d else 0 for d in data]
        for i, (mean, std) in enumerate(zip(means, stds)):
            stats_text = f'μ={mean:.3f}\nσ={std:.3f}'
            ax.text(i+1, -0.15, stats_text, ha='center',
va='top', fontsize=8)

    plt.tight_layout()
    return fig

def plot_sparsity_analysis(sparsity_stats):
```

```
    """
    Візуалізує аналіз розрідженості
    """
    fig, axes = plt.subplots(1, 2, figsize=(15, 5))
    fig.suptitle('Аналіз розрідженості матриці взаємодій',
y=1.02, fontsize=14)

    # Розподіл взаємодій вакансій
    sns.histplot(sparsity_stats['vacancy_interactions'],
                 bins=30, ax=axes[0], color='#3498db')
    axes[0].set_title('Розподіл взаємодій вакансій')
    axes[0].set_xlabel('Кількість взаємодій')
    axes[0].set_ylabel('Кількість')

    # Розподіл взаємодій моряків
    sns.histplot(sparsity_stats['sailor_interactions'],
                 bins=30, ax=axes[1], color='#2ecc71')
    axes[1].set_title('Розподіл взаємодій моряків')
    axes[1].set_xlabel('Кількість взаємодій')
    axes[1].set_ylabel('Кількість')

    plt.tight_layout()
    return fig

def evaluate_and_visualize_knn(model, interaction_df,
assignments_df):
    """
    Повна оцінка KNN моделі з візуалізаціями
    """
    print("Оцінюємо KNN модель...")
    metrics = evaluate_knn_model(model, interaction_df,
assignments_df)

    print("Аналізуємо розрідженість...")
    sparsity_stats = analyze_sparsity(interaction_df)

    print("\nАналіз розрідженості:")
    print(f"Розрідженість матриці:
{sparsity_stats['sparsity']:.2%}")
    print(f"Середня кількість взаємодій на вакансію:
{sparsity_stats['avg_vacancy_interactions']:.2f}")
    print(f"Середня кількість взаємодій на моряка:
{sparsity_stats['avg_sailor_interactions']:.2f}")

    print("\nСтворюємо візуалізації...")
    metrics_fig = plot_knn_metrics(metrics)
    sparsity_fig = plot_sparsity_analysis(sparsity_stats)

    return metrics, sparsity_stats, metrics_fig, sparsity_fig

# Оцінка KNN моделі
metrics, sparsity_stats, metrics_fig, sparsity_fig =
evaluate_and_visualize_knn(
```

```
    model_cf,         # навчена KNN модель
    interaction_df,   # DataFrame з матрицею взаємодій
    assignments_df    # DataFrame| з даними про призначення
)

# Відображення графіків
plt.figure(1)
metrics_fig.show()
plt.figure(2)
sparsity_fig.show()
```

ДОДАТОК М

Формування аналітики для комбінованої моделі

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import ndcg_score, precision_score,
recall_score
from collections import defaultdict

def evaluate_hybrid_model(interaction_df, assignments_df,
content_model, feature_columns, k_values=[5, 10, 20],
k_alpha=5):
    """
    Оцінює гібридну модель з динамічним коефіцієнтом альфа
    """
    metrics = defaultdict(list)
    alpha_values = []

    for vacancy_id in interaction_df.index:
        # Обчислюємо динамічний коефіцієнт альфа
        num_interactions = interaction_df.loc[vacancy_id].sum()
        alpha = num_interactions / (num_interactions + k_alpha)
        alpha_values.append(alpha)

        # Отримуємо фактичних успішних кандидатів
        actual_sailors = assignments_df[
            (assignments_df['vacancy_id'] == vacancy_id) &
            (assignments_df['assignment_success'] == 1)
        ]['sailor_id'].tolist()

        if not actual_sailors:
            continue

        # Отримуємо рекомендації для різних k
        for k in k_values:
            recommendations =
combined_recommendations_dynamic_alpha(
                vacancy_id, N=k, k=k_alpha
            )

            if recommendations.empty:
                continue

            recommended_sailors =
recommendations['sailor_id'].tolist()
            scores = recommendations['combined_score'].values

            # Обчислюємо метрики
```

```
            precision = len(set(recommended_sailors) &
set(actual_sailors)) / len(recommended_sailors)
            recall = len(set(recommended_sailors) &
set(actual_sailors)) / len(actual_sailors)

            # Обчислення NDCG
            y_true = [1 if sailor in actual_sailors else 0 for
sailor in recommended_sailors]
            ndcg = ndcg_score([y_true], [scores])

            # Покриття відносно обох пулів (контентного та
колаборативного)
            coverage = len(set(recommended_sailors)) /
len(interaction_df.columns)

            metrics[f'precision_k{k}'].append(precision)
            metrics[f'recall_k{k}'].append(recall)
            metrics[f'ndcg_k{k}'].append(ndcg)
            metrics[f'coverage_k{k}'].append(coverage)

    return metrics, alpha_values

def plot_hybrid_metrics(metrics, k_values=[5, 10, 20]):
    """
    Візуалізує метрики гібридної моделі
    """
    fig, axes = plt.subplots(2, 2, figsize=(20, 12))
    fig.suptitle('Метрики ефективності гібридної моделі',
y=1.02, fontsize=14)

    metric_names = ['Precision', 'Recall', 'NDCG', 'Coverage']
    metric_keys = ['precision', 'recall', 'ndcg', 'coverage']
    colors = ['#2ecc71', '#3498db', '#e74c3c', '#9b59b6']

    for idx, (metric_name, metric_key, color) in
enumerate(zip(metric_names, metric_keys, colors)):
        ax = axes[idx//2, idx%2]

        data = [metrics[f'{metric_key}_k{k}'] for k in k_values]

        # Створюємо boxplot
        bp = ax.boxplot(data, labels=[f'K={k}' for k in
k_values], patch_artist=True)

        # Налаштовуємо кольори
        for box in bp['boxes']:
            box.set(facecolor=color, alpha=0.6)
        plt.setp(bp['medians'], color='darkred', linewidth=1.5)

        # Додаємо точки розсіювання
        for i, d in enumerate(data):
            y = d
            x = np.random.normal(i+1, 0.04, size=len(y))
```

```
            ax.scatter(x, y, color=color, alpha=0.2, s=20)

        # Налаштовуємо вигляд
        ax.set_title(f'{metric_name}@K')
        ax.set_ylabel('Оцінка')
        ax.grid(True, alpha=0.2)
        ax.set_ylim(0, 1)

        # Додаємо статистику
        means = [np.mean(d) for d in data]
        stds = [np.std(d) for d in data]
        for i, (mean, std) in enumerate(zip(means, stds)):
            stats_text = f'μ={mean:.3f}\nσ={std:.3f}'
            ax.text(i+1, -0.15, stats_text, ha='center',
va='top', fontsize=8)

    plt.tight_layout()
    return fig

def plot_alpha_distribution(alpha_values):
    """
    Візуалізує розподіл значень динамічного коефіцієнта альфа
    """
    plt.figure(figsize=(10, 6))
    sns.histplot(alpha_values, bins=30)
    plt.title('Розподіл значень динамічного коефіцієнта альфа')
    plt.xlabel('Значення альфа')
    plt.ylabel('Кількість')
    plt.grid(True, alpha=0.2)
    return plt.gcf()

def analyze_hybrid_model(interaction_df, assignments_df,
content_model, feature_columns, k_values=[5, 10, 20],
k_alpha=5):
    """
    Повний аналіз гібридної моделі
    """
    print("Оцінюємо гібридну модель...")
    metrics, alpha_values = evaluate_hybrid_model(
        interaction_df, assignments_df, content_model,
feature_columns, k_values, k_alpha
    )

    print("\nСтворюємо візуалізації...")
    metrics_fig = plot_hybrid_metrics(metrics, k_values)
    alpha_fig = plot_alpha_distribution(alpha_values)

    # Виводимо статистику
    print("\nЗведення метрик:")
    print("=" * 50)
    for k in k_values:
        print(f"\nПри K={k}:")
```

```
        for metric in ['precision', 'recall', 'ndcg',
'coverage']:
            values = metrics[f'{metric}_k{k}']
            metric_names = {
                'precision': 'Precision',
                'recall': 'Recall',
                'ndcg': 'NDCG',
                'coverage': 'Coverage'
            }
            print(f"{metric_names[metric]}@{k}:")
            print(f"  Середнє: {np.mean(values):.4f}")
            print(f"  Станд. відхил.: {np.std(values):.4f}")
            print(f"  Мінімум: {np.min(values):.4f}")
            print(f"  Максимум: {np.max(values):.4f}")

    print("\nСтатистика коефіцієнта альфа:")
    print(f"Середнє значення альфа:
{np.mean(alpha_values):.4f}")
    print(f"Станд. відхил. альфа: {np.std(alpha_values):.4f}")

    return metrics, alpha_values, metrics_fig, alpha_fig

# Оцінка комбінованої моделі
metrics, alpha_values, metrics_fig, alpha_fig =
analyze_hybrid_model(
    interaction_df,
    assignments_df,
    model_content,
    feature_columns,
    k_values=[5, 10, 20],
    k_alpha=5
)

# Відображення графіків
plt.figure(1)
metrics_fig.show()
plt.figure(2)
alpha_fig.show()
```

ДОДАТОК Н

Запит на виконання кваліфікаційної роботи

Завідувачу кафедри
МЗКС, ФМФІТ
ОНУ імені І.І. Мечникова
д.т.н., професору
Малахову Є.В.

Шановний Євгеній Валерійович,

ПрАТ «СТАФФ ЦЕНТР» звертається з проханням призначити здобувачу 2-го рівня ВО за спеціальністю 126 «Інформаційні системи та технології» ОНУ імені І.І. Мечникова Джигову Дмитру Юрійовичу тему кваліфікаційної роботи «Інформаційна технологія підбору та рекомендації кандидатів на вакансію на основі компетентнісної оцінки».

Означена тема відповідає актуальним потребам нашої компанії в контексті автоматизації процесів підбору персоналу, а її результати будуть впроваджені в інформаційну систему «Crewisor» та сприятимуть подальшій діяльності випускника у компанії.

Директор
ПрАТ "СТАФФ ЦЕНТР" Крилов Ю.В.
"18" березня 2023 р.

SCG Global
Cyprus / Ukraine
P +35799874942
P +380503163396
svs@scgglobal.group
https://scglobal.group/

SCG

ДОДАТОК О

Довідка про впровадження

ДОВІДКА
про впровадження

Інформаційна технологія підбору та рекомендації кандидатів на вакансії на основі компетентнісної оцінки, розроблена студентом Одеського національного університету імені І.І. Мечникова Джиговим Дмитром Юрійовичем під час виконання кваліфікаційної роботи магістра на кафедрі МЗКС, успішно пройшла виробничі випробування в ПрАТ «СТАФФ ЦЕНТР» та запланована до впровадження як складова інформаційної системи «Crewisor».

Директор
ПрАТ "СТАФФ ЦЕНТР" Крилов Ю.В.
"1" січня 2024 р.

SCG Global
Cyprus / Ukraine

P +35799874942
P +380503163396

svs@scgglobal.group
https://scglobal.group/

SCG