НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

НН Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Прогнозування психічного здоров'я на основі даних про музичні вподобання»

Виконала: студентка IV курсу, групи КА-93
Ланько Анна Анатоліївна _____

Керівник: д.т.н., доцент
Недашківська Надія Іванівна _____

Консультант з нормоконтролю: к.ф-м.н.
Статкевич Віталій Михайлович _____

Консультант з економічного розділу: к.е.н., доцент
Рощина Надія Василівна _____

Рецензент: професор кафедри штучного інтелекту
КПІ ім. Ігоря Сікорського, д.т.н.,
Данилов Валерій Якович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студентка _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«__» травня 2023 р.

ЗАВДАННЯ
на дипломну роботу студенту
Ланько Анні Анатоліївні

1. Тема роботи «Прогнозування психічного здоров'я на основі даних про музичні вподобання», керівник роботи Недашківська Надія Іванівна, д.т.н., доцент, затверджені наказом по університету від «__» травня 2023 р. № _____

2. Термін подання студентом роботи: __.06.2023.

3. Вихідні дані до роботи: опитувальник за 33 питаннями щодо музичних вподобань респондентів у формі табличних даних.

4. Зміст роботи: огляд вихідних даних та методів машинного навчання для розв'язання задачі регресії; математичний опис методів машинного навчання для розв'язання задачі регресії та прикладних алгоритмів; результати експериментальних досліджень моделей прогнозування оцінок ментального здоров'я; функціонально-вартісний аналіз програмного продукту для прогнозування оцінок ментального здоров'я.

5. Перелік ілюстративного матеріалу: презентація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., к.е.н., доцент		

7. Дата видачі завдання: 17.04.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Дослідження літератури за обраною темою	21.04.2023	виконано
2.	Вибір вихідних даних	22.04.2023	виконано
3.	Постановка задачі	23.04.2023	виконано
4.	Підготовка першого розділу	30.04.2023	виконано
5.	Підготовка другого розділу	10.05.2023	виконано
6.	Підготовка третього розділу	21.05.2023	виконано
7.	Підготовка програмного продукту	21.05.2023	виконано
8.	Підготовка четвертого розділу	24.05.2023	виконано
9.	Допрацювання текстової частини та програмного продукту	26.05.2023	виконано
10.	Підготовка пояснювальної записки та презентації	28.05.2023	виконано

Студентка _____ Анна ЛАНЬКО

Керівник _____ Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Дипломна робота: 122 с., 20 рис., 20 табл., 2 додатки, 34 джерела.

ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ, РЕГРЕСІЯ, АНСАМБЛЕВІ МЕТОДИ, МЕТОД ОПОРНИХ ВЕКТОРІВ, ПСИХІЧНЕ ЗДОРОВ’Я.

Окремі категорії соціологічних даних мають дуже складні залежності між чинниками та досліджуваним явищем, оскільки вони є продуктом суб'єктивної думки неекспертних осіб з великою кількістю більш-менш рівнозначних факторів впливу.

Метою даного дослідження є якомога точніше застосування методології прогнозування до даних такого рівня складності.

Об’єктом дослідження є прогнозування на основі метаданих про музичні вподобання оцінок від 0 до 10, що відповідають за ступінь наявності чотирьох ментальних розладів: тривожності, депресії, безсоння та ОКР.

Предметом дослідження є алгоритми машинного навчання, що розв’язують задачу регресії: ансамблеві методи та метод опорних векторів.

Результатом дослідження є алгоритм прогнозування оцінок, розроблений мовою програмування Python на основі найкращої за метриками якості моделі регресії, який працює як на досліджуваних вихідних даних, так і власних вибірках.

Ключовою особливістю розробленого алгоритму є швидкодія та універсальність, що дає реалізувати його як додатковий функціонал більш глобальних, повноцінних програмних продуктів з метою покращення їхніх рекомендаційних систем та створення сприятливого середовища для спільноти користувачів.

ABSTRACT

Bachelor thesis: 122 p., 20 fig., 20 tabl., 2 append., 34 sources.

PREDICTION, MACHINE LEARNING, REGRESSION, ENSEMBLE METHODS, SUPPORT VECTOR MACHINE, MENTAL HEALTH.

Certain categories of sociological data have complicated relationships between causes and the studied phenomenon, as they are a product of non-expert subjective opinion with a large number of factors with nearly equal significance levels.

The purpose of this research is to apply the prediction methodology as accurately as possible to data of such a complexity level.

The object of this research is the prediction of scores from 0 to 10, which measure severity for 4 mental disorders: anxiety, depression, insomnia, and OCD, based on music preferences metadata.

The subject of this research are the machine learning algorithms that solve a regression problem: ensemble methods and the method of support vectors.

The result of this research is the score prediction algorithm developed in Python programming language and based on the best-performing regression model, which works both on the studied and manually collected data.

The key features of the developed algorithm are the speed of response and flexibility that allows to implement it as an additional feature of more global, full-fledged software products to improve their recommendation systems and create healthier environment for the user community.

ЗМІСТ

ВСТУП

Моделі прогнозування отримали широке розповсюдження у точних галузях, де наявність суттєвих причинно-наслідкових зв'язків між подіями дозволяє доволі якісно спрогнозувати поведінку досліджуваного явища на основі відомих факторів. Це дозволяє використовувати отриманий результат для визначення стратегії розвитку галузі або окремого продукту, оптимізації та мінімізації ризиків.

Соціологічне прогнозування є більш складною та цікавою задачею, тому що в такому випадку фактичні значення цільових змінних часто є не обґрунтованими фактами, а репрезентацією суб'єктивної думки певних, не обов'язково експертних осіб. Щодо вихідних даних також існує невизначеність, оскільки важко з'ясувати, які саме фактори та якою мірою вплинули на результат.

Завдання даної роботи полягає у створенні алгоритму практично прийнятної точності для оцінки стану психічного здоров'я на основі даних про музичні вподобання. Цьому сприяє доступність та репрезентативність вихідних даних, що дає змогу робити незалежні прогнози для різних соціальних груп і таким чином запобігти розвитку серйозних психічних патологій у окремих осіб, поліпшити загальний стан ментального здоров'я у суспільстві.

Пояснювальна записка складається з чотирьох розділів. У першому розділі поставлено та формалізовано задачу, проведено огляд і аналіз даних та доступних методів машинного навчання для розв'язання задач регресії. У другому розділі подається математична сутність використаних методів машинного навчання та алгоритмів обробки даних. У третьому розділі обрано найкращу з побудованих прогнозувальних моделей та перевірено її на власних даних, запропоновано методи удосконалення результативності. У четвертому розділі проводиться функціонально-вартісний аналіз розробленого продукту.

1 ОГЛЯД ВИХІДНИХ ДАНИХ ТА МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗВ’ЯЗАННЯ ЗАДАЧІ РЕГРЕСІЇ

У даному розділі наведено обґрунтування актуальності теми дослідження та постановку задачі, розглянуто наявні методи машинного навчання для її розв’язання. Окремо здійснено опис вихідних даних та проведено їхній аналіз за допомогою статистичних методів для кращого розуміння їхньої структури та визначення стратегії підготовки до навчання під час реалізації практичної частини у розділі 3.

1.1 Актуальність теми дослідження

Розглянуті у даній роботі порушення ментального спектру: тривожність, безсоння, депресія та обсесивно-компульсивний розлад – часто з'являються у психічно здорових людей на фоні постійних стресів, зумовлених жвавим темпом життя, кар’єрними викликами та надзвичайними ситуаціями, з якими потерпілий має співіснувати у часі (наприклад, пандемія, війна). За таких умов люди зазвичай усвідомлюють причини виникнення свого стану – і тому не вважають за потрібне вживати серйозних заходів щодо боротьби з наслідками. До інших причин ігнорування проблеми можуть відноситись особливості культурного середовища (культури певних регіонів світу засуджують емоційність, демонстрацію вразливості), небажання залучати сторонніх осіб до особистого життя тощо.

Хронічна наявність розглянутих порушень не тільки погіршує якість життя, а й послаблює нервову систему: збільшує її лабільність та підвищує чутливість до зовнішніх подразників. Саме тому досліджувані відхилення

часто передують багатьом складнішим психічним станам, які можна було б попередити вчасним зверненням до спеціалістів у галузі психічного здоров’я.

Актуальність даного дослідження зумовлена тим, що воно є інструментом попередження – найбільш дієвої стратегії боротьби зі збільшенням кількості ментальних та психічних розладів у населення. Тривіальність та доступність вихідних даних – інформація про музичні уподобання та звички – дозволяє робити прогнози для будь-яких соціальних груп, а головне – для осіб, що не звертаються по психологічну допомогу, проте потенційно можуть мати ментальні розлади. Цікавим втіленням алгоритму є інтеграція зі стрімінговими платформами (сервісами для легального прослуховування музики), що дозволятиме надсилати попередження та надавати екстрену інформаційну довідку користувачам у разі високо ймовірної наявності ментального порушення.

1.2 Опис даних

Дані було зібрано користувачем платформи Kaggle (Інтернет-спільноти для обміну досвідом та матеріалами в галузі машинного навчання) за допомогою форм, розміщених як у мережі (посилання на опитувальник у соціальних мережах, на форумах та Discord-серверах), так і в реальному житті (опитування людей на вулицях, у громадських місцях). Отриманий датасет був опублікований на вищезгаданій платформі у відкритому доступі у науково-дослідницьких цілях [1].

Кожен з 736 рядків файлу mxmh_survey_results.csv, вміст якого наведено на рисунку 1.1, є заповненим опитувальником унікальної особи про її музичні уподобання та звички.

	Timestamp	Age	Primary streaming service	Hours per day	While working	Instrumentalist	Composer	Fav genre	Exploratory	Foreign languages	...	Frequency [R&B]	Frequency [Rap]	Frequency [Rock]	Frequency [Video game music]	Anxiety	Depression	Insomnia	OCD	Music effects	Permissions
0	8/27/2022 19:29:02	18.0	Spotify	3.0	Yes	Yes	Yes	Latin	Yes	Yes	...	Sometimes	Very frequently	Never	Sometimes	3.0	0.0	1.0	0.0	NaN	I understand.
1	8/27/2022 19:57:31	63.0	Pandora	1.5	Yes	No	No	Rock	Yes	No	...	Sometimes	Rarely	Very frequently	Rarely	7.0	2.0	2.0	1.0	NaN	I understand.
2	8/27/2022 21:28:18	18.0	Spotify	4.0	No	No	No	Video game music	No	Yes	...	Never	Rarely	Rarely	Very frequently	7.0	7.0	10.0	2.0	No effect	I understand.
3	8/27/2022 21:40:40	61.0	YouTube Music	2.5	Yes	No	Yes	Jazz	Yes	Yes	...	Sometimes	Never	Never	Never	9.0	7.0	3.0	3.0	Improve	I understand.
4	8/27/2022 21:54:47	18.0	Spotify	4.0	Yes	No	No	R&B	Yes	No	...	Very frequently	Very frequently	Never	Rarely	7.0	2.0	5.0	9.0	Improve	I understand.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
731	10/30/2022 14:37:28	17.0	Spotify	2.0	Yes	Yes	No	Rock	Yes	Yes	...	Never	Rarely	Very frequently	Never	7.0	6.0	0.0	9.0	Improve	I understand.
732	11/1/2022 22:26:42	18.0	Spotify	1.0	Yes	Yes	No	Pop	Yes	Yes	...	Never	Never	Sometimes	Sometimes	3.0	2.0	2.0	5.0	Improve	I understand.
733	11/3/2022 23:24:38	19.0	Other streaming service	6.0	Yes	No	Yes	Rap	Yes	No	...	Sometimes	Sometimes	Rarely	Rarely	2.0	2.0	2.0	2.0	Improve	I understand.
734	11/4/2022 17:31:47	19.0	Spotify	5.0	Yes	Yes	No	Classical	No	No	...	Never	Never	Never	Sometimes	2.0	3.0	2.0	1.0	Improve	I understand.
735	11/9/2022 1:55:20	29.0	YouTube Music	2.0	Yes	No	No	Hip hop	Yes	Yes	...	Very frequently	Very frequently	Very frequently	Rarely	2.0	2.0	2.0	5.0	Improve	I understand.

736 rows × 33 columns

Рисунок 1.1 – Початковий вигляд табличних даних

Дані містять 33 атрибути, які можна умовно розділити на наступні категорії:

а) загальна інформація:

1) «Timestamp»: дата та час заповнення форми;
2) «Age»: вік респондента;
3) «Primary streaming service»: стрімінгова платформа, на якій респондент переважно прослуховує музику;
4) «Hours per day»: кількість годин прослуховування на день;
5) «BPM»: ритм (кількість ударів за хвилину) улюбленого жанру;
6) «Music effects»: вплив прослуховування музики на стан (погіршує / не змінює / покращує);

б) питання (відповіді: так / ні):

1) «While working»: чи слухає музику під час роботи
2) «Instrumentalist»: чи володіє музичними інструментами;
3) «Composer»: чи створює музику;
4) «Fav genre»: улюблений жанр;
5) «Exploratory»: чи цікавиться новинками, шукає інші жанри/виконавців;
6) «Foreign languages»: чи слухає музику іноземною мовою (якою добре не володіє);

в) оцінка частоти прослуховування для всіх 16 жанрів (значення: ніколи / дуже рідко / інколи / часто):
    1) «Frequency [назва жанру]»: для класичної музики, кантрі, електронної музики, фолку, госпелу, хіп-хопу, джазу, кей-попу, латини, лофі, металу, поп-музики, R&B, репу, року та саундтреків до відеоігор;
г) оцінка стану (значення: оцінка від 0 до 10, де 0 – абсолютно не відповідає поточному стану, 10 – повністю відповідає поточному стану):
    1) «Anxiety»: ступінь тривожності;
    2) «Depression»: ступінь депресивності;
    3) «Insomnia»: рівень безсоння;
    4) «OCD»: ступінь наявності обсесивно-компульсивного розладу;
д) Повідомлення про згоду респондента на відкриту публікацію його відповіді (значення: я згоден / я не згоден):
    1) «Permissions».

Ознаки «Timestamp» та «Permissions» є формальним додатком до основного, змістовного набору, тому ними можна знехтувати.

1.3 Аналіз даних

Отриманий в результаті ігнорування неінформативних (див. 1.2) ознак датасет містить 31 атрибут двох типів:

1) 7 числових (вік, кількість годин прослуховування на день, ритмічність та оцінки до кожного з 4 ментальних порушень) – кількісні ознаки, що приймають певні числові значення;
2) 24 категоріальні (решта) – якісні ознаки, що приймають певні значення з обмеженого набору категорій або рівнів.

Цільовими змінними, тобто об’єктом прогнозування [2], є 4 атрибути з оцінками ментального стану: «Anxiety», «Depression», «Insomnia», «OCD». Решта атрибутів є ознаками датасету – основою для визначення залежностей.

Атрибути з категорії «Оцінка частоти прослуховування» (див. 1.2) містять квадратні дужки у назві, що є небажаним для деяких алгоритмів машинного навчання (наприклад, XGBoost), тому доречно відформувати назви атрибутів.

Аномалії даних (нетипові значення) досліджуються за допомогою коробкових графіків, що зображені на рисунку 1.2 [3]. Для ознаки «BPM» виявлено суттєву аномалію (на рисунку позначена колом). Аномалії для решти ознак є прийнятними з огляду на логічно допустимий діапазон значень.

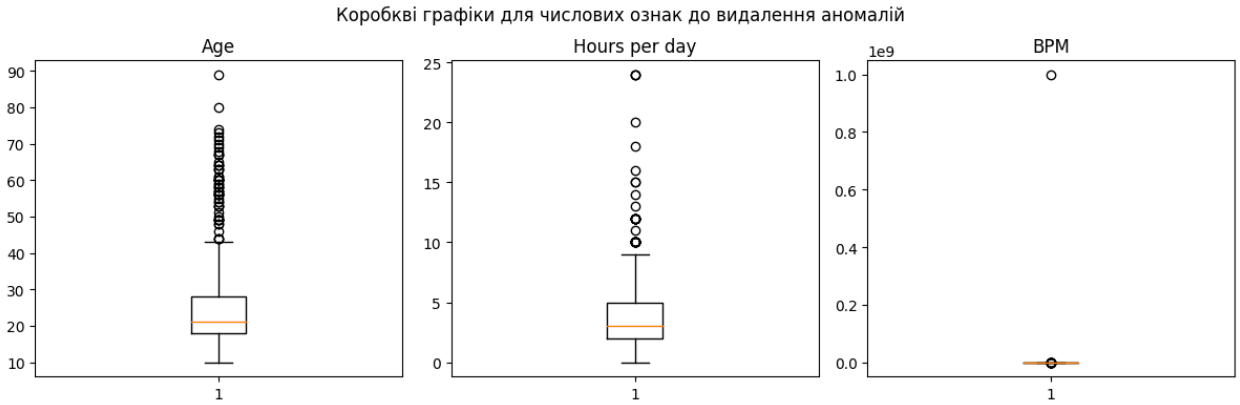


Рисунок 1.2 – Коробкові графіки для числових ознак

З огляду на критичну кількість відсутніх даних (значення NaN) за визначеним пороговим відсотком атрибути можуть видалятись [4]. Оскільки в жодному з 8 випадків частка NaN не перевищувала 15% (рис. 1.3), що є допустимим для даних такого рівня складності та обсягу інформації, кількість атрибутів датасету варто залишити без змін.

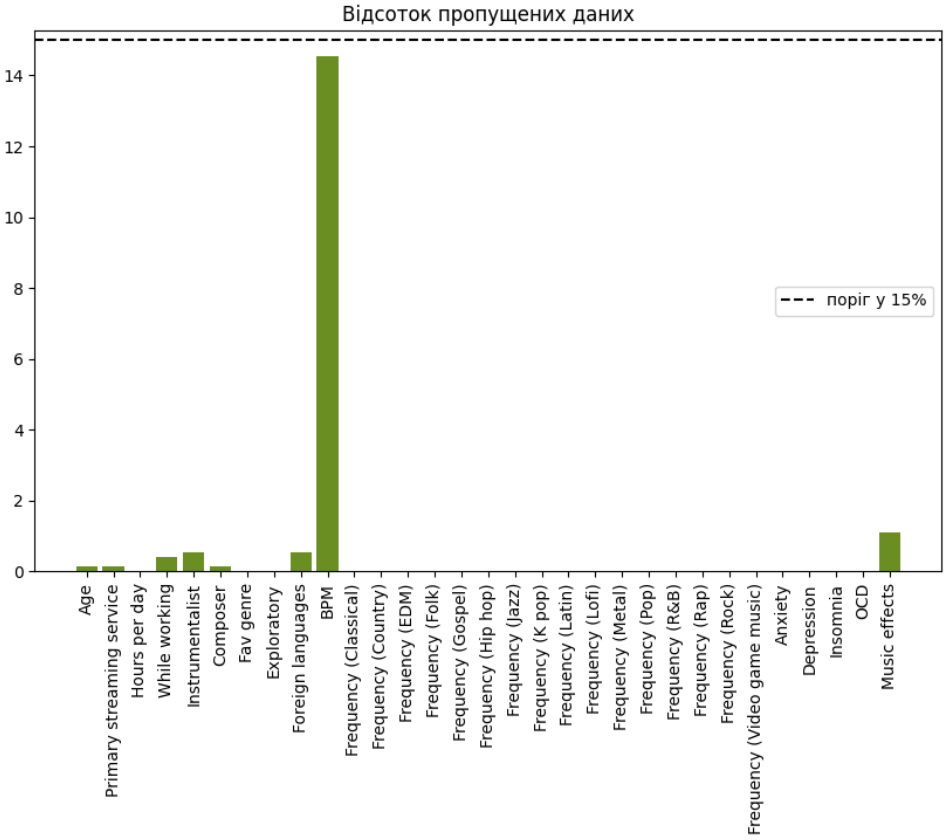


Рисунок 1.3 – Відсоток пропущених даних за ознаками

Побудова кореляційної матриці ознак (рис. 1.5) показала відсутність суттєвої кореляції за винятком нечисленних логічно споріднених факторів [5].

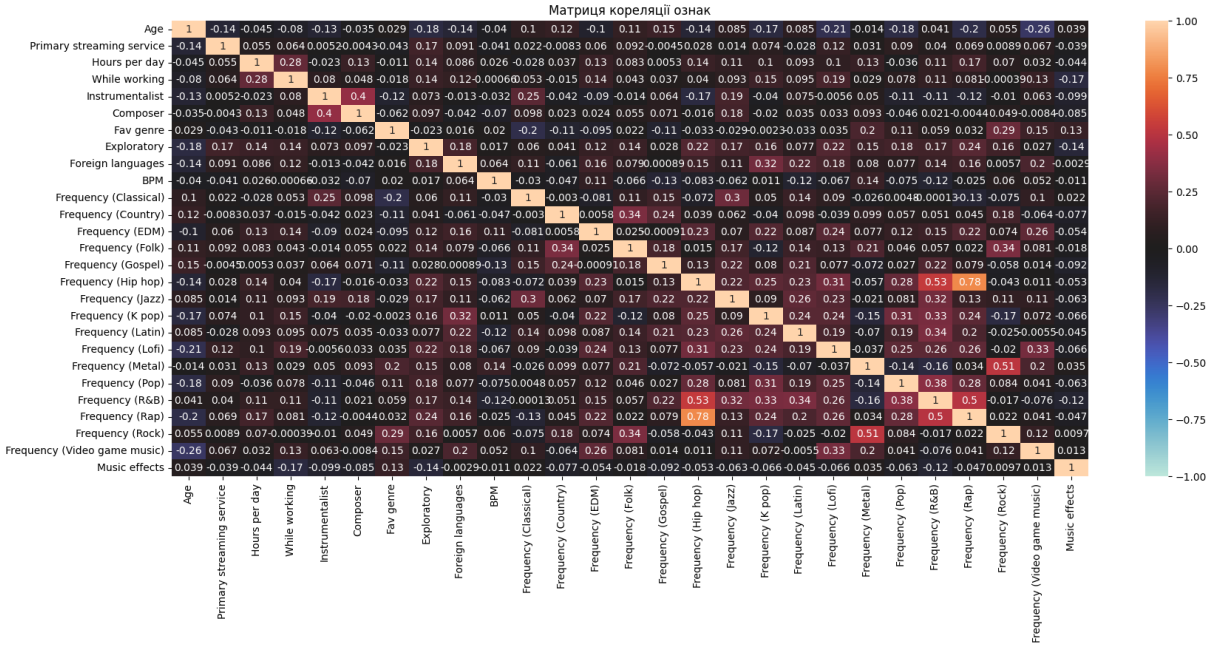


Рисунок 1.5 – Кореляційна матриця ознак

Побудова кореляційної матриці цільових змінних (рис. 1.6) показала суттєвий рівень кореляції, а отже доцільно здійснювати багатоцільовий прогноз [6], якщо це дозволяє архітектура обраного методу машинного навчання.

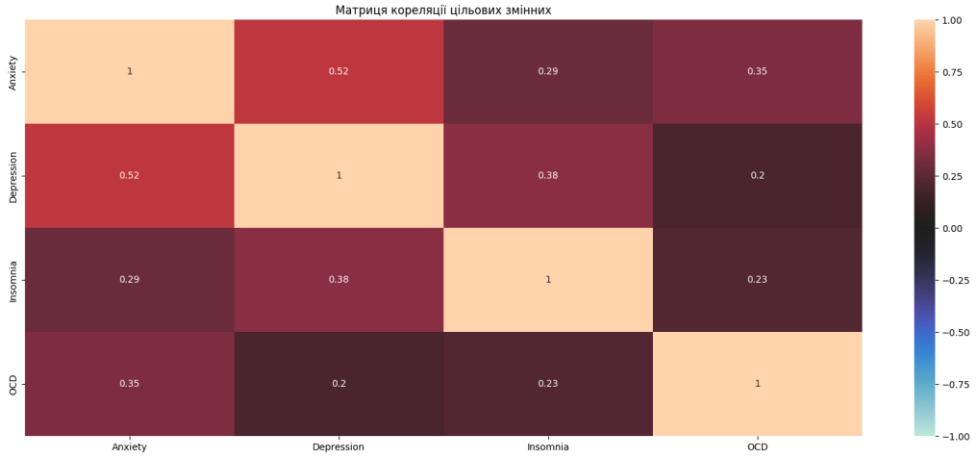


Рисунок 1.6 – Кореляційна матриця цільових змінних

З розподілу значень цільових змінних (рис. 1.7) видно, що значення можуть бути нецілими, багато цілих значень з доступного діапазону можуть бути відсутніми у зразках. Отже, маємо справу з прогнозуванням числового значення, а не розподілом результату до певної категорії (класу) [7].

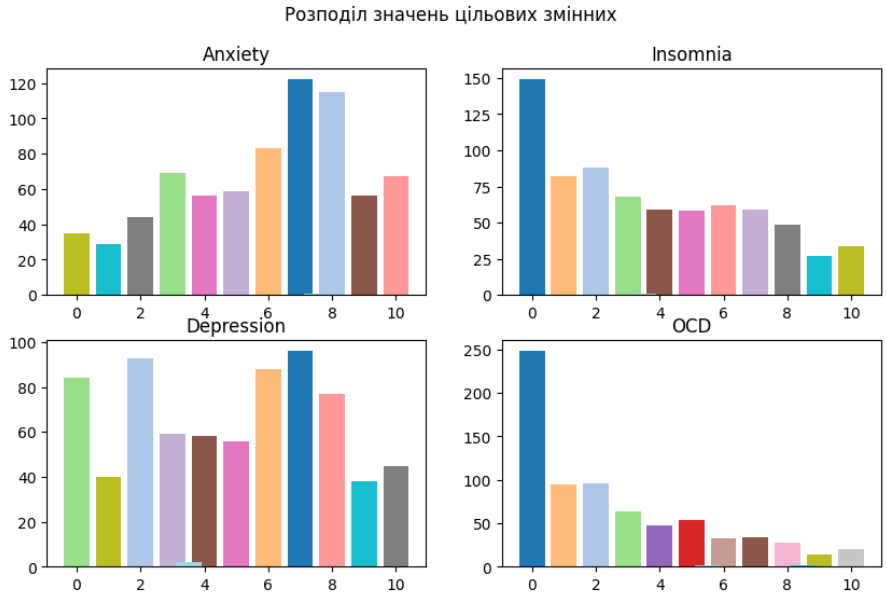


Рисунок 1.7 – Розподіл значень цільових змінних

1.4 Постановка задачі

Суть задачі полягає в прогнозуванні ступеня наявності тривожності, безсоння, депресії та обсесивно-компульсивного розладу в певної особи на основі інформації про її музичні уподобання та звички у прослуховуванні музики, яка представлена у вигляді 27 категоріальних та числових атрибутів. Ступінь наявності являтиме собою певну (не обов'язково цілу) оцінку від 0 до 10.

Формалізувати дану задачу можна наступним чином. Нехай X – множина ознак, а Y – множина цільових змінних. Існує невідома цільова залежність (відображення) $y^*: X \to Y$, значення якої відомі лише для об'єктів скінченної вибірки $X^m = \{(x_1; y_1), \dots, (x_m; y_m)\}$. Потрібно побудувати функцію $f: X \to Y$, яка допоможе прогнозувати оцінки станів для довільного зразка $x \in X$.

Оскільки прогнозується числове значення, а не належність до класу, маємо задачу регресії, що розв'язується у наступні кроки [8, с. 1-5].

1. Підготовка даних (див. 1.3). Оскільки маємо справу з табличними даними, необхідно провести попередню обробку: за потреби масштабувати дані, обрати значущі ознаки, опрацювати наявність аномалій, пропущених значень.
2. Розбиття даних. Дані розділяються на тренувальний, тестовий та валідаційний набори:
    1) тренувальна вибірка – вибірка, на основі якої відбувається налаштування параметрів моделі шляхом вивчення залежностей між ознаками та цільовою змінною;
    2) валідаційна вибірка – вибірка для підбору гіперпараметрів моделі (параметрів, що задаються перед початком навчання, а не налаштовуються в процесі навчання). Цей підбір дозволяє покращити отриманий результат та уникнути проблем на кшталт

перенавчання (перенавчена модель має слабку здатність до узагальнення залежностей та прогнозування, оскільки є занадто підлаштованою під специфіку тренувального набору);

3) тестова вибірка – незалежна вибірка для оцінки якості навчання моделі на тренувальній вибірці.

3. Формування набору альтернативних методів. Враховуючи переваги й недоліки відомих методів машинного навчання, обрати низку таких, що за своєю архітектурою підходять до наявного набору даних.
4. Побудова моделей за кожним методом. Необхідно налаштувати гіперпараметри та навчити моделі на тренувальному наборі даних. Також необхідно провести оцінку їхньої ефективності як на тестовій, так і тренувальній вибірці за допомогою відповідних метрик якості для уникнення перенавчання, виявлення інших проблем, зокрема помилок оцінювання та недостатнього обсягу використаних наборів даних.
5. Вибір найкращої моделі. Після виправлення можливих помилок та отримання остаточних свідчень про ефективність моделей різних методів маємо обрати найкращу.

1.5 Огляд методів розв’язку задачі

При роботі з табличними даними, що містять багато ознак (як категоріальних, так і числових), краще використовувати ансамблеві методи на основі дерев рішень.

Дерева рішень – це моделі, що розбивають дані на менші групи шляхом послідовного використання умовних розбиттів (рис. 1.8). У кінці кожної гілки дерева рішень зазвичай стоїть лист, який містить прогнозоване значення для цієї групи даних. Дані моделі використовуються як для задач класифікації, так і регресії [9].

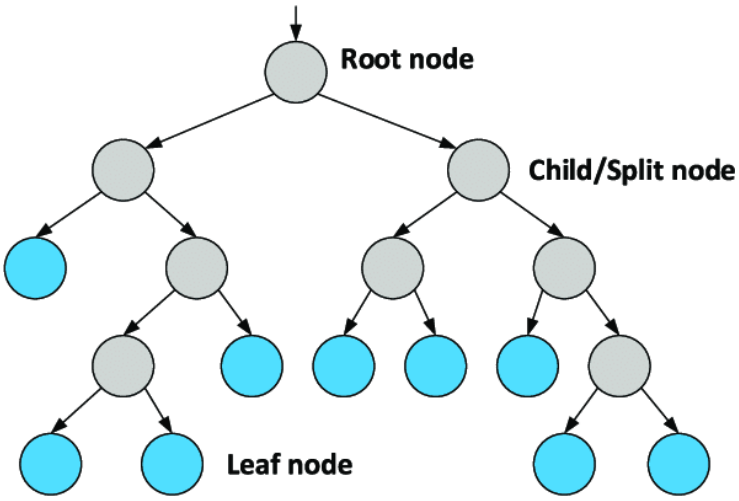


Рисунок 1.8 – Графічне представлення дерева рішень [10]

Ансамблеві методи – це моделі, що комбінують кілька дерев рішень для покращення точності прогнозування [11]. У порівнянні з поодиноким деревом рішень дають більш точний та незалежний прогноз, оскільки фінальний прогноз складається шляхом усереднення або голосування за прогнозами всіх дерев, що наочно демонструє рисунок 1.9.

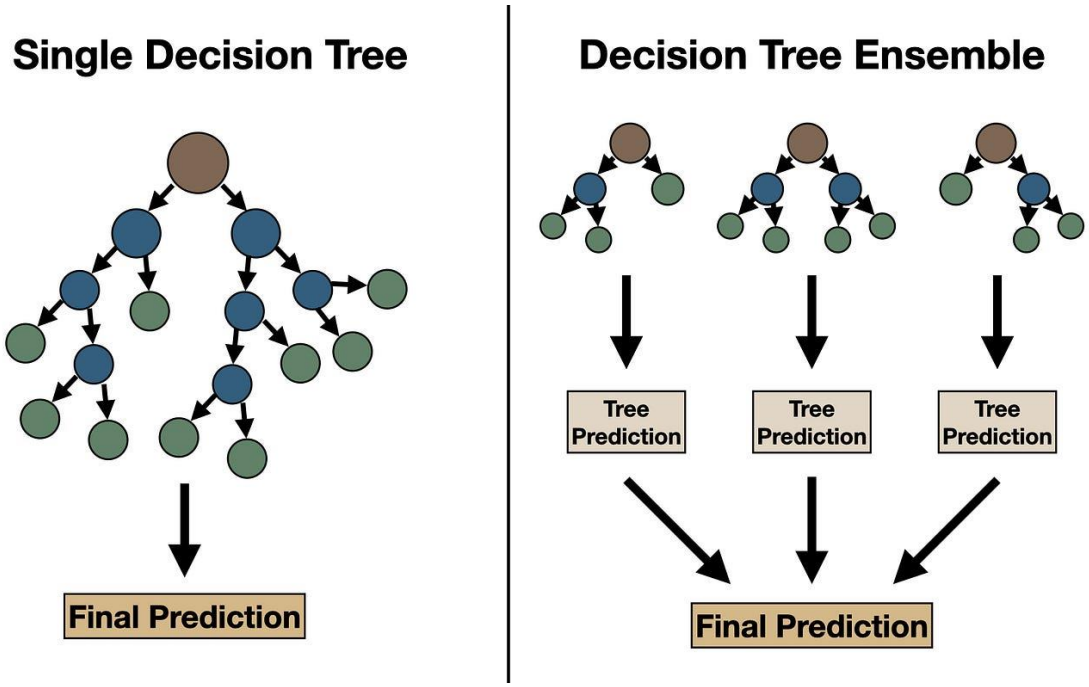


Рисунок 1.9 – Порівняння функціонування окремого дерева рішень та ансамблю дерев [11]

До ансамблевих методів відносяться такі алгоритми, як [12, с. 4]:

1) випадковий ліс – комбінація результатів навчання декількох дерев рішень, побудованих на основі випадкових підмножин даних та випадкових підмножин ознак (рис. 1.10);

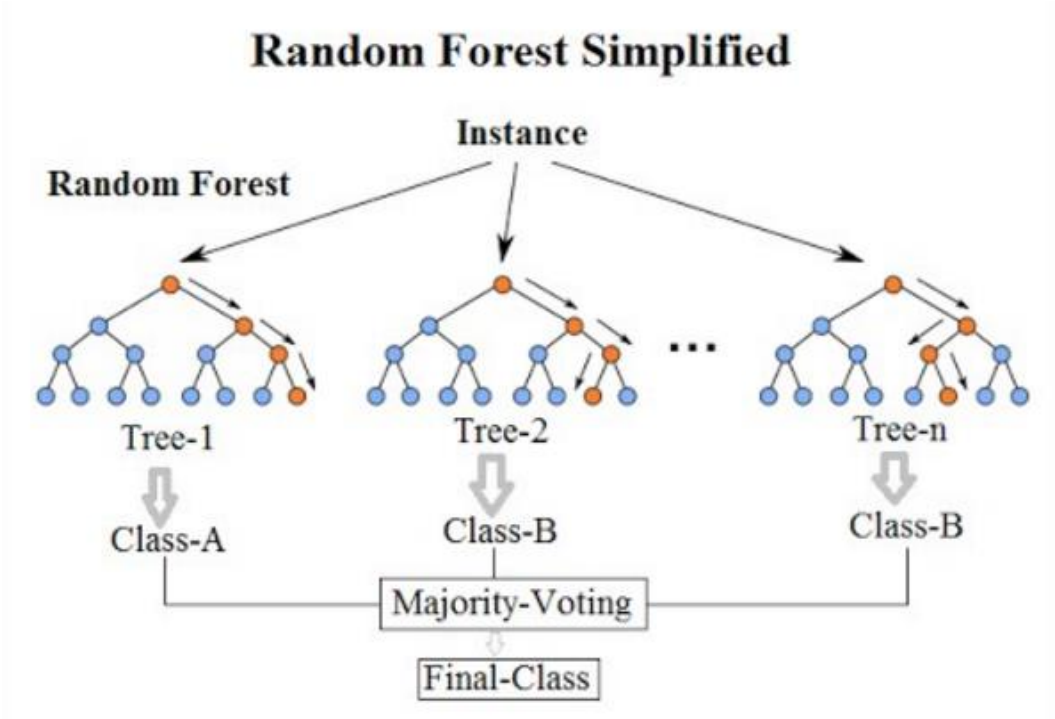


Рисунок 1.10 – Функціонування випадкового лісу [13]

2) бегінг – кожне дерево будується на випадкових множинах з повтореннями, що дозволяє кожному дереву отримати доступ до унікальних даних та зменшувати кореляцію між деревами;
3) градієнтний бустінг – на відміну від бегінгу (рис. 1.11), моделі будуються послідовно та фокусуються на зменшенні помилок свого попередника, для навчання кожної нової моделі використовується метод градієнтного спуску;

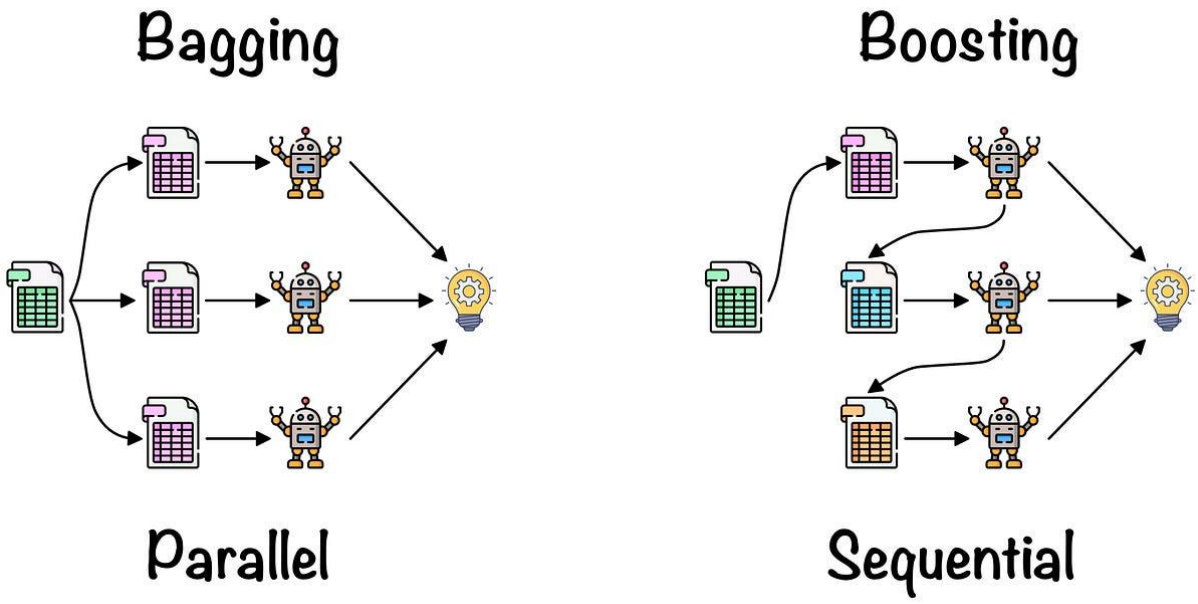


Рисунок 1.11 Графічна інтерпретація різниці між бегінгом та бустінгом [14]

4) стекінг – результати кожної моделі використовуються як вихідні дані для тренування більш складної моделі, яка може мати інший вигляд – наприклад, бути нейронною мережею (рис. 1.12).

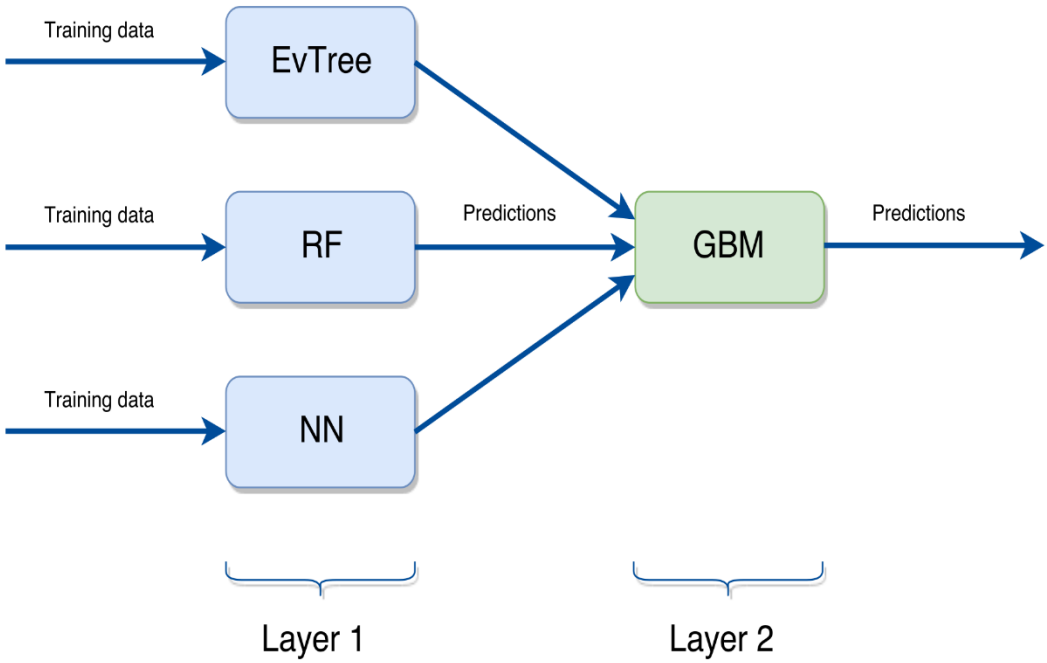


Рисунок 1.12 – Приклад стекінгової ансамблевої моделі [15, с. 6]

Ансамблеві методи мають наступні переваги [12, с. 15].

1. Покращення якості прогнозів. Навіть якщо окремі моделі є не дуже точними, їхня комбінація зазвичай показує дещо кращі результати. Ця властивість ансамблевих методів корисна у випадку, коли табличні дані містять багато ознак, що може ускладнити процес знаходження оптимальної моделі.
2. Наявність великої кількості категоріальних ознак. Більшість ансамблевих здатні автоматично обробляти категоріальні ознаки і враховувати їх взаємозв'язок з числовими ознаками.
3. Здатність вирішувати багатоцільові задачі. Ансамблеві моделі можна використовувати як для передбачення кожної цільової змінної окремо, так і для одночасного прогнозування значень змінних, не пов'язаних між собою.
4. Уникнення перенавчання моделі на табличних даних з великою кількістю ознак. Ансамблеві методи менш схильні до перенавчання, оскільки використовують випадкові підмножини даних та множини ознак, що знижує ризик детального вивчення набору даних. Також вони використовують регуляризацію (додаткове «штрафування» моделі за великі значення коефіцієнтів).

Метод опорних векторів (англ. support vector machine) розглядається у роботі як альтернатива ансамблевим методам, на порядок складніша за базові алгоритми на кшталт лінійної регресії. Алгоритм здатний працювати з даними великих обсягів та розмірності, тому також підходить для табличних даних.

Support Vector Regression (SVR) ефективний у випадку, якщо між ознаками та цільовою змінною не є лінійною. Цій властивості сприяє використання ядра для переведення ознак у вищу розмірність, в якій вони можуть бути лінійно розділені, як це показано на рисунку 1.13. Від правильності вибору ядра залежить точність пронозу. Метод також має здатність до управління перенавчанням завдяки використанню регуляризації.

Це допомагає уникнути використання надмірно складних моделей та покращити їхню універсальність [16].

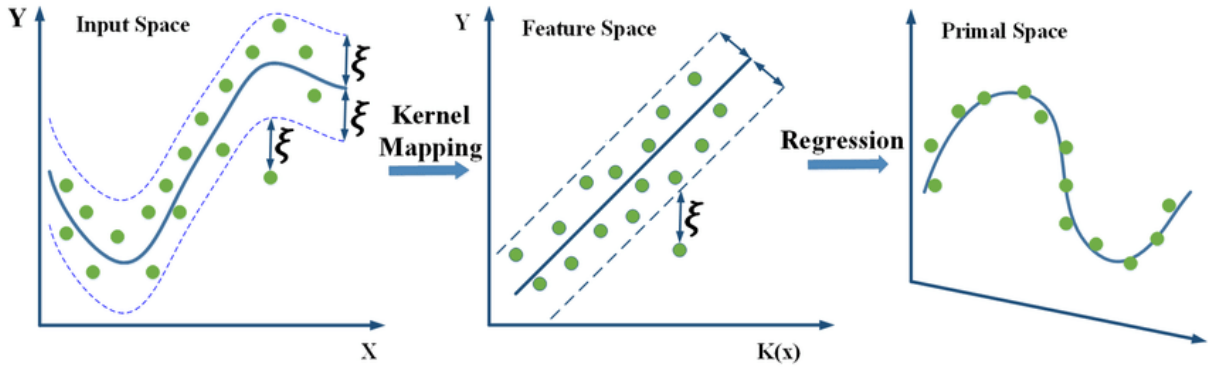


Рисунок 1.13 – Графічне представлення алгоритму SVR [17]

1.6 Висновки до розділу 1

У даному розділі було досліджено вихідні дані та методи розв’язання задачі прогнозування психічного здоров’я на основі даних про музичні вподобання.

Спочатку було обґрунтовано актуальність даної задачі на основі аналізу викликів реального світу спорідненої тематики. Наступним кроком було проведено детальний опис вихідних даних та проведено їхній аналіз за допомогою статистичних методів: визначено незалежні ознаки та цільові змінні, проведено аналіз внутрішніх розподілів та кореляції – з метою виявлення потенційної специфіки та недоліків. Заключним і головним етапом дослідження стала формалізація поставленої задачі та огляд наявних методів її розв’язку.

2 МАТЕМАТИЧНИЙ ОПИС МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ РЕГРЕСІЇ ТА ПРИКЛАДНИХ АЛГОРИТМІВ

У даному розділі наведено теоретичний матеріал за всіма використаними у роботі методами обробки та підготовки даних до навчання, налаштування моделей, прогнозування та оцінки його якості, а саме: сутність та область застосування методу, покроковий математичний опис алгоритмів, оцінка переваг та недоліків.

2.1 Методи попередньої обробки даних

Метою попередньої обробки даних є представлення у вигляді, якого потребують моделі машинного навчання. Спочатку застосовується унікальний набір кроків форматування та статистичних перетворень, що залежить від специфіки даних. Далі вживається класичний набір заходів: кодування категоріальних атрибутів, масштабування числових та вибір значущих ознак.

2.1.1 Кодування категоріальних атрибутів

Кодування категоріальних атрибутів проводиться для перетворення їх на числові на вимогу алгоритмів машинного навчання. При цьому має бути збережено унікальність закодованого значення та можливість декодування, тобто повернення до вихідного вигляду.

2.1.1.1 Однорозрядне кодування

Однорозрядне кодування (англ. one-hot-encoding, у даній роботі – вбудована функція get_dummies) передбачає перетворення кожного значення категоріальної змінної на бінарний вектор, що репрезентує його належність до певної унікальної категорії [18].

Нехай маємо множину категорій $C = \{c_1, c_2, \ldots, c_n\}$ для ознаки та m зразків. Після кодування отримуємо множину бінарних векторів $V = \{v_1, v_2, \ldots, v_m\}$, кожен k-ий вектор складається з n значень $v_k = \{a_1, a_2, \ldots, a_n\}$. Кожне i-те значення визначається за формулою:

$$a_i = \begin{cases} 1, & a_i \in c_j \\ 0, & \text{інакше.} \end{cases}$$

Перевагами OneHotEncoder є [19]:

1) збереження повної інформації про ознаку;
2) застосування до категорій з різними ступенями важливості;
3) сумісність зі всіма алгоритмами машинного навчання.

Недоліком OneHotEncoder є суттєве збільшення розмірності даних у випадку наявності великої кількості зразків та унікальних категорій ознак [19].

2.1.1.2 LabelEncoder

Основна ідея методу LabelEncoder полягає в тому, щоб кожному унікальному значенню категоріальної змінної призначити числовий ідентифікатор [18].

Нехай маємо множину категорій $C = \{c_1, c_2, \ldots, c_n\}$ для ознаки і множину відповідних числових значень $V = \{v_1, v_2, \ldots, v_n\}$. Тоді LabelEncoder перетворює кожну категорію c_i на відповідне числове значення v_i, тобто ми отримуємо відображення [19]:

$$f: c_i \to v_i, i \in \overline{1, n}.$$

Перевагами LabelEncoder є [19]:

1) легкість у використанні та інтерпретації;
2) швидкість виконання.

Недоліком LabelEncoder є нездатність алгоритму враховувати взаємозв'язки між категоріями, розглядати їх не лише як абсолютні, а й як відносні. Це може призвести до помилок інтерпретації результату, якщо за специфікою даних категорії не можна надавати у довільному порядку [19].

2.1.2 Масштабування числових ознак

Масштабування числових ознак прибирає різність розподілів, яка призводить до проблем у реалізації низки алгоритмів машинного навчання, оскільки велике значення певної ознаки робить її вплив на результат буде більш значущим за інші [20].

2.1.2.1 MinMaxScaler

MinMaxScaler – це метод масштабування даних, що дозволяє представити дані різного розподілу у діапазоні від 0 та 1 [20].

Для кожної ознаки $x \in X$, що має мінімальне значення x_{min} та максимальне значення x_{max}, формула для масштабування за допомогою MinMaxScaler виглядає наступним чином [18]:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}.$$

Переваги MinMaxScaler [20]:

1) зберігає властивості розподілу оригінальних даних;
2) простота у швидкому зіставленні, візуалізації нормалізованих даних;
3) підходить для роботи з алгоритмами, шо використовують евклідову відстань.

Недоліки MinMaxScaler [20]:

1) чутливість до аномальних значень (перед застосуванням алгоритму бажано виявити та видалити викиди);
2) втрата частини важливої інформації при суттєвому зменшенні масштабу.

2.1.2.2 StandardScaler

StandardScaler – це метод центрування та масштабування даних, що стандартизує дані, тобто змінює масштаб ознак таким чином, аби вони мали нульове середнє та одиничну дисперсію [20].

Для кожної ознаки $x \in X$, що має середнє значення μ та дисперсію σ, формула для масштабування за допомогою StandardScaler виглядає наступним чином [18]:

$$x_{scaled} = \frac{x - \mu}{\sigma}.$$

Переваги StandardScaler [20]:

1) зберігає властивості розподілу оригінальних даних;
2) ефективний у роботі з даними, що мають великий або суттєво різний діапазон значень;
3) ефективний при застосуванні алгоритмів машинного навчання, що використовують градієнтний спуск.

Недоліком StandardScaler є те, що метод менш ефективний для розподілів, далеких від нормального [20].

2.1.3 Методи вибору значущих ознак

Вибір значущих ознак дозволяє обґрунтовано зменшити їхню кількість, що позитивно впливає на час навчання моделі та її результативність з огляду на зменшення складності.

2.1.3.1 На основі кореляційної матриці

Кореляційна матриця – це матриця розмірності $n \times n$, яка містить кореляційні коефіцієнти Пірсона між кожною парою ознак у наборі даних [5]. Коефіцієнт кореляції вказує на ступінь та напрямок залежності однієї ознаки від іншої. Коефіцієнт кореляції має діапазон значень [-1;1], де -1 вказує на повну негативну кореляцію, 0 – на відсутність кореляції, а 1 – повну позитивну кореляцію [18].

Принцип вибору значущих ознак на основі кореляційної матриці полягає у наступному: якщо ознаки мають сильну кореляцію між собою, буде

корисно вибрати одну з них, аби зменшити розмірність даних і уникнути перенавчання моделі.

Окрім того, кореляційна матриця може допомогти виявити ознаки, які мають сильну кореляцію з цільовою змінною. Ці ознаки можуть бути корисними для побудови моделі, оскільки вони мають високу інформативність щодо прогнозування цільової змінної [21]. Також досліджують кореляцію між цільовими змінними у випадку наявності декількох (див 1.3).

Однак, кореляційна матриця може бути неінформативною, якщо всі ознаки слабо корелюють між собою або якщо наявна велика кількість ознак, які не мають кореляції з цільовою змінною. Також важливо, що кореляція не відображає причинно-наслідкові залежності, тому зробити висновки щодо походження залежностей між ознаками, спираючись на кореляційну матрицю, неможливо. В таких випадках корисно залучати методи машинного навчання для вибору значущих ознак [21].

2.1.3.2 На основі ансамблевих методів

RandomForestRegressor – це алгоритм, що використовує ансамблі дерев рішень для задач регресії. Принцип його роботи полягає в тому, що дерева рішень створюються на основі випадкової підвибірки ознак, що дозволяє зменшити зв'язок між деревами і підвищити точність прогнозування, знизити його варіативність [18].

Математичний опис алгоритму будем наступним [22, с. 176-178].

1. Для побудови кожного дерева T в ансамблі випадково обирається піднабір даних D та m ознак з їхнього загального числа n.
2. Для нового набору даних X робиться прогноз за допомогою середнього арифметичного з виходів з усіх дерев ансамблю. Якщо n – кількість

дерев у ансамблі, T – дерево рішень, а x_i – i-ий набір даних, то прогнозоване значення матиме вигляд:

$$y_{pred} = \frac{1}{n}\sum T(x_i).$$

Переваги RandomForestRegressor [22, с. 179]:

1) висока точність прогнозів;
2) зниження дисперсії моделі завдяки випадковому вибору підмножин даних та ознак;
3) можливість обробки великих наборів даних з багатьма ознаками.

Недоліки RandomForestRegressor [22, с. 180]:

1) значний час навчання та прогнозування у випадку великих наборів даних;
2) важкість інтерпретації загальної структури моделі через велику кількість дерев.

Ступінь важливості ознак за допомогою RandomForestRegressor визначається на основі частоти їхнього використання для розділення вузлів дерев рішень. Для отримання значущих ознак можна використовувати два підходи [21].

1. Встановлення порогу важливості. Задається певне відсоткове значення, перевищення якого свідчитиме про значущість ознаки. Цей метод відрізняється своєю простотою, проте потребує експертного визначення та вагомого обґрунтування.
2. Рекурсивне видалення. Метод RFECV (Recursive Feature Elimination with Cross-Validation) є більш надійним, оскільки полягає у призначенні ваги w для кожної з m ознак за ступенем важливості дерева T. Далі запускається процес видалення найменш важливих за вагою ознак з кожною ітерацією, при цьому використовується перехресна перевірка

для набору даних X, поки не буде досягнуто оптимального значення обраної метрики.

2.2 Методи прогнозування

У роботі розглянуто метод опорних векторів, як представник більш класичної регресійної моделі для даних поточного рівня складності, та різні регресори основних видів ансамблевих методів для подальшого порівняння їхньої результативності. У розділі 3 розглядаються класифікатори вищезгаданих методів та метод пошуку асоціативних правил.

2.2.1 SVR

SVR (Support Vector Regression) – це алгоритм машинного навчання, що знаходить гіперплощину, яка найкраще розділяє дані у просторі ознак. Це означає, що вона максимально відділяє точки даних від нуля. Для цього існує задача оптимізації, що мінімізує суму відхилень від гіперплощини за наступним алгоритмом [16].

1. Нехай задані функція відстані $D(x, x') = ||x - x'||$, функція ядра $K(x, x') = \varphi(x)^T \varphi(x')$, де φ – функція перетворення вихідних ознак в простір вищої розмірності.
2. Функція прогнозування має вигляд $f(x) = b + \sum w_i y_i K(x, x_i)$, де x_i, y_i – навчальний зразок та значення цільової змінної для цього зразка відповідно, w_i — ваговий коефіцієнтом, що визначає вклад відповідного вектора у результуюче передбачення, а b – відхилення.
3. Маємо наступну оптимізаційну задачу:

$$\frac{1}{2}w^T w + C\sum_n (x_i + x_i^*) \to min$$

$$y_i(w^T\varphi(x_i) + b) - \varepsilon \le y_i(w^T\varphi(x_i) + b) + \varepsilon,$$

де C – параметр регуляризації,

ε – параметр межі.

Переваги SVR [16]:

1) здатність працювати з даними, що мають велику кількість ознак та досить малий обсяг даних;
2) висока точність прогнозу на даних з високою ступеню шуму та складними залежностями.

Недоліки SVR [16]:

1) високі вимоги до налаштування параметрів;
2) складність у визначенні оптимального ядра;
3) потреба в стандартизації даних.

2.2.2 ExtraTreesRegressor

ExtraTreesRegressor – це алгоритм, що використовує ансамблі дерев рішень для задач регресії. Принцип його роботи полягає в тому, що він використовує випадкові розбиття для вузлів дерева замість звичайного пошуку найкращого розбиття. При цьому використовується повний набір даних, а не їхні підмножини [18].

Математичний опис алгоритму будем наступним [22, с. 182-183].

1. Випадково обирається m ознак з їхнього загального числа n, для кожної – поріг розділення p на основі діапазону значень даної ознаки.

2. Знаходиться ознака та поріг, які дають найбільшу якість при розділенні, всі зразки, для яких значення вибраної ознаки менше або дорівнює обраному порогу, потрапляють у перший дочірній вузол, а всі інші - у другий. Дерево T будується шляхом рекурсивного розділення вузлів до досягнення максимальної глибини дерева або до досягнення мінімальної кількості зразків в вузлі.
3. У процесі прогнозування i-ий зразок з цільовою функцією y_i переходить від кореня до листового вузла за допомогою порівнянь ознак та порогів.
4. Якщо b_i – середнє значення цільової змінної для об'єктів, які потрапляють в i-ий листовий вузол, R_i – регіон, визначений як послідовність умовних операторів над ознаками, що відбирають об'єкти, які потрапляють в цей вузол, а $I()$ – індикаторна функція, яка приймає значення 1, якщо вихідні дані належать відповідному регіону, та 0, якщо ні – то для нового набору даних X робиться прогноз за наступною формулою:

$$y_{pred} = \sum b_i I(x \in R_i).$$

Переваги ExtraTreesRegressor [22, с. 184]:

1) швидкість у порівнянні з іншими алгоритмами, оскільки випадкові розбиття дозволяють уникнути надмірної оптимізації та перенавчання моделі;
2) здатність працювати з даними, що містять багато шуму та високу кількість незначних ознак, оскільки такі бути виключені в процесі роботи алгоритму.

Недоліки ExtraTreesRegressor [22, с. 184]:

1) можливість недооцінки важливих ознак;
2) висока вразливість до шуму;

3) обмежена точність моделі на даних, що мають сильну залежність між ознаками.

2.2.3 BaggingRegressor

BaggingRegressor – це алгоритм, що використовує ансамблі дерев рішень для задач регресії. Принцип його роботи полягає в генерації кількох вибірок з навчальних даних і тренуванні на кожній з них окремої моделі [18].

Математичний опис алгоритму є наступним [23, с. 57-58].

1. На наборі даних з множиною ознак X та множиною цільових змінних Y розмірності n генерується k вибірок, обираючи n об'єктів з повторенням із вихідного набору X.
2. Для кожної вибірки створюється окрема модель регресії $F_k(X)$. Для передбачення нового значення X_{new} шукаємо усереднене значення передбачень для всіх k моделей:

$$F_k(X_{new}) = \frac{1}{k}\sum F_k(X_{new}).$$

Переваги BaggingRegressor [23, с. 59]:

1) зменшує перенавчання;
2) покращує стійкість моделі до випадкових збурень;
3) вигідний з точки зору обчислювальної складності, оскільки проводить паралельні операції.

Недоліки BaggingRegressor [23, с. 59]:

1) може займати більше часу на навчання через кількість натренованих моделей;

2) має гіршу інтерпретованість, оскільки результат передбачення усереднений.

2.2.4 XGBoost

XGBoost (eXtreme Gradient Boosting) – це алгоритм машинного навчання, що базується на градієнтному бустінгу та використовує дерева рішень. На відміну від бегінгу, принцип його роботи полягає в послідовному навчанні набору слабких моделей з поступовим покращенням їхніх результатів шляхом зменшення помилок прогнозування. Кожна наступна модель намагається покращити результати попередньої моделі, коригуючи її помилки [18].

Математично алгоритм можна подати у наступному вигляді [23, с. 23-26].

1. Маємо набір даних $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, де x_i – вектор ознак, а y_i – цільова змінна. Побудова моделі відбувається ітеративно, на m-му кроці будується дерево рішень $f_t(x)$, яке намагається скорегувати помилки попередніх дерев. Щоб змінити значення на t-му кроці, спочатку потрібно додати вагу кожному з n навчальних прикладів (x_i, y_i), згідно з їх попередніми значеннями:

$$w_i^{(t)} = \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}},$$

де $L(y_i, \hat{y}_i^{(t-1)})$ – функція втрат;

$\hat{y}_i^{(t-1)} = f_{t-1}(x_i)$ – прогнозоване значення на $(t-1)$-му кроці для i-го прикладу.

2. Будується дерево рішень, яке намагається знайти оптимальні розділення, щоб якомога краще підігнатися до ваг $w_i^{(t)}$. Для цього використовується функція втрат з градієнтним бустінгом:

$$obj^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t),$$

де L – функція втрат;

$\hat{y}_i^{(t-1)}$ – прогноз моделі на i-му навчальному зразку на $m-1$ ітерації;

$f_m(x_i)$ – прогноз дерева t для вихідних даних;

$\Omega(f_t)$ – функція регуляризації для дерева t.

3. Формули для розрахунку градієнта та гессіана виглядають наступним чином:

$$g_{it} = \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}},$$

$$h_{it} = \frac{\partial^2 L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i{}^{2(t-1)}}.$$

4. Застосовується метод другого порядку:
    1) обчислюється градієнт та гессіан функції втрат для кожного зразка в навчальному наборі даних;
    2) створюються дерева рішень для прогнозування помилки на попередній ітерації;
    3) обчислюються значення листків дерева, що мінімізують функцію втрат з урахуванням обчисленого градієнта та гессіана;
    4) оновлюються прогнози на основі значень листків дерева;

5) кроки 1-4 повторюються до досягнення заданої кількості дерев або заданої точності.

Переваги XGBoost [23, с. 27]:

1) висока швидкість та точність прогнозування;
2) можливість автоматичного визначення оптимальних гіперпараметрів.

Недоліки XGBoost [23, с. 27]:

1) складність настройки гіперпараметрів;
2) відносна складність інтерпретації результатів;
3) вразливість до перенавчання.

2.2.5 LightGBM

LightGBM (Light Gradient Boosting Machine) – це алгоритм машинного навчання, який базується на градієнтному бустінгу та використовує дерева рішень. Його основна ідея LightGBM полягає у використанні лише тих ознаки, які дійсно мають вплив на прогнозування значення цільової змінної [18].

Алгоритм використовує вертикальний підхід до побудови дерев. Це означає, що замість обробки всіх ознак в одному дереві, LightGBM розбиває їх на групи, на базі яких будує дерева рішень. Математично це подається у вигляді наступного набору кроків [22, с. 149-151].

1. Ініціалізується модель з нульовим значенням $F_0(x) = 0$.
2. Для кожного дерева $t \in T$ обчислюється градієнт g_{it} та гессіан h_{it}:

$$g_{it} = \frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)},$$

$$h_{it} = \frac{\partial^2 L\big(y_i, F_{t-1}(x_i)\big)}{\partial F_{t-1}^2(x_i)}.$$

3. Будується дерево рішень, максимізуючи функцію важливості ознак (зазвичай навчальний набір розбивається на дві частини, що максимально зменшує дисперсію градієнтів для об'єктів в кожній з частин). Цей процес описує наступна формула:

$$\varphi_j = \frac{\sum_{i \in R_j} g_{it}}{\sum_{i \in R_j} h_{it} + \lambda},$$

де $R_j -$ підмножина навчального набору, яка потрапляє до j-го листа;

$\lambda -$ гіперпараметр регуляризації.

4. Проводиться процес глобальної оптимізації, у ході якого виконується підбір оптимальних значеннь з мінімізацією функції втрат на навчальному наборі. Для цього використовуються методи оптимізації, такі як градієнтний спуск, стохастичний градієнтний спуск, адаптивний градієнтний спуск, а також байєсівська оптимізація.

Переваги LightGBM [22, с. 152]:

1) швидкість роботи та висока точність прогнозування;
2) здатність працювати з великими обсягами даних та зберігати великі обсяги інформації про кожен об'єкт;
3) здатність обробляти категоріальні ознаки без необхідності їх конвертації в числові значення;
4) гнучкість налаштування гіперпараметрів.

Недоліки LightGBM [22, с. 153]:

1) складність моделей часто призводить до перенавчання;
2) реалізація алгоритму потребує значного об’єму пам'яті та потужності обчислювальних ресурсів;
3) необхідність заповнення пропущених даних.

2.2.6 CatBoost

CatBoost (повна назва, яка не є скороченням) – це алгоритм машинного навчання, що базується на градієнтному бустінгу та використовує дерева рішень. Кожне дерево побудованого ансамблю навчається на збалансованій вибірці об'єктів з вагою, що враховує значимість кожної категоріальної ознаки, автоматично перетвореної на числову (англ. target encoding) [24].

Математично алгоритм можна подати у наступні кроки [23, с. 72].

1. Основна формула має вигляд:

$$F_M(x) = \sum_{m=1}^{M} \gamma_m h_m(x),$$

де $F_M(x)$ – прогнозоване значення на M-ому кроці;

γ_m – коефіцієнт, визначений у процесі градієнтного спуску;

h_m – дерево рішень, яке додається на кожному кроці.

2. Для кожного дерева $h_m(x)$ на кроці m спочатку обчислюються градієнт g_{it} і гессіан h_{it} для кожного зразка в навчальному наборі за формулами:

$$g_{it} = \frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)},$$

$$h_{it} = \frac{\partial^2 L\big(y_i, F_{t-1}(x_i)\big)}{\partial F_{t-1}^2(x_i)}.$$

3. Категоріальні ознаки кодуються за допомогою target encoding, зразки сортуються у листках дерев рішень відповідно до значення кодування. Для кожного листка находяться оптимальні значення критерію вибору розділу за наступною формулою:

$$\varphi_j = \frac{\sum_{i \in R_j} g_{it}}{\sum_{i \in R_j} h_{it} + \lambda},$$

де R_j – j-ий листок дерева;

λ – параметр реггуляризації.

4. Проводиться процедура аналізу перестановок для відбору найбільш інформативних ознак шляхом перевірки значень метрик на валідаційному наборі в залежності від перестановки значень на тренувальному (чим більша зміна метрики, тим вищий ранг ознаки).
5. Відбувається пошук найкращої комбінації гіперпараметрів. Для цього на кожній ітерації випадковим чином обираються їхні значення, після чого здійснюється перехресна перевірка моделі на кількох блоках валідаційного датасету. Параметри, що дають найкращий результат на валідаційній вибірці, використовуються на наступній ітерації навчання.
6. Проведення остаточного процесу навчання остаточний процес навчання з об'єднанням згенерованих дерев в ансамбль за допомогою градієнтного бустінгу.

Переваги CatBoost [23, с. 74]:

1) робота з даними, що містять багато категоріальних ознак;
2) використання різних регуляризацій, що зменшує ймовірність перенавчання та підвищує стійкість до шуму;
3) здатність автоматично розпізнавати та обробляти пропущені дані;
4) швидкість роботи та висока точність прогнозування.

Недоліки CatBoost [23, с. 74]:

1) високі вимоги до обчислювальних ресурсів;
2) тривалий час навчання моделі;
3) обмежена підтримка для нетипових, незбалансованих даних.

2.2.7 StackingCVRegressor

StackingCVRegressor (Stacking Cross-Validation Regressor) – це ансамблевий метод, що використовується для розв'язку задач регресії. Його суть полягає у комбінуванні кількох моделей для покращення якості прогнозування [18].

Математично метод можна описати наступним набором кроків [22, с. 95-97].

1. Навчальний набір X_{train} розбивається на два піднабори даних: перший використовується для навчання кожного прогнозатора першого рівня з множини $M = \{M_1, M_2, \ldots, M_n\}$; на другому (утриманому), що не брав участі у навчанні прогнозаторів, будуються прогнози першого рівня. Нехай утриманий набір даних налічує k зразків, тоді в результаті для кожного зразка цього набору отримується спрогнозоване значення для кожної n-ої моделі.
2. Створюється новий навчальний набір, що використовує прогнози попереднього рівня у якості вихідних ознак (матриця розмірності $k \times n$) і зберігає значення цільової змінної. Змішувач, або метамодель, навчається на цьому наборі, даючи на вихід остаточний прогноз цільової змінної.

Переваги StackingCVRegressor [22, с. 98]:

1) підвищення точності прогнозування;
2) використання перехресної перевірки (процесу навчання та оцінки якості моделей на різних комбінаціях підмножин даних, що дозволяє підвищити їхню ефективність та уникнути перенавчання)і;
3) використання різних алгоритмів машинного навчання у якості моделей, що підвищує різноманітність ансамблю.

Недоліки StackingCVRegressor [22, с. 95]:

1) потреба у великій обчислювальній потужності;

2) залежність від якості базових прогнозаторів та налаштування гіперпараметрів;
3) вразливість до перенавчання у випадку поєднання великої кількості моделей достатнього рівня якості;
4) зі збільшенням кількості моделей ансамблю збільшуються вимоги щодо кількості зразків у наборі даних.

2.2.8 Apriori

Алгоритм полягає у визначенні для масиву транзакцій низки асоціативних правил – висловлювань виду $X \to Y$, де X та Y є підмножинами деякої множини об'єктів O та у перетині являють собою порожню множину. Основна його ідея полягає у виявленні частої комбінації елементів, які можуть вказувати на корисні асоціації між елементами [25, с. 99].

Математично алгоритм можна подати у вигляді наступного набору кроків [25, с.101].

1. Для всієї множини об'єктів O генерується множина одноелементних кандидатів у часті набори C_1. З тих, що задовольняють встановлене мінімальне порогове значення підтримки, формується множина одноелементних частих наборів L_1.
2. Формуються множини кандидатів у часті набори C_k наступних $k = \overline{2,n}$ рівнів, що мають спільну $k-1$ кількість елементів з множиною частих наборів попереднього рівня. Як і на попередньому кроці, остаточну множину L_k складають лише ті кандидати, що мають достатню підтримку. Процес зупиняється, коли множина L_k буде порожньою.
3. Застосовується алгоритм association_rules, який перевіряє частий набір F з підмножини L_k множини частих наборів L на умову X та наслідок Y.

Отримана множина асоціативних правил R скорочується до такої, що відповідає встановленому мінімальному порогу достовірності.

Переваги Apriori [25, с.102]:

1) Простота реалізації;
2) Здатність працювати з великими масивами даних;
3) Здатність працювати з даними, що мають слабку кореляцію.

Недоліки Apriori [25, с.102]:

1) Велика кількість правил, що зумовлює потребу здійснення якісного відбору;
2) Складність пошуку оптимального значення підтримки.

2.3 Метрики якості

Метрики якості є величинами, що репрезентують певні аспекти якості моделей та дають змогу порівняти їхню результативність, визначити причину можливих неточностей.

2.3.1 Метрики регресії

Регресійні метрики порівнюють набори y (відомих значень цільової змінної) та y_{pred} (прогнозованих). З огляду на можливе різне уявлення про точність моделі (наприклад, дослідження або квадратичних, або абсолютних відхилень), погану сумісність з даними (наприклад, чутливість до викидів) та ступінь інтерпретованості результатів, варто застосовувати оптимальний за кількістю комплекс метрик, що дозволить отримати повне уявлення про результативність моделі та запобігти хибній інтерпретації [26].

2.3.1.1 MSE

Mean Squared Error (MSE) – це середнє значення квадратів різниць між прогнозованими значеннями та фактичними значеннями вибірки. Чим менше значення MSE, тим кращою є модель. Якщо n – кількість спостережень, то значення метрики обчислюється за формулою [18]:

$$MSE = \frac{1}{n}\sum(y - y_{pred})^2.$$

Простота обчислень зумовлює використання метрики у цілях оптимізації моделі регресії, наприклад, у процесі підбору гіперпараметрів, однак при цьому варто зважати на її чутливість до аномалій та надмірне «штрафування» великих помилок, порівняно з малими [26].

2.3.1.2 RMSE

Root Mean Squared Error (RMSE) є квадратним корінем від MSE ($RMSE = \sqrt{MSE}$). Значення метрики інтерпретується за аналогічним до MSE принципом [18].

Метрика є корисною у випадку присутності значних викидів даних, оскільки менш чутлива до них. У випадку наявності великих значень вихідних змінних результат метрики може бути легше інтерпретованим порівняно з MSE [18].

2.3.1.3 MAE

Mean Absolute Error (MAE) – це середня абсолютна похибка між прогнозованими та фактичними значеннями. цільової змінної. Діапазон можливих значень не є обмеженим зверху, проте має нижню межу 0 – найкраще значення. Якщо n – кількість спостережень, то значення метрики обчислюється за формулою [18]:

$$MAE = \frac{1}{n}\sum |y - y_{pred}|.$$

Метрика застосовується за потреби оцінити абсолютну точність моделі без урахування квадратичних відхилень, отриманий показник легко інтерпретується, оскільки має спільну одиницю виміру з вихідними даними Метрика не є чутливою до викидів, проте ігнорує значущість відхилень та специфіку їхнього розподілу [26].

2.3.1.4 R^2

Коефіцієнт детермінації (R^2) – це метрика, що виражає міру варіації прогнозованих значень від варіації фактичних значень. Діапазон можливих значень не є обмеженим знизу, проте має верхню межу 1. Значення 0 вказує на неспроможність до прогнозування, оскільки в такому випадку вона не пояснює жодної варіації цільової змінної, натомість значення 1 є індикатором ідеальної моделі. Від'ємні значення вказують на те, що модель пояснює дані гірше за усереднене значення цільової змінної. Якщо y_{mean} – це середнє від фактичних значень цільової функції, то метрика обчислюється за формулою [18]:

$$R^2 = 1 - \frac{\sum(y - y_{pred})^2}{\sum(y - y_{mean})^2}.$$

Значення метрики може давати хибні уявлення про якість моделі у випадках, коли дані мають високу кореляцію або велику кількість незначних ознак [26].

2.3.2 Метрики класифікації

Базовою метрикою для перевірки якості класифікаціє є точність, що є відношенням правильно класифікованих елементів до їхньої загальної кількості. Точність також можна розписати наступним чином [18]:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$

де TP – кількість правильно класифікованих позитивних екземплярів,

TN – кількість правильно класифікованих негативних екземплярів,

FP – кількість хибно класифікованих позитивних екземплярів,

FN – кількість хибно класифікованих позитивних екземплярів.

Діапазон значень метрики варіюється від 0 до 1, де 0 – найгірше значення, 1 – найкраще. Вона є легкою в інтерпретації, проте не враховує хибно позитивні та хибно негативні помилки, тому для аналізу більш високого рівня потрібно обчислювати також чіткості (англ. precision) – частки об’єктів, призначених до певного класу, яка насправді належить цьому класу – та повноти (англ. recall) – частки об’єктів, що справді належить до певного класу, яка була правильно призначена до цього класу.

2.3.3 Метрики пошуку асоціативних правил

Базовою метрикою для пошуку частих наборів є підтримка – частка транзакцій, в якій об’єкти зустрічаються разом, яка для набору A обчислюється за формулою [25, с.100]:

$$support(A) = \frac{|T_A|}{|T|},$$

де $|T_A|$ – кількість транзакцій, що містять даний набір,

$|T|$ – загальна кількість транзакцій.

Базовою метрикою для визначення важливості асоціативного правила є достовірність – частота спостереження отриманого правила, яка для набору умов X та наслідків Y обчислюється за формулою [25]:

$$confidence(X \to Y) = \frac{support(X \to Y)}{support(X)},$$

де $support(X \to Y)$ – частка правил, що містять X та Y разом,

$support(X)$ – частка правил, що містять X.

Оскільки описані метрики відображають частку, їхні значення варіюються від 0 до 1, де 0 – найслабше значення, 1 – найсильніше.

2.4 Підготовка моделей до навчання

Для навчання моделей необхідно визначитися із набором даних, що будуть вихідними для алгоритмів машинного навчання, та налаштувати

гіперпараметри. Від кількості зразків та репрезентативності набору залежить якість моделі, правильний підбір гіперпараметрів також впливає на цей показник.

2.4.1 Розділення даних на набори

Розділення даних необхідно для того, аби множини даних, на яких навчають модель та роблять прогноз із обчисленням метрик не перетинались. Гіперпараметри також бажано підбирати на окремому наборі.

2.4.1.1 train_test_split

train_test_split – це функція бібліотеки scikit-learn, що реалізує розділення набору даних на навчальну та тестову вибірки, що дозволяє навчати модель та оцінювати її якість на незалежних піднаборах даних [18].

Нехай маємо вихідну матрицю X розмірності $m \times n$ та вектор цільових змінних y розмірності m. Якщо k – бажана частка тестової вибірки від повного набору, то дані можна поділити наступному співвідношенні [18]:

1) тестовий набір X_{test} розмірності $m * k \times n$;
2) тренувальний набір X_{train} розмірності $m * (1 - k) \times n$.

Параметр random_state задає початковий стан генератора випадкових чисел. Використання константи дозволяє отримувати відтворювані результати у випадку повторного застосування функції.

Переваги train_test_split [27]:

1) простота перевірки ефективності моделі на даних, що не брали участь у навчанні моделі;

2) чітке розділення на вибірки, що є перевагою при наявності інших складних задач, наприклад, підбору гіперпараметрів моделі.

Недоліки train_test_split [27]:

1) на складних наборах даних результат значно залежить від вибору random_state;
2) мала інформативність щодо статистичної значущості різниці в ефективності між двома моделями;
3) нездатність виявляти складні взаємозв'язки між ознаками та цільовою змінною, що може призвести до перенавчання.

2.4.1.2 cross_validate

cross_validate – це функція бібліотеки scikit-learn, що реалізує алгоритм перехресної перевірки на даних, що дозволяє отримати більш точне уявлення про якість побудованої моделі [18].

Кроки реалізації алгоритму наступні [18].

1. Розбиття вихідних даних на задану кількість k рівних блоків (фолдів) з рівною кількістю зразків у кожному.
2. Почергове утримання k-го блоку для подальшого обчислення на ньому обраних метрик якості. Решта блоків у цей час беруть участь у навчанні заданої моделі, для них також можна обчислити метрики, якщо увімкнений параметр return_train_score.
3. Обчислення середнього значення метрик по всіх блоках.

Переваги cross_validate [27]:

1) підвищення об’єктивності результату за рахунок перевірки на незалежних вибірках;
2) дозволяє проводити модифікації класичної перехресної перевірки;
3) дозволяє задавати набір цільових метрик.

Недоліки cross_validate [27]:

1) час обчислення збільшується пропорційно до кількості фолдів;
2) відсутність можливості змінювати розмір тренувальної та тестової вибірок;
3) відсутність можливості зафіксувати певне розбиття.

2.4.2 Підбір гіперпараметрів

GridSearch – це метод оптимізації гіперпараметрів, який здійснює пошук таого набору гіперпараметрів моделі, що дає найкращу результативність на перевірочному наборі даних. GridSearchCV (Grid Search Cross-Validation) є удосконаленням алгоритму з використанням перехресної перевірки [18].

Алгоритм виконується у наступні кроки [27].

1. Побудова сітки гіперпараметрів $S = \{s_1, s_2, \dots, s_n\}$ (n – кількість гіперпараметрів, s_i – вектор значень) за принципом рівномірного кроку або за набором вказаних значень для кожного гіперпараметра.
2. Розбиття тренувального набору на K блоків для проведення перехресної перевірки.
3. Для кожного набору гіперпараметрів g_i з матриці G можливих комбінацій n гіперпараметрів за сіткою S):
    1) кожен з K блоків утримується для перевірки точності моделі на ньому, решта блоків у цей час задіяні у процесі навчання моделі;
    2) обчислюється середня точність за всіма блоками.
4. Обирається набір гіперпараметрів g_i, що максимізує середню точність.
5. Модель з обраними гіперпараметрами навчається на повному тренувальному наборі.
6. Оцінюється точність навченої моделі на тестовій вибірці.

Переваги GridSearchCV [28]:

1) автоматичне знаходження оптимальної комбінації параметрів для моделі;
2) застосування перехресної перевірки.

Недоліки GridSearchCV [28]:

1) велика часова вартість обчислень у випадку великої кількості комбінацій, представлених до розгляду;
2) відсутність гарантії знаходження глобального екстремуму метрики;
3) схильність до перенавчання за неправильної побудови сітки гіперпараметрів.

2.5 Висновки до розділу 2

У даному розділі було розглянуто математичні основи методів, обраних для розв’язання поставленої задачі, та супутніх алгоритмів підготовки даних та моделей до навчання, метрик якості.

Спочатку було досліджено сутність методів попередньої обробки даних, що використовувались для кодування категоріальних ознак, масштабування даних та вибору значущих ознак. Наступним кроком було описано альтернативні методи машинного навчання для прогнозування: представники основних різновидів ансамблевих методів, метод опорних векторів та пошуку асоціативних правил. Для кожного методу було викладено сутність, покроковий формульний опис реалізації та перелік переваг, недоліків. Заключним етапом було наведено опис використаних метрик оцінювання якості моделей та методів підготовки моделей до навчання, зокрема методів розділення вихідних даних на робочі вибірки та підбору гіперпараметрів моделі.

3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ МОДЕЛЕЙ ПРОГНОЗУВАННЯ ОЦІНОК МЕНТАЛЬНОГО ЗДОРОВ'Я

У даному розділі оглядово наводиться попередня обробка даних та налаштування значущих гіперпараметрів моделей з метою покращення їхньої результативності за рахунок більшої пристосованості до обраних даних. Основною задачею є навчання побудованих моделей, порівняння їхньої результативності та реалізація методів покращення якості прогнозування.

3.1 Підготовка даних до навчання

Існує декілька стратегій щодо підготовки даних до навчання [29]:

1) модифікація даних на основі експертного статистичного аналізу;
2) приведення даних до формату, якого вимагають алгоритми, за усталеним набором кроків;
3) варіація кроків під час навчання моделей, порівнюючи результат.

З огляду на складність та обсяг вихідних табличних даних доцільно обрати перший варіант.

Спочатку було проведено форматування назв атрибутів: замінено наявні квадратні дужки на круглі для сумісності даних з деякими алгоритмами (див. 1.3).

Після видалення явної аномалії в атрибуті «BPM» (див. 1.3) було побудовано оновлені коробкові графіки (рис 3.1) для перевірки прийнятності інших аномалій з точки зору сумісності з логічно допустимим діапазоном значень.

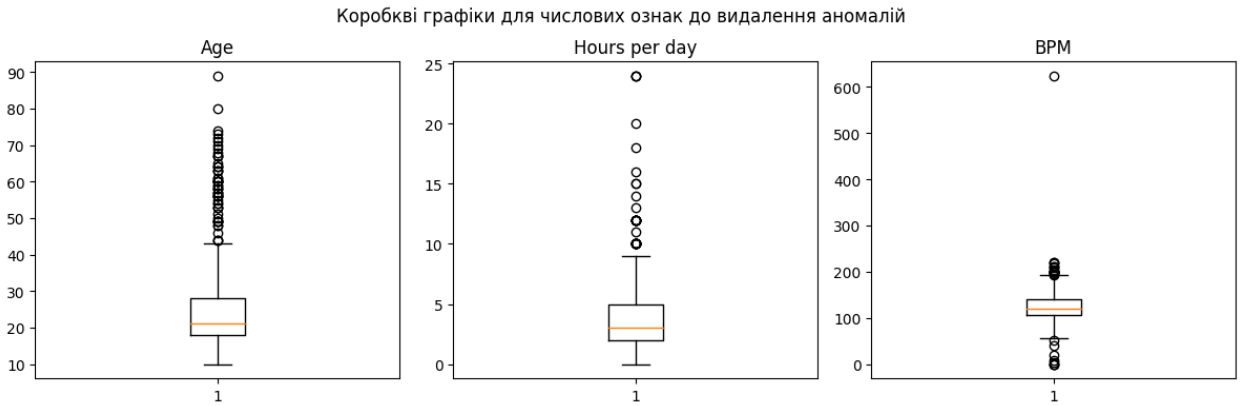


Рисунок 3.1 – Коробкові графіки для числових ознак після видалення аномалій

Значущість ознак було визначено за допомогою рекурсивного видалення на основі ансамблевого методу RandomForestRegressor (рис. 3.2), оскільки цей метод виявився абсолютно кращим за експертно обране порогове значення (див. 2.1.3.2) для експериментально побудованих базових моделей.

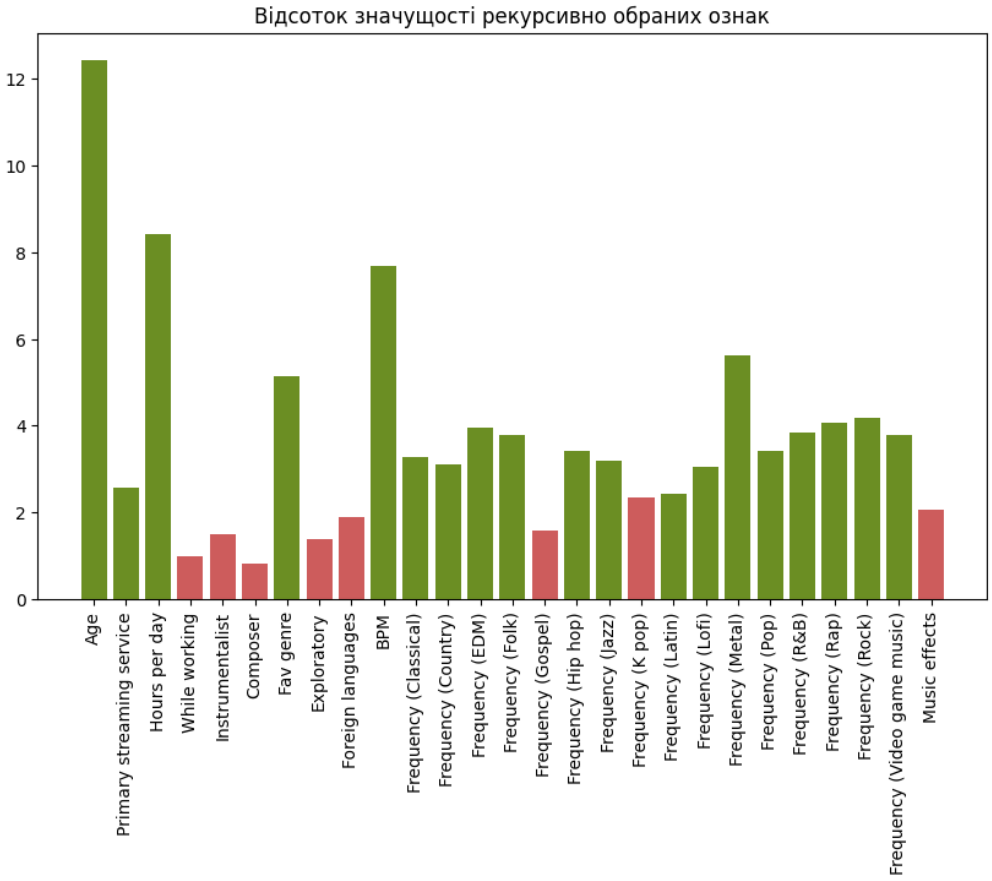


Рисунок 3.2 – Рекурсивно обрані значущі ознаки (зеленим кольором)

Пропуски для даних типу float64 (числові ознаки) було заповнено медіаною, оскільки такий варіант є найбільш вигідним у випадку наявності аномалій (порівняно з заповненням нульовим значенням або середнім) [30]. Дані типу object (категоріальні ознаки) було заповнено найбільш частим значенням, нові значення (наприклад «Пропущено») не вводились з огляду на невеликий відсоток пропущених даних та їхню специфіку.

Категоріальні дані було закодовано за допомогою функціоналу Label Encoding бібліотеки scikit-learn, оскільки такого формату потребує більшість моделей машинного навчання. Числові дані, з огляду на розподіл (рис. 3.3), було масштабовано. З огляду на складність даних доцільно реалізувати обидва методи метод (стандартизація або нормалізація) та обрати найкращий за результативністю.

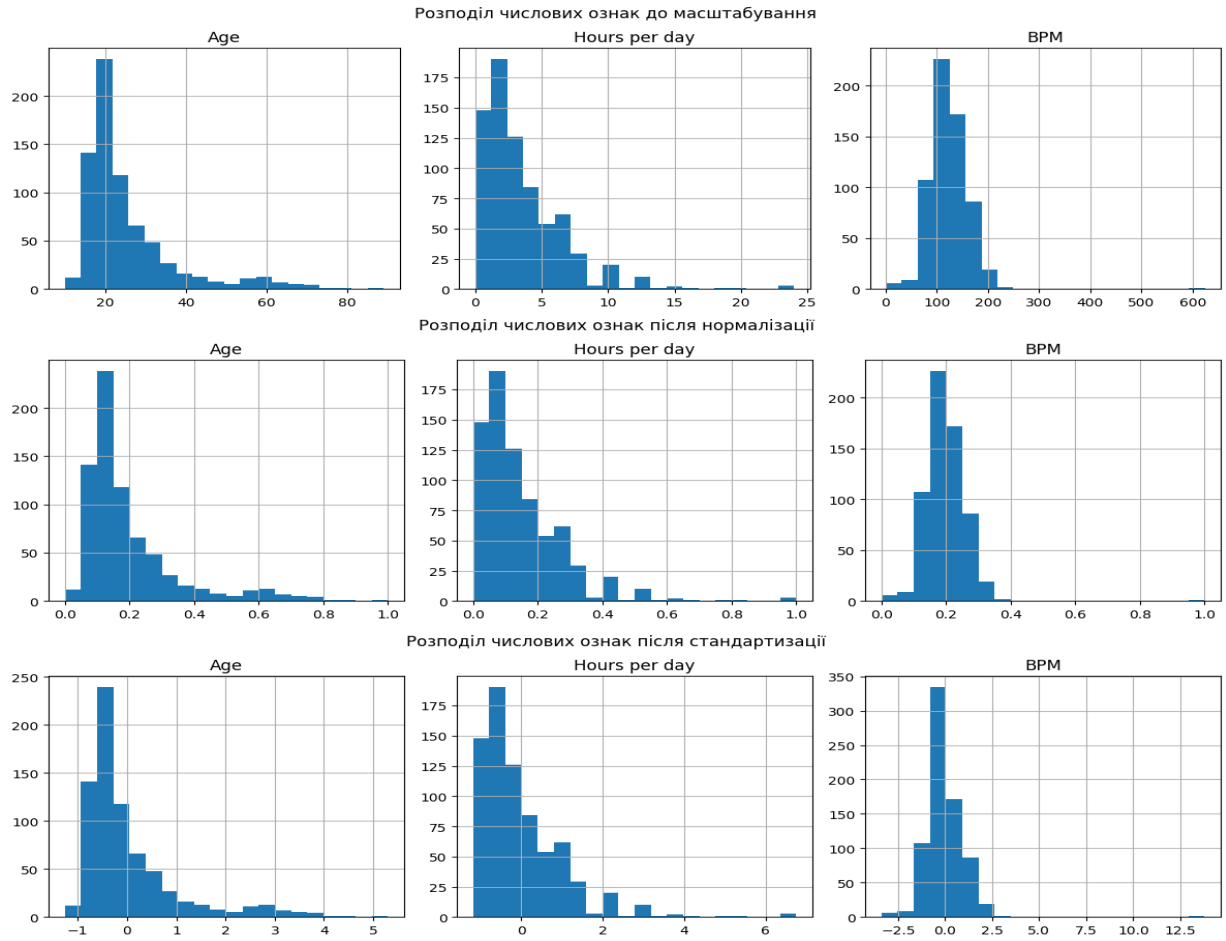


Рисунок 3.3 – Варіації масштабування числових даних

З огляду на кількість ознак та слабку кореляцію між ними (див. 1.3), велику кількість зразків, а головне – використання перехресної перевірки багатьма ансамблевими методами, дані було розділено на набори (див. 1.4) за заданим співвідношенням 70:15:15.

3.2 Підбір гіперпараметрів моделей

З переліку гіперпараметрів, запропонованих бібліотеками scikit-learn [18] та CatBoost [24] для розглянутих методів, було обрано найсуттєвіші для процесу навчання моделі. Для кожного з них було розглянуто найдоцільніші варіанти значень. За найкращий набір гіперпараметрів для кожної моделі було обрано комбінацію, що дає мінімальну середньоквадратичну похибку.

3.2.1 Метод опорних векторів

Для регресора, побудованого на основі методу опорних векторів, було підібрано значення наступних гіперпараметрів (табл. 3.1):

1) «kernel»: тип ядра;
2) «C»: коефіцієнт ядра;
3) «gamma»: параметр регуляризації;
4) «epsilon»: похибку.

Таблиця 3.1 – Набір гіперпараметрів для SVR

Назва гіперпараметра	Розглянуті значення	Найкраще значення
kernel	linear, poly, rbf	rbf

Продовження таблиці 3.1

Назва гіперпараметра	Розглянуті значення	Найкраще значення
C	0.1, 1, 10	1
gamma	scale, auto	auto
epsilon	0.1, 0.2, 0.3	0.3

3.2.2 Випадковий ліс

Для моделей випадкового лісу (табл. 3.3) було підібрано значення наступних гіперпараметрів:

1) «n_estimators»: кількість дерев;
2) «max_depth»: максимальну глибину дерева;
3) «min_samples_split»: мінімальну кількість зразків, необхідну для розділення вузла;
4) «min_samples_leaf»: мінімальну кількість зразків у листовому вузлі;
5) «max_features»: спосіб визначення кількості ознак, шо враховуються при пошуку найкращого розбиття.

Таблиця 3.2 – Набір гіперпараметрів для RandomForestRegressor

Назва гіперпараметра	Розглянуті значення	Найкраще значення	
		RandomForestRegressor	ExtraTreesRegressor
n_estimators	100, 200, 300	200	100
max_depth	None, 5, 10	None	5
min_samples_split	2, 5, 10	10	10
min_samples_leaf	1, 2, 4	4	4
max_features	auto, sqrt, log2	sqrt	log2

3.2.3 Бегінг

Для бегінгової моделі було обрано випадковий ліс у якості базового оцінювача та підібрано значення наступних гіперпараметрів (табл. 3.3):

1) «n_estimators»: кількість дерев ансамблю;
2) «max_samples»: частку використаних зразків з навчального набору;
3) «max_features»: частку використаних ознак;
4) «bootstrap»: необхідність заміни зразків;
5) «bootstrap_features»: необхідність заміни ознак.

Таблиця 3.3 – Набір гіперпараметрів для BaggingRegressor

Назва гіперпараметра	Розглянуті значення	Найкраще значення
n_estimators	50, 100, 150	50
max_samples	0.6, 0.8, 1.0	0.8
max_features	0.6, 0.8, 1.0	0.6
bootstrap	True, False	False
bootstrap_features	True, False	False

3.2.4 Градієнтний бустінг

Для моделей градієнтного бустингу – XGBRegressor (табл. 3.4), LGBMRegressor (табл. 3.5), CatBoostRegressor (табл. 3.6) – було підібрано значення певної, доцільної з огляду на специфіку методу, вибірки з наступних гіперпараметрів:

1) «n_estimators»: кількість дерев ансамблю;
2) «learning_rate»: швидкість навчання;

3) «max_depth»: максимальну глибину дерева;
4) «subsample»: співвідношення підвибірок навчальних зразків;
5) «colsample_bytree»: співвідношення підвибірок ознак, що використовуються при побудові дерев;
6) «gamma»: мінімальну втрату для необхідності розбиття у листовому вузлі;
7) «reg_alpha» та «reg_lambda»: коефіцієнти регуляризації L1 та L2 відповідно.

Таблиця 3.4 – Набір гіперпараметрів для XGBoostRegressor

Назва гіперпараметра	Розглянуті значення	Найкраще значення
n_estimators	100, 200, 300	300
learning_rate	0.1, 0.01, 0.001	0.01
max_depth	3, 5, 7	7
subsample	0.6, 0.8, 1.0	0.6
colsample_bytree	0.6, 0.8, 1.0	0.6
gamma	0, 0.1, 0.5	0.1
reg_alpha	0, 0.1, 0.5	0.1
reg_lambda	0, 0.1, 0.5	0.5

Для LGBMRegressor параметр «gamma» не представлений.

Таблиця 3.5 – Набір гіперпараметрів для LGBMRegressor

Назва гіперпараметра	Розглянуті значення	Найкраще значення
n_estimators	100, 200, 300	100
learning_rate	0.1, 0.01, 0.001	0.001
max_depth	3, 5, 7	3

Продовження таблиці 3.5

Назва гіперпараметра	Розглянуті значення	Найкраще значення
subsample	0.6, 0.8, 1.0	0.6
colsample_bytree	0.6, 0.8, 1.0	0.6
reg_alpha	0, 0.1, 0.5	0.5
reg_lambda	0, 0.1, 0.5	0.5

Для CatBoostRegressor параметр «gamma» не представлений, деякі параметри мають інші назви («iterations» замість «n_estimators», «colsample_bylevel» замість «colsample_bytree»).

Таблиця 3.6 – Набір гіперпараметрів для CatBoostRegressor

Назва гіперпараметра	Розглянуті значення	Найкраще значення
iterations	100, 200, 300	300
learning_rate	0.1, 0.01, 0.001	0.1
depth	3, 5, 7	7
subsample	0.6, 0.8, 1.0	0.8
colsample_bylevel	0.6, 0.8, 1.0	1.0
reg_lambda	0, 0.1, 0.5	0.5

3.2.5 Стекінг

Для стекінгової моделі у якості базових оцінювачів було обрано всі налаштовані моделі градієнтного бустінгу (див. 3.1.4), а у якості метамоделі – налаштовану модель випадкового лісу ExtraTreesRegressor (див 3.1.2).

3.3 Порівняльний аналіз результативності моделей

Для алгоритмів, що дозволяють багатоцільовий прогноз (див 1.3), метрики обчислювались за єдиним отриманим результатом. Для решти у таблиці наводиться усереднене значення за чотирма моделями для кожної цільової змінної, дані розбивались на тренувальну та тестову вибірки у пропорції 80:20.

Для обраних даних масштабування ознак StandardScaler виявилось кращим за застосування MinMaxScaler (див 1.3), тому порівняння результатів для всіх методів наводити недоцільно.

Метрики якості обчислюються як для тренувального, так і для тестового набору, оскільки за характером співвідношення цих значень можна визначити причину проблематики моделі. Результати дослідження наводяться у таблиці 3.7.

Таблиця 3.7 – Результати регресії для побудованих моделей

Модель	Метрики якості						Час навчання, с
	RMSE		MAE		R2		
	train	test	train	test	train	test	
SVR	2,565	3,149	1,865	2,489	0,212	-0,173	0,03
ExtraTreesRegressor	2,715	2,905	2,313	2,458	0,154	0,018	0,16
BaggingRegressor	2,737	2,894	2,331	2,448	0,141	0,013	5,98
XGBRegressor	1,402	2,967	1,147	2,477	0,774	-0,008	1,2
LGBMRegressor	2,878	2,921	2,426	2,455	0,013	-0,03	0,02
CatBoostRegreesor	2,38	3,031	1,983	2,517	0,323	-0,068	0,15
StackingCVRegressor	2,788	2,928	2,347	2,46	0,067	-0,015	4,21

Найкращою результат за метриками показала модель BaggingRegressor, що використовує ExtraTreesRegressor як базовий оцінювач. Час його виконання є більшим за інші алгоритми, проте ця різниця становить декілька секунд, тому не є суттєвою. Такий результат можна пояснити випадковістю розбиттів та незалежним навчанням дерев з подальшою агрегацією прогнозу, що попереджує перенавчання, а також здатністю алгоритму знаходити складні нелінійні залежності.

Утім, навіть для найкращої моделі має місце дуже велика помилка регресії та ознаки випадковості прогнозу. Походження цієї проблеми може бути наступним:

1) використання некоректних методів розбиття даних;
2) неправильне застосування метрик;
3) неправильна інтерпретація задачі;
4) низька якість даних з точки зору кореляції.

Обрані дані відносяться до категорії соціологічних і мають суб’єктивний характер, тому найімовірнішою причиною низьких показників є 4-та. Утім, аби виключити причини 1-3, доцільно провести удосконалення моделей за цими ознаками.

3.4 Покращення отриманих результатів

Оскільки було проведено якісну підготовку даних (див. 3.1) та підбір гіперпараметрів (див 3.2), для покращення результатів необхідно працювати над переосмисленням розбиття даних на набори, обробкою отриманого вектора прогнозованих значень або альтернативними методами прогнозування.

3.4.1 Перехресна перевірка

Оскільки стале розбиття на тренувальну та тестову вибірки за вказаним співвідношенням є випадковими, доцільно використовувати перехресну перевірку для розглянутих метрик при навчанні моделі та прогнозуванні [31]. Результати дослідження наводяться у таблиці 3.8.

Таблиця 3.8 – Результати регресії для побудованих моделей з перехресною перевіркою

Модель	Метрики якості						Час навчання, с
	RMSE		MAE		R2		
	train	test	train	test	train	test	
SVR	2,513	2,989	1,925	2,45	0,264	-0,051	0,02
ExtraTreesRegressor	2,665	2,897	2,267	2,463	0,175	0,013	0,09
BaggingRegressor	2,709	2,897	2,305	2,466	0,148	0,011	6,53
XGBRegressor	1,436	2,929	1,174	2,466	0,76	-0,009	0,62
LGBMRegressor	2,911	2,932	2,482	2,503	0,016	-0,01	0,02
CatBoostRegreesor	2,403	2,916	2,02	2,449	0,329	0	0,05
StackingCVRegressor	2,706	2,908	2,297	2,48	0,147	0,006	3,25

Перехресна перевірка мінімально (з точністю до сотих) покращила результати, при цьому найкращою моделлю став ExtraTreesRegressor, оскільки у попередній (див. 3.3) сталась накладка розбиттів (зовнішня та вбудована в алгоритм), що зробило вибірки менш репрезентативними.

Загалом отриманий результат свідчить про те, що погана результативність не була пов’язана з помилковістю рішення щодо сталих

розбиттів даних на підмножини, які могли бути сформовані на основі нерепрезентативних або малозначущих зразків.

3.4.2 Округлення прогнозованих значень

Погані показники метрик можуть бути пов’язані з тим, що прогнозами регресії є певні раціональні числа, у той час як фактичними значеннями цільових змінних даної задачі були переважно цілими. Саме тому при розв'язанні прогнозування оцінок існує практика округлення масиву прогнозованих значень із заданою точністю (у даному випадку 0,5) з метою поліпшення результатів [32, с. 4-5]. Результати дослідження наводяться у таблиці 3.9.

Таблиця 3.9 – Результати регресії для побудованих моделей з округленням прогнозованих значень

Модель	Метрики якості						Час навчання, с
	RMSE		MAE		R2		
	train	test	train	test	train	test	
SVR	2,589	3,146	1,82	2,481	0,197	-0,174	0,02
ExtraTreesRegressor	2,717	2,897	2,292	2,441	0,153	0,012	0,08
BaggingRegressor	2,745	2,915	2,311	2,441	0,137	0,008	4,38
XGBRegressor	1,429	2,984	1,126	2,484	0,765	-0,028	0,96
LGBMRegressor	2,891	2,92	2,982	2,438	0,004	-0,007	0,02
CatBoostRegreesor	2,394	3,047	1,961	2,509	0,315	-0,078	0,04
StackingCVRegressor	2,956	2,929	2,495	2,29	-0,054	-0,016	3,15

Результат подекуди мінімально покращився, іноді – навпаки погіршився. Найкращою моделлю залишається ExtraTreesRegressor, що пояснюється у пункті 3.4.1.

Даний експеримент доводить, що помилок при використанні метрик не було: результат зумовлений значним відхиленням прогнозованих значень від фактичних, а не мінімальними похибками, які можна виправити округленням.

3.4.3 Задача класифікації

З огляду на відсутність або нерепрезентативну кількість певних оцінок (див. 1.3), корисно розглянути предмет дослідження як задачу класифікації і прогнозувати належність зразка до певного класу-оцінки з переліку унікальних значень за кожною цільовою змінною. Кількісно незначущі класи при цьому вилучаються для уникнення помилок при навчанні.

Для кожного з розглянутих методів існує як регресор, так і класифікатор. Оскільки задача розв’язується оглядово, для них можна буде використовувати підібрані гіперпараметри (див. 3.1) і оцінювати результат лише за метрикою точності. Якість класифікатора від цього не погіршиться, адже було налаштовано гіперпараметри для розбиттів, суттєві для здійснення класифікації. Результати дослідження наводяться у таблиці 3.10.

Таблиця 3.10 – Результати класифікації для побудованих моделей

Модель	Точність		Час навчання, с
	train	test	
SVC	0,436	0,254	0,04
ExtraTreesClassifier	0,349	0,245	0,12
BaggingClassifier	0,337	0,25	1,27

Продовження таблиці 3.10

XGBClassifier	0,967	0,217	2,2
LGBMClassifier	0,283	0,253	0,17
CatBoostClassifier	0,601	0,22	0,12
StackingCVClassifier	0,264	0,257	16,02

Найкращою моделю є StackingClassifier з моделями градієнтного бустінгу (див. 3.2.4) у якості базових оцінювачів та ExtraTreesClassifier як метамоделлю. Це зумовлено високою спроможністю стекінгу до узагальнення та попередження перенавчання окремих його компонентів (наприклад, XGBClassifier є перенавченим, оскільки має значущу різницю якості на навчальній та тестовій вибірках).

Даний експеримент доводить, що помилки інтерпретації задачі не було, оскільки загальна якість класифікації також є низькою через велику кількість класів з нерівномірною кількістю екземплярів. Отже, єдиною причиною слабкої результативності залишається саме специфіка вихідних даних зі слабкою кореляцією та суб'єктивністю ознак та оцінок.

3.4.4 Альтернативні постановки задачі

Для даних, що мають різні способи використання, доцільно розглянути альтернативні підходи до постановки задачі, аби впевнитись у правильності початкової версії або навпаки покращити результат дослідження. У даному випадку варто спробувати не лише прогнозування оцінки, а й визначення загальної якості ментального здоров'я або причинно-наслідкових зв'язків.

3.4.4.1 Модифікована класифікація

З огляду на неконзистентність оцінок (див. 1.3) діапазон значень цільових змінних можна розбити на класи-рівні, що вказують на ступінь наявності ментальних розладів. Прогноз здійснюється для всіх наявних оцінок щодо належності зразка до певного рівня [33]:

1) 0-3 – «Низький»;
2) 4-7 – «Середній»;
3) 8-10 – «Високий».

Результати дослідження наводяться у таблиці 3.12.

Таблиця 3.12 – Результати модифікованої класифікації для побудованих моделей

Модель	Точність		Час навчання, с
	train	test	
SVC	0,686	0,473	0,03
ExtraTreesClassifier	0,609	0,498	0,11
BaggingClassifier	0,606	0,493	1,14
XGBClassifier	0,968	0,49	0,81
LGBMClassifier	0,535	0,507	0,06
CatBoostClassifier	0,684	0,471	0,07
StackingCVClassifier	0,614	0,5	6,95

Найкращою моделлю є LGBMClassifier, що зумовлено здатністю алгоритму підлаштовуватись до складних залежностей у даних та здатністю обробляти великі масиви даних з істотною часткою категоріальних ознак.

У даному експерименті вирівнялись показники метрик за всіма моделями, що свідчить про поліпшення їхньої здатності до узагальнення та знаходження закономірностей. Отриману точність можна вважати прийнятною для прогнозування на складних даних реального світу.

3.4.4.2 Пошук асоціативних правил

З огляду на слабку кореляцію, можна перевірити дані на загальну якість та знайти більш явні закономірності шляхом пошуку асоціативних правил. Часті набори доцільно шукати для повного обсягу вихідних даних, перетвореного на набір транзакцій за допомогою однорозрядного кодування. Оскільки цікаво дослідити лише суттєві чинники, оцінки за кожною цільовою змінною було розділено на два класи за рівнем допустимості стану [33]:

1) 0-4 – «Прийнятний рівень»;
2) 5-10 – «Суттєвий рівень».

Результати дослідження для чотирьох розладів у порядку важливості наведено у таблиці 3.13.

Таблиця 3.13 – Найвпливовіші набори умов виникнення ментальних розладів

Розлад	№	Набір чинників
Тривожність	1	«While working»: так, «Exploratory»: так, «Primary streaming service»: Spotify, «Music effects»: покращує
	2	«While working»: так, «Primary streaming service»: Spotify, «Frequency (Gospel)»: ніколи, «Music effects»: покращує
	3	«While working»: так, «Exploratory»: так, «Frequency (Gospel)»: ніколи, «Music effects»: покращує

Продовження таблиці 3.13

Розлад	№	Набір чинників
Депресія	1	«While working»: так, «Exploratory»: так, «Primary streaming service»: Spotify, «Music effects»: покращує
	2	«While working»: так, «Exploratory»: так, «Foreign languages»: так, «Composer»: ні)
	3	«While working»: так, «Exploratory»: так, «Foreign languages»: так, «Music effects»: покращує
Безсоння	1	«Instrumentalist»: ні, «Exploratory»: так, «Primary streaming service»: Spotify, «Music effects»: покращує
	2	«While working»: так, «Exploratory»: так, «Composer»: ні, «Instrumentalist»: ні
	3	«Instrumentalist»: ні, «Exploratory»: так, «Composer»: ні, «Music effects»: покращує
ОКР	1	«While working»: так, «Exploratory»: так, «Primary streaming service»: Spotify, «Music effects»: покращує
	2	«While working»: так, «Exploratory»: так, «Composer»: ні, «Instrumentalist»: ні
	3	While working»: так, «Composer»: ні, «Music effects»: покращує, «Instrumentalist»: ні

Рівень підтримки правил становив 10-21%, достовірності – 27-75% (найкраще – для тривожності, найгірше для ОКР), максимальна довжина за всіма змінними – 4, що є задовільним результатом з точки зору результативності алгоритму.

Утім, отримані набори не сягають бажаного рівня інформативності. Наприклад, активне дослідження нових жанрів, прослуховування музики під час роботи та її сприятливий вплив на ментальний стан за відсутності музичної компетенції (володіння інструментами, навичками написання) є необхідними,

проте не достатніми факторами наявності розладу. Більш вагомими з точки зору змісту виявилися б правила, що містять набори умов, пов’язані саме з жанрами та частотою їх прослуховування. Це явище вкотре підтверджує факт складності та проблематично низького рівня кореляції всередині даних.

3.5 Результативність найкращої моделі на власному наборі даних

Було створено власний варіант опитування щодо музичних вподобань [34], аналогічний до описаного у підрозділі 1.2. Питання про оцінки ментального стану були поставлені для можливості зіставлення прогнозованих значень з фактичними, проте були не обов’язковими до відповіді. Результати для 5 анонімних респондентів наведено у таблиці 3.14.

Таблиця 3.14 – Найвпливовіші набори, що спричиняють ОКР

ID	Тривожність		Депресія		Безсоння		ОКР	
	pred	true	pred	true	pred	true	pred	true
1	5,5	5	4,5	3	3	2	2	2
2	5,5	6	4	2	3	1	2,5	1
3	5,5	4	4,5	3	3,5	4	2,5	0
4	5	3	3,5	4	3	2	0	1
5	5,5	7	4,5	5	3	0	2,5	5

Отримані оцінки з мінімальною похибкою репрезентують реальний ментальний стан респондентів. Ознаки «Безсоння» та «ОКР» наочно демонструють, що алгоритм не дає усереднені значення, як це можна було б підсумувати за першими двома, а знаходить справжню залежність.

Дієвий результат на фоні значень метрик якості за даною моделлю можна пояснити якістю оцінок, проставлених респондентами, що на практиці дає більш сильні залежності.

3.6 Висновки до розділу 3

У даному розділі було здійснено прогнози ментального стану для різних задач та даних на основі побудованих моделей.

Спочатку було описано процес попередньої обробки даних. Далі було визначено значущі для обраної задачі та набору даних гіперпараметри моделей і підібрано їхні найкращі конфігурації. Наступним кроком було навчено моделі регресії та здійснено прогноз на обраних даних. Найбільш результативною моделлю виявився бегінг ансамблів випадкового лісу, на основі якої було реалізовано можливість здійснювати прогноз для власноруч зібраних даних, який виявився практично прийнятним та послідовним.

З огляду на низьку результативність метрик було визначено потенційні помилки у ході розв’язання поставленої задачі, на основі оптимізаційних експериментів: застосування перехресної перевірки, округлення результатів, класифікації – та альтернативних задач: загальної класифікації рівнів стану, пошуку асоціативних правил – було зроблено висновок щодо складності залежностей всередині даних та суттєвого відсотку їхньої випадковості, що зумовлено суб’єктивністю їхньої тематики.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ ПРОГНОЗУВАННЯ ОЦІНОК МЕНТАЛЬНОГО ЗДОРОВ'Я

Метою даного розділу є огляд альтернативних варіантів реалізації програмного продукту (ПП) для прогнозування психічного здоров'я на основі даних про музичні уподобання та подальший вибір оптимального щодо визначених характеристик, вимог та економічних факторів із застосуванням апарату функціонально-вартісного аналізу (ФВА).

ФВА – це технологія, що дозволяє оцінити реальну вартість продукту незалежно від організаційної структури сторони-розробника, виявити резерви для зменшення витрат за рахунок оптимізації варіантів виробництва і співвідношення між споживчою вартістю виробу та витратами на його виготовлення.

Алгоритм ФВА складається з визначення послідовності етапів розробки продукту, повних витрат (річних), їхніх джерел та кількості робочих часів, кінцевого розрахунку вартості ПП.

4.1 Постановка задачі проектування

Рішення стосовно проектування та реалізації компонентів системи прогнозу психічного здоров'я на основі даних про музичні вподобання, що розробляється у даній роботі, мають задовольняються кожною її підсистемою, тому фактичний аналіз проводиться як аналіз функцій ПП.

Цільовий ПП має наступні технічні вимоги:

1) функціонування на персональних комп'ютерах зі стандартною конфігурацією;

2) швидкість обробки даних та отримання прогнозів;
3) зручний доступ до результатів дослідження;
4) легка інтеграція з тематично спорідненими програмними продуктами на кшталт стрімінгових платформ, рекомендаційних систем;
5) мінімальні витрати на розробку та впровадження.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка алгоритмів прогнозування стану психічного здоров'я на основі метаданих, що свідчать про музичні уподобання особи. На її основі можна визначити наступні основні функції:

1) F_1 – вибір мови програмування;
2) F_2 – вибір класу методів машинного навчання;
3) F_3 – вибір середовища розробки.

Кожна з вищезазначених функцій має декілька варіантів реалізації:

1) Функція F_1:
    а) Python:
    б) R.
2) Функція F_2:
    а) метод опорних векторів;
    б) ансамблеві методи.
3) Функція F_3:
    а) PyCharm;
    б) Google Colaboratory.

Для спрощення аналізу доцільно будувати морфологічну карти системи, що відображає множину всіх можливих варіантів їхньої реалізації (рис. 4.1).

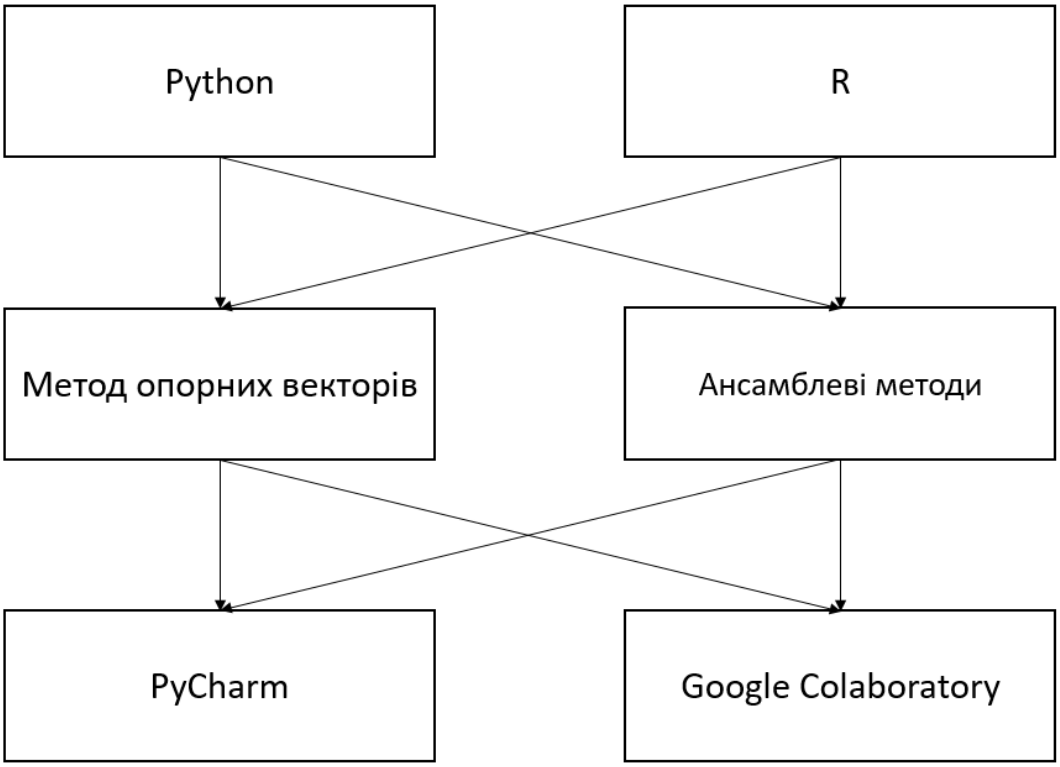


Рисунок 4.1 – Морфологічна карта

Позитивно-негативна матриця системи представлена у таблиці 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Простота синтаксису, висока продуктивність, кросплатформеність, широкий вибір бібліотек та фреймворків	Повільність виконання, використання значного об'єму пам'яті
	Б	Наявність потужних засобів для статистичного аналізу та візуалізації даних	Відсутність універсальності, повільність виконання, використання значного об'єму пам'яті

Продовження таблиці 4.1

Функції	Варіанти реалізації	Переваги	Недоліки
F_2	*А*	Висока точність прогнозування, ефективність при роботі з розрідженими даними, даними великих обсягів	Вразливість до зашумлених даних, складність налаштування гіперпараметрів під специфіку даних
	Б	Висока точність прогнозування, сумісність з різними типами даних, відсутність схильності до перенавчання	Складність архітектури, високі вимоги до обчислювальної потужності ПК
F_3	*А*	Розширений функціонал, використання повного потенціалу потужності ПК	Локальне середовище, високі вимоги до обчислювальної потужності ПК та об'єму пам'яті
	Б	Хмарне середовище, інтеграція із сервісами Google, зручне подання візуалізацій та результатів	Залежність від підключення до мережі, платне підсилення потужності

З позитивно-негативної матриці видно, що переваги деяких варіантів не є глобально вагомими, або перекриваються суттєвими недоліками, або абсолютно переважають альтернативні. Враховуючи цей факт, можна прийняти рішення про відкидання тих альтернатив, що не відповідають вимогам до ПП.

Оновлені варіанти реалізації для основних функцій будуть наступними:

1. F_1: доцільно надати перевагу простоті та загальнодоступності у вигляді варіанту А.
2. F_2: обидва варіанти (А та Б) є прийнятними, оскільки є повноцінними альтернативами одне одного згідно з перевагами та недоліками.
3. F_3: доцільно надати перевагу хмарному середовищу з високим ступенем інтегрованості та візуальним потенціалом у вигляді варіанту Б.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

1. $F_1А$ – $F_2А$ – $F_3Б$;
2. $F_1А$ – $F_2Б$– $F_3Б$.

4.3 Обґрунтування системи параметрів програмного продукту

На основі вищерозглянутих варіантів реалізації визначаються основні параметри вибору, що суттєво впливають на якість ПП, а отже будуть використані у процесі розрахунку коефіцієнта технічного рівня. Для даного ПП вони будуть наступними:

1) X_1 – швидкодія мови програмування (вимірюється у кількості операцій на мілісекунду);
2) X_2 – об’єм оперативної пам’яті (вимірюється у гігабайтах);
3) X_3 – час повної попередньої обробки даних (за даного рівня складності найдоцільніше вимірювати у секундах);
4) X_4 – час навчання моделі (за даного рівня складності найдоцільніше вимірювати у мілісекундах).

Гірші, середні та кращі значення параметрів визначаються експертно, спираючись на вимоги до розробки та експлуатації ПП, поточні характеристики обраних параметрів (табл. 4.2).

Таблиця 4.2 – Основні параметри ПП

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X_1	Оп/мс	10000	15000	20000
Об’єм оперативної пам’яті	X_2	ГБ	4	16	32
Час попередньої обробки даних	X_3	с	20	10	5
Час навчання моделі	X_4	мс	1500	800	200

За даними таблиці 4.3 будуються графічні характеристики параметрів (рис. 4.2 – рис. 4.5).

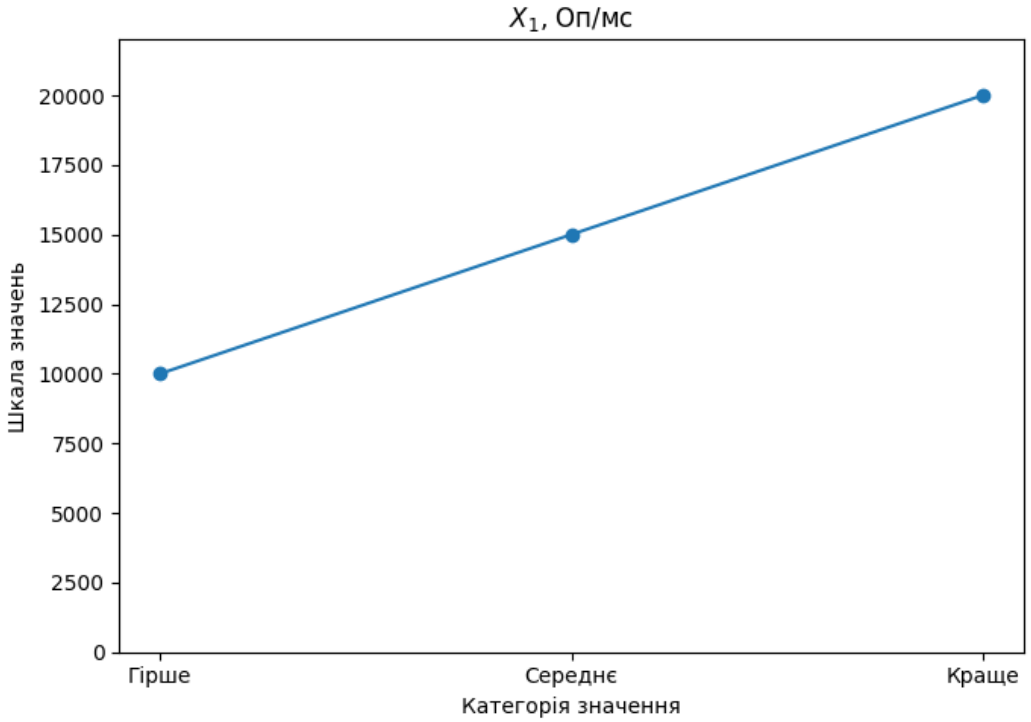


Рисунок 4.2 – X_1, швидкодія мови програмування

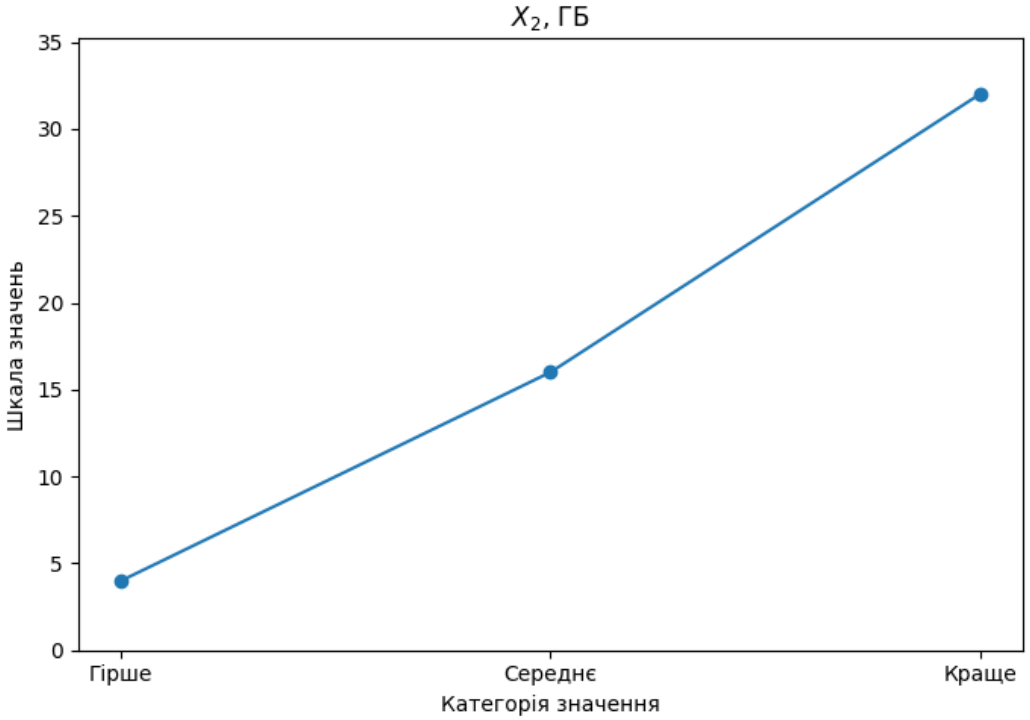


Рисунок 4.3 – X_2, об’єм оперативної пам’яті

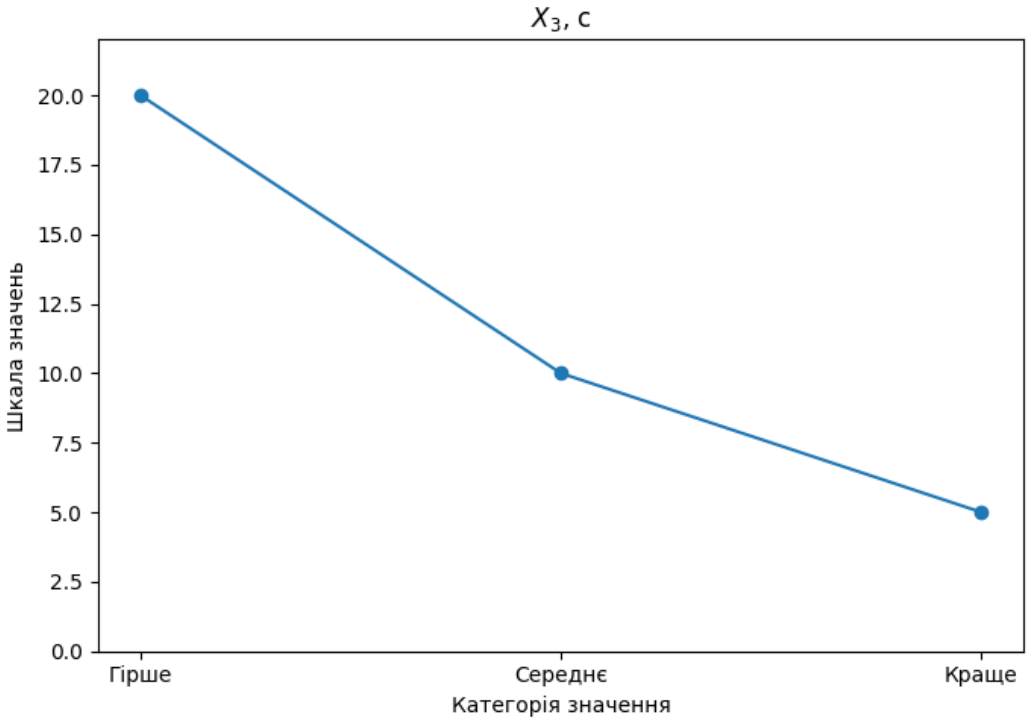


Рисунок 4.4 – X_3, час попередньої обробки даних

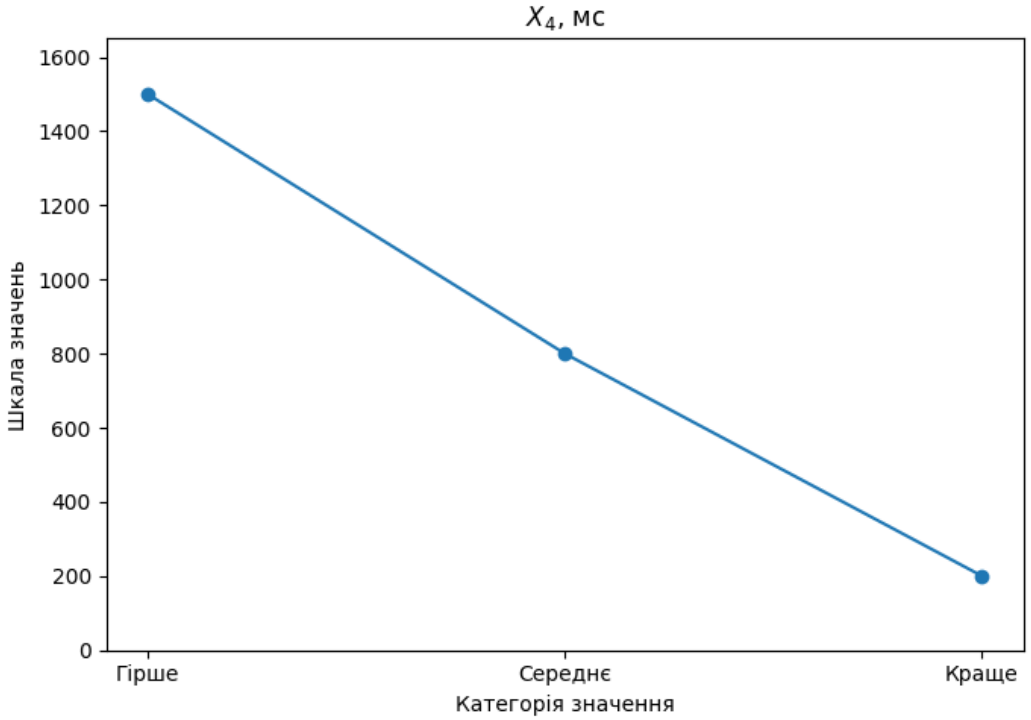


Рисунок 4.5 – X_4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

На підставах спільного аналізу група з 7 експертів незалежно одне від одного оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Визначення коефіцієнтів значимості передбачає:

1) визначення рівня значимості параметра шляхом присвоєння різних рангів;
2) перевірку придатності експертних оцінок для подальшого використання;
3) визначення оцінки попарного пріоритету параметрів;

4) обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів Ri	Відхи-лення Δi	Δi2
			1	2	3	4	5	6	7			
X_1	Швидкодія мови програмування	Оп/мс	1	1	2	1	1	2	1	9	-8,5	72,25
X_2	Об’єм оперативної пам’яті	ГБ	3	2	1	2	2	1	3	14	-3,5	12,25
X_3	Час попередньої обробки даних	с	4	4	4	4	4	3	4	27	9,5	90,25
X_4	Час навчання моделі	мс	2	3	3	3	3	4	2	20	2,5	6,25
	Разом		10	10	10	10	10	10	10	70	0	181

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів (табл. 4.3):

$$R_i = \sum_{j=1}^{N} r_{ij}\,, \qquad (4.1)$$

$$R_{ij} = \frac{Nn(n+1)}{2}, \tag{4.2}$$

де N – число експертів;

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij}, \tag{4.3}$$

$$T = \frac{1}{7} * 70 = 17{,}5$$

в) відхилення суми рангів кожного параметра від середньої суми рангів, сума яких повинна дорівнювати 0 (табл. 4.3):

$$\Delta_i = R_i - T. \tag{4.4}$$

г) загальна сума квадратів відхилення (табл. 4.3):

$$S = \sum_{i=1}^{N} \Delta_i^2, \tag{4.5}$$

Обчислимо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)}, \tag{4.6}$$

$$W = \frac{12 \cdot 181}{7^2(4^3 - 4)} = 0{,}739 > W_k = 0{,}67.$$

Експертне ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, що дорівнює 0,67.

На основі результатів ранжування, проводиться попарне порівняння всіх параметрів, результати якого подані у таблиці 4.4.

Таблиця 4.4 – Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X_1 і X_2	<	<	>	<	<	>	<	<	0,5
X_1 і X_3	<	<	<	<	<	<	<	<	0,5
X_1 і X_4	<	<	<	<	<	<	<	<	0,5
X_2 і X_3	<	<	<	<	<	<	<	<	0,5
X_2 і X_4	>	<	<	<	<	<	>	<	0,5
X_3 і X_4	>	>	>	>	>	<	>	>	1,5

Числове значення, що визначає ступінь переваги і–го параметра над j–тим, аіj визначається за формулою (таблиця 4.4):

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.7)$$

Отримані числові оцінки є компонентами матриці А=‖ аіj ‖ (табл. 4.5).

На першому етапі для кожного параметра проводиться розрахунок вагомості Кві за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.8)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \tag{4.9}$$

Відносні оцінки розраховуються у декілька етапів, поки наступні значення не будуть відрізнятися від попередніх менше, ніж на 2%. На наступних ітераціях відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.10}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \tag{4.11}$$

Згідно з таблицею 4.5, різниця між значеннями коефіцієнтів вагомості першої та другої ітерацій не перевищує 2%, тому отримувати більш точне значення на наступних ітераціях недоцільно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітерація		Друга ітерація	
	X_1	X_2	X_3	X_4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$
X_1	1	0,5	0,5	0,5	2,5	0,156	9,25	0,157
X_2	1,5	1	0,5	0,5	3,5	0,219	12,25	0,208
X_3	1,5	1,5	1	1,5	5,5	0,344	21,25	0,36
X_4	1,5	1,5	0,5	1	4,5	0,281	16,25	0,275
Всього:					16	1	59	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Рівень якості кожного варіанту виконання визначається для кожної з основних функцій окремо.

Абсолютні значення параметрів X_2 (об'єм оперативної пам'яті), X_3 (час попередньої обробки даних) та X_4 (час навчання моделі) відповідають технічним вимогам умов функціонування даного ПП. Абсолютне значення параметра X_1 (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП (таблиця 4.6) розраховується за наступною формулою:

$$K_K(j) = \sum_{i=1}^{n} K_{\text{ві},j} B_{i,j}\,, \tag{4.12}$$

де n – кількість параметрів;

$K_{\text{ві}}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F_1	*А*	X_1	14000	5	0,157	0,785
F_2	*А*	X_2	16	7	0,208	1,456
	Б	X_3	10	7	0,36	2,52
F_3	*А*	X_4	320	8	0,275	2,2

За даними з таблиці 4.6 визначаємо рівень якості кожного з варіантів за наступною формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \quad (4.13)$$

$$K_{K1} = 0{,}785 + 1.456 + 2{,}2 = 4{,}441,$$

$$K_{K2} = 0{,}785 + 2{,}52 + 2{,}2 = 5{,}505.$$

З розрахунків випливає, що кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП необхідно спочатку провести розрахунок трудомісткості.

Всі варіанти складаються з двох окремих завдань:

1. Проектування ПП;
2. Розробка ПП.

Завдання 1 за ступенем новизни відноситься до групи А (задачі, які передбачають використання принципово нових методів розробки, проведення науково-дослідних робіт), завдання 2 – до групи Б (типові проектні рішення, оригінальні задачі і системи, які не мають аналогів). За складністю алгоритмизавдання 1 належать до групи 1 (алгоритми оптимізації та моделювання систем і об’єктів), а в завданні 2 – до групи 3 (алгоритми, які реалізують стандартні методи рішень, а також не передбачають використання складних чисельних і логічних методів).

Завдання 1 використовує довідкову інформацію та експертний досвід, завдання 2 – інформацію у вигляді даних. Розрахунок норм часу на розробку та програмування проводиться для кожного завдання.

Загальна трудомісткість обчислюється як:

$$\mathrm{T_O} = \mathrm{T_P} \cdot \mathrm{K_П} \cdot \mathrm{K_{СК}} \cdot \mathrm{K_М} \cdot \mathrm{K_{СТ}} \cdot \mathrm{K_{СТ.М}}, \quad (4.14)$$

де $\mathrm{T_P}$ – трудомісткість розробки ПП;

$\mathrm{K_П}$ – поправочний коефіцієнт;

$\mathrm{K_{СК}}$ – коефіцієнт на складність вихідної інформації;

$\mathrm{K_М}$ – коефіцієнт рівня мови програмування;

$\mathrm{K_{СТ}}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$\mathrm{K_{СТ.М}}$ – коефіцієнт стандартного математичного забезпечення.

У всіх завданнях використовується мова програмування високого рівня, тому коефіцієнт КМ = КСТ.П = 1, також вважаємо, що за рівня складності контролю вихідної інформації (змінного типу) 5,2, а вихідної – 5,4 КСК = 1.

Відповідно до норм часу для завдань розрахункового характеру, для першого маємо ТР = 90 людино-днів, КП = 1,7, КСТ = 0,7; для другого – ТР = ТР = 27 людино-днів, КП = 0,9, КСТ = 0,7. Якщо не враховувати коефіцієнти, що дорівнюють 1, то загальна трудомісткість програмування для кожного з завдань дорівнює:

$$T_1 = 90 \cdot 1{,}7 \cdot 0{,}7 = 107{,}1 \text{ людино} - \text{днів},$$

$$T_2 = 27 \cdot 0{,}9 \cdot 0{,}7 = 17{,}01 \text{ людино} - \text{днів}.$$

Далі складається трудомісткість для кожного з обраних варіантів реалізації програми, щоб отримати загальне значення за двома завданнями:

$$\mathrm{T_I} = (107{,}1\ + 17{,}01 + 4{,}8 + 17{,}01) \cdot 8 = 1167{,}36\ \text{людино} - \text{годин},$$

$$\mathrm{T_{II}} = (107{,}1\ + 17{,}01 + 6{,}91 + 17{,}01) \cdot 8 = 1184{,}24\ \text{людино} - \text{годин}.$$

Отже, найбільш високу трудомісткість має варіант II.

В розробці беруть участь два дата-аналітики з окладом 24000 грн. та один програміст з окладом 20000. Середню зарплату співродітників за годину за формулою:

$$\mathrm{C_Ч} = \frac{\mathrm{M}}{T_m \cdot t}\ \text{грн.}, \tag{4.15}$$

де M – місячний оклад працівників;

T_m – кількість робочих днів на місяць;

t – кількість робочих годин на день.

$$\mathrm{C_Ч} = \frac{2 \cdot 24000 + 20000}{3 \cdot 21 \cdot 8} = 134{,}92\ \text{грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$\mathrm{C_{ЗП}} = \mathrm{C_Ч} \cdot \mathrm{T}_i \cdot \mathrm{КД}, \tag{4.16}$$

де $\mathrm{C_Ч}$ – величина погодинної оплати праці працівника;

T_i – трудомісткість відповідного завдання;

$\mathrm{К_Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I.}\ \mathrm{C_{ЗП}} = 134{,}92 \cdot 1167{,}36 \cdot 1{,}2 = 189000{,}25\ \text{грн.}$$

$$\text{II. } С_{ЗП} = 134{,}92 \cdot 1184{,}24 \cdot 1{,}2 = 191733{,}19 \text{ грн.}$$

Відрахування на єдиний соціальний внесок (ЄСВ), що становить 22%, обчислюється наступним чином:

$$\text{I. } С_{ВІД} = С_{ЗП} \cdot 0.22 = 189000{,}25 \cdot 0.22 = 41580{,}06 \text{ грн.}$$

$$\text{II. } С_{ВІД} = С_{ЗП} \cdot 0.22 = 191733{,}19 \cdot 0.22 = 42181{,}26 \text{ грн.}$$

Далі необхідно визначити витрати на оплату однієї машино-години (СМ).

Одна ЕОМ обслуговує одного програміста з окладом 20000 грн. з коефіцієнтом зайнятості 0,2, тому $С_Г$ обчислюється наступним чином:

$$С_Г = 12 \cdot M \cdot К_З = 12 \cdot 20000 \cdot 0{,}2 = 48000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 48000 \cdot (1 + 0.2) = 57600 \text{ грн.}$$

Відрахування на ЄСВ становить:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 57600 \cdot 0{,}22 = 12672{,}2 \text{ грн.}$$

Амортизаційні відрахування розраховуються для амортизації 30% та вартості ЕОМ 35000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1{,}15 \cdot 0{,}3 \cdot 35000 = 12075 \text{ грн.},$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$ – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1{,}15 \cdot 35000 \cdot 0{,}05 = 2012{,}5 \text{ грн.},$$

де $К_Р$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t \cdot К_В = (365 - 104 - 12 - 16) \cdot 8 \cdot 0{,}9 = \\ = 1677{,}6 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день, $К_В$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуються за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1677{,}6 \cdot 0{,}3 \cdot 0{,}8 \cdot 4{,}86 = 1956{,}75 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$ – коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуються за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 35000 \cdot 0{,}67 = 23450 \text{ грн.}$$

Тоді, річні експлуатаційні витрати визначаються як:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_{А} + С_{Р} + С_{ЕЛ} + С_{Н}, \quad (4.17)$$

$$С_{ЕКС} = 57600 + 12672{,}2 + 12075 + 2012{,}5 + 1956{,}75 + 23450 = 109766{,}45 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М-Г} = \frac{С_{ЕКС}}{Т_{ЕФ}} = \frac{109766{,}45}{1677{,}6} = 65{,}43 \text{ грн/год.}$$

Оскільки всі роботи, пов‘язані з розробкою ПП, проводяться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складатимуть:

$$С_{М} = С_{М-Г} \cdot T, \quad (4.18)$$

$$\text{I. } С_{М} = 65{,}43 \cdot 1167{,}36 = 76380{,}37 \text{ грн.}$$

$$\text{II. } С_{М} = 65{,}43 \cdot 1184{,}24 = 77484{,}82 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$С_{Н} = С_{ЗП} \cdot 0{,}67, \quad (4.19)$$

$$\text{I. } С_{Н} = 189000{,}25 \cdot 0{,}67 = 126630{,}17 \text{ грн.}$$

$$\text{II. } С_{Н} = 191733{,}19 \cdot 0{,}67 = 128461{,}24 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_{М} + C_{Н}, \quad (4.20)$$

I. $C_{ПП} = 189000{,}25 + 41580{,}06 + 76380{,}37 + 126630{,}17 = 433590{,}85$ грн.

II. $C_{ПП} = 191733{,}19 + 42181{,}26 + 77484{,}82 + 128461{,}24 = 439860{,}51$ грн.

4.7 Вибір кращого варіанту програмного продукту техніко економічного рівня

Коефіцієнт техніко-економічного рівня розраховується за формулою:

$$К_{ТЕР j} = \frac{К_{К j}}{С_{Ф j}}, \quad (4.21)$$

$$К_{ТЕР1} = \frac{4{,}441}{433590{,}85} = 1{,}024 \cdot 10^{-5},$$

$$К_{ТЕР2} = \frac{5{,}505}{439860{,}51} = 1{,}252 \cdot 10^{-5}.$$

Отже, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР2} = 1{,}252 \cdot 10^{-5}$.

За найкращий варіант реалізації програмного продукту має такі параметри:

1. Вибір мови програмування: Python;
2. Вибір класу методів машинного навчання: ансамблеві методи;

3. Вибір середовища розробки: Google Colaboratory.

Обраний варіант виконання програмного комплексу дозволяє розробникам зменшити вимоги до обчислювальної потужності та об’єму пам’яті, а користувачам – зручно та швидко отримувати дані у формі гарно інтерпретованих візуалізацій.

4.8 Висновки до розділу 4

У даному розділі було проведено повний функціонально-вартісний аналіз програмного продукту та визначено оцінку його основних функцій.

Спочатку було визначено вимоги до програмного продукту та основні функції, що його характеризують. Наступним кроком було обрано варіанти реалізації процесу розробки за кожною з основних функцій, з яких на основі аналізу позитивно-негативної матриці залишили до розгляду лише доцільні. Далі було визначено параметри, що характеризують програмний продукт, за результатами експертного оцінювання було проведено аналіз їхньої якості та застосовано методи економічного аналізу для вибору найкращого як для розробників, так і користувачів програмного продукту.

ВИСНОВКИ

У даній дипломній роботі було створено програмний продукт, що прогнозує стан психічного здоров’я особи на основі даних про її музичні вподобання: загальних інтересів, сформованої культури прослуховування, частоти звернення до різних жанрів тощо. Результатом прогнозу є оцінка від 0 до 10 ступеня наявності в особи тривожності, безсоння, депресії та обсесивно-компульсивного розладу.

У першому розділі роботи було поставлено задачу та розглянуто найпоширеніші методи її розв’язання, описано та проаналізовано вихідні дані. У другому розділі було наведено сутність та математичний опис використаних у роботі методів машинного навчання і супутніх алгоритмів, зазначено їхні переваги та недоліки. У третьому розділі було побудовано та навчено альтернативні моделі з попереднім підбором гіперпараметрів, проведено оптимізаційні та альтернативні експерименти, що спростували факт наявності помилок у розв’язанні задачі. У четвертому розділі шляхом функціонально-вартісного аналізу було визначено найкращий з запропонованих варіант реалізації програмного продукту на основі його основних вимог та характеристик.

Найкращою моделлю є бегінг ансамблів на основі особливо випадкового лісу. Це пояснюється випадковістю розбиттів та незалежним навчанням дерев з подальшою агрегацією прогнозу, що попереджує перенавчання і допомагає знаходити складні нелінійні залежності. За даною моделлю було успішно прогнозовано оцінки ментального здоров’я на власноруч створеній вибірці.

З огляду на експериментально обґрунтований факт складної залежності всередині даних та суб’єктивності визначення фактичних значень цільових змінних, задачу можна вважати успішно розв’язаною, а розроблений алгоритм – практично прийнятним з точки зору результативності. Його недоліком є

відсутність бажаного рівня точності, яку можна покращити шляхом застосування нейронних мереж. Цей підхід не був розглянутий у роботі з огляду на архітектурну та обчислювальну складність, масштаб такого предмету дослідження, а відсутність набору даних достатнього обсягу.

Розроблений прогнозатор є прикладним алгоритмом, який можна використовувати у науково-дослідницьких цілях та додавати до різних програмних продуктів. Найбільш простим та доцільним способом застосування є інтеграція у стрімінгові платформи для попередження користувачів про можливу наявність у них психічних розладів та налагодження для такої особи спеціальної рекомендаційної системи, що мінімізує доступ до провокативного, вразливого контенту, натомість пропонуючи музичні матеріали, що сприяють покращенню ментального здоров’я.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Music & mental health survey results. *Kaggle*. URL: https://www.kaggle.com/datasets/catherinerasgaitis/mxmh-survey-results.
2. What is a target variable in machine learning?. *H2O.ai*. URL: https://h2o.ai/wiki/target-variable/#:~:text=The%20target%20variable%20is%20the%20variable%20whose%20values%20are%20modeled,value%20of%20the%20target%20variable.
3. Boxplot for anomaly detection. *Towards Data Science*. URL: https://towardsdatascience.com/boxplot-for-anomaly-detection-9eac783382fd.
4. What should be the allowed percentage of missing data?. *Analytics Vidhya*. URL: https://discuss.analyticsvidhya.com/t/what-should-be-the-allowed-percentage-of-missing-values/2456.
5. How to read a correlation matrix. *Statology*. URL: https://www.statology.org/how-to-read-a-correlation-matrix/.
6. Regression Models with multiple target variables. *Towards Data Science*. URL: https://towardsdatascience.com/regression-models-with-multiple-target-variables-8baa75aacd.
7. Regression vs. classification in machine learning. *JavaTPoint*. URL: https://www.javatpoint.com/regression-vs-classification-in-machine-learning#:~:text=The%20main%20difference%20between%20Regression,Spam%20or%20Not%20Spam,%20etc.
8. Step by step guide to regression analysis. *Becoming human: artificial intelligence magazine*. 2021. P. 1–5.

9. Decision trees: understanding the basis of ensemble methods. *Towards Data Science*. URL: https://towardsdatascience.com/decision-trees-understanding-the-basis-of-ensemble-methods-e075d5bfa704.
10. Illustration of a decision tree. *ResearchGate*. URL: https://www.researchgate.net/figure/Illustration-of-a-decision-tree_fig3_338780422.
11. 10 decision trees are better than 1. *Towards Data Science*. URL: https://towardsdatascience.com/10-decision-trees-are-better-than-1-719406680564.
12. Okun O., Valentini G. Applications of supervised and unsupervised ensemble methods. *Workshop on supervised and unsupervised ensemble methods and their applications*: European Conference on Artificial Intelligence, Patrai, 21 June 2008. Berlin, 2009. P. 4–20.
13. Random forest. *Roshan Talimi*. URL: http://talimi.se/ml/random-forest/.
14. Bagging vs boosting machine learning methods. *Data Analytics*. URL: https://vitalflux.com/bagging-vs-boosting-machine-learning-methods/.
15. Stacking ensemble learning for short-term electricity consumption forecasting. *Energies*. 2018. P. 6.
16. Support vector regression tutorial for machine learning. *Analytics Vidhya*. URL: https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/#:~:text=Support%20Vector%20Regression%20(SVR)%20is,while%20minimizing%20the%20prediction%20error.
17. The principle of support vector regression (SVR). *ResearchGate*. URL: https://www.researchgate.net/figure/The-principle-of-support-vector-regression-SVR_fig1_341779056.
18. User guide. *scikit-learn*. URL: https://scikit-learn.org/stable/user_guide.html.
19. One-Hot Encoding vs. Label Encoding using Scikit-Learn. *Analytics Vidhya*. URL: https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/.

20. Feature scaling techniques in python – A complete guide. *Towards Data Science*. URL: https://www.analyticsvidhya.com/blog/2021/05/feature-scaling-techniques-in-python-a-complete-guide/.
21. Feature selection techniques in machine learning. *Analytics Vidhya*. URL: https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/.
22. Kyriakides G., Margaritis K. G. Hands-on ensemble learning with python: build highly optimized ensemble machine learning models using scikit-learn and Keras. Birmingham: Packt Publishing, 2019. 273 p.
23. Zhou, Z. Ensemble methods: foundations and algorithms. Boca Raton, FL: CRC Press, 2012. 222 p.
24. CatBoost open source library documentation. *CatBoost*. URL: https://catboost.ai/en/docs/.
25. Інтелектуальний аналіз даних. Практикум: навчальний посібник для студентів, які навчаються за спеціальністю 124 «Системний аналіз», освітніх програм «Системний аналіз і управління», «Системний аналіз фінансового ринку». Київ : КПІ ім. Ігоря Сікор., 2021. 106 с. URL: https://ela.kpi.ua/handle/123456789/53763.
26. 3 best metrics to evaluate regression model?. *Towards Data Science*. URL: https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b.
27. Train/Test split and cross validation in python. Towards Data Science. URL: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6.
28. Hyperparameters tuning for machine learning model ensembles. *Towards Data Science*. URL: https://towardsdatascience.com/hyperparameters-tuning-for-machine-learning-model-ensembles-8051782b538b.
29. How to choose data preparation methods for machine learning. *Machine Learning Mastery*. URL: https://machinelearningmastery.com/choose-data-preparation-methods-for-machine-learning/.

30. Filling in the gaps: imputation 3 ways. *Towards Data Science*. URL: https://towardsdatascience.com/filling-in-the-gaps-imputation-3-ways-6056c09b6417.
31. 5 reasons why you should use cross-validation in your data science projects. *Towards Data Science*. URL: https://towardsdatascience.com/5-reasons-why-you-should-use-cross-validation-in-your-data-science-project-8163311a1e79.
32. Marland E., Bosse M. J. Rounding the regression. *PRIMUS: problems, resources, and issues in mathematics undergraduate studies*. 2017. 23 p.
33. Dividing a continuous variable into categories. *COMMON MISTAKES IN USING STATISTICS: Spotting and Avoiding Them*. URL: https://web.ma.utexas.edu/users/mks/statmistakes/dividingcontinuousintocategories.html.
34. Власний варіант опитування щодо музичних вподобань. URL: https://forms.gle/MhcZrpoDnu8NoGcm9.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
# -*- coding: utf-8 -*-
"""diploma_lanko.ipynb

Automatically generated by Colaboratory.

Original file is located at
      https://colab.research.google.com/drive/1dhEV1DZHZdM7Ip_Wue7Mi9QMyldw0VIo#scrollTo=oMGFgGw60Yg8
Original data is located at
      https://drive.google.com/drive/folders/1Q1rF0y3X3ShdjHNSKHKpurNJK060cphC?usp=drive_link

# Бібліотеки, модулі
"""

from google.colab import drive
drive.mount("/content/drive")

!pip install catboost

import pickle
import time
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import RFECV
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler
from sklearn.svm import SVR, SVC
from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor,
ExtraTreesClassifier, BaggingRegressor, BaggingClassifier, StackingRegressor,
StackingClassifier
from xgboost import XGBRegressor, XGBClassifier
from lightgbm import LGBMRegressor, LGBMClassifier
from catboost import CatBoostRegressor, CatBoostClassifier
from sklearn.model_selection import train_test_split, cross_validate,
GridSearchCV
from sklearn.metrics import make_scorer, mean_squared_error,
mean_absolute_error, r2_score, accuracy_score
from mlxtend.frequent_patterns import apriori, association_rules

"""# Функції"""

def df_stats(df):
    print("Загальна інформація про дані:")
    print(df.info())
    print("\nКількість унікальних значень за всіма ознаками:")
    print(df.nunique())
```

```
    print("\nРозрахунок основних статистичних показників за числовими
ознаками:")
    print(df.describe())

def plot_nan_percentage(df):
    percentage = np.around(df.isna().mean() * 100, decimals=2)
    importance = [{p < 15: "olivedrab", p >= 15: "indianred"}[True] for p in
percentage]
    plt.figure(figsize=(10, 6))
    plt.bar(df.columns, height=list(percentage), color=importance)
    plt.axhline(y=15, color='black', linestyle='--', label='поріг у 15%')
    plt.xticks(rotation=90)
    plt.legend()
    plt.title("Відсоток пропущених даних")
    plt.show()

def fill_nan(df):
    for col in df.columns:
        if df[col].dtype == 'object':
            df[col] = df[col].fillna(df[col].mode().iloc[0])
        elif df[col].dtype == 'float64':
            df[col] = df[col].fillna(df[col].median())
    if not df.isna().any().any():
        print("NaN для числових ознак успішно заповнені медіаною, для
категоріальних - модою")

def anomaly_check(df, mode='init'):
    float_cols = df.loc[:, "Age":"Frequency (Video game
music)"].select_dtypes(include=['float64'])
    fig, axs = plt.subplots(1, 3, figsize=(12, 4))
    for i, column in enumerate(float_cols.columns):
        axs[i].boxplot(float_cols[column])
        axs[i].set_title(column)
    plt.tight_layout()
    plt.subplots_adjust(top=0.85)
    if mode == 'init':
        plt.suptitle("Коробкві графіки для числових ознак до видалення
аномалій")
    elif mode == 'check':
        plt.suptitle("Коробкві графіки для числових ознак після видалення
аномалій")
    plt.show()

def get_X_y(df):
    drop =
df.columns[df.columns.get_loc('Anxiety'):df.columns.get_loc('OCD')+1]
    X = df.drop(columns=drop)
    y = df.loc[:, "Anxiety":"OCD"]
    return X, y

def encode(arr, mode='obj'):
    arr_enc = arr.copy()
    le = LabelEncoder()
    if mode == 'obj':
        for col in arr.select_dtypes(include=['object']):
```

```
            arr_enc[col] = le.fit_transform(arr[col])
    elif mode == 'all':
        for col in arr:
            arr_enc[col] = le.fit_transform(arr[col])
    return arr_enc

def scale(X, mode):
    X_scaled = X.copy()
    if mode == 'minmax':
        scaler = MinMaxScaler()
    elif mode == 'standard':
        scaler = StandardScaler()
    for col in X_scaled.select_dtypes(include=['float64']):
        X_scaled[col] = scaler.fit_transform(X_scaled[[col]])
    return X_scaled

def plot_distribution(X, mode='init'):
    fig, axs = plt.subplots(1, 3, figsize=(12, 4))
    for i, col in enumerate(X.select_dtypes(include=['float64']).columns):
        X[col].hist(bins=20, ax=axs[i])
        axs[i].set_title(col)
    plt.tight_layout()
    plt.subplots_adjust(top=0.85)
    if mode == 'init':
        plt.suptitle("Розподіл числових ознак до масштабування")
    elif mode == 'minmax':
        plt.suptitle("Розподіл числових ознак після нормалізації")
    elif mode == 'standard':
        plt.suptitle("Розподіл числових ознак після стандартизації")
    plt.show()

def plot_variables(y):
    fig, axs = plt.subplots(2, 2, figsize=(10, 6))
    axs[0, 0].bar(y["Anxiety"].value_counts().index,
y["Anxiety"].value_counts().values,
color=matplotlib.colormaps.get_cmap('tab20')(np.linspace(0, 1,
len(y["Anxiety"].value_counts()))))
    axs[0, 1].bar(y["Insomnia"].value_counts().index,
y["Insomnia"].value_counts().values,
color=matplotlib.colormaps.get_cmap('tab20')(np.linspace(0, 1,
len(y["Insomnia"].value_counts()))))
    axs[1, 0].bar(y["Depression"].value_counts().index,
y["Depression"].value_counts().values,
color=matplotlib.colormaps.get_cmap('tab20')(np.linspace(0, 1,
len(y["Depression"].value_counts()))))
    axs[1, 1].bar(y["OCD"].value_counts().index,
y["OCD"].value_counts().values,
color=matplotlib.colormaps.get_cmap('tab20')(np.linspace(0, 1,
len(y["OCD"].value_counts()))))
    axs[0, 0].set_title("Anxiety")
    axs[0, 1].set_title("Insomnia")
    axs[1, 0].set_title("Depression")
    axs[1, 1].set_title("OCD")
    fig.suptitle("Розподіл значень цільових змінних")
    plt.show()
```

```
def correlation(arr, mode='features'):
    plt.figure(figsize=(20, 8))
    sns.heatmap(arr.corr(), annot=True, vmin=-1, vmax=1, center=0)
    sns.heatmap(arr.corr(), annot=True, vmin=-1, vmax=1, center=0)
    if mode == 'features':
        plt.title("Матриця кореляції ознак")
    elif mode == 'variables':
        plt.title("Матриця кореляції цільових змінних")
    plt.tight_layout()
    plt.show()

def grid_search(X, y, param_grid, method):
    if method == 'svr':
        model = SVR()
    elif method == 'rf':
        model = RandomForestRegressor(random_state=42)
    elif method == 'et':
        model = ExtraTreesRegressor(random_state=42)
    elif method == 'bg':
        with
open(f'/content/drive/MyDrive/Diploma/Regression/best_model_et.pkl', 'rb') as
f:
            base_estimator = pickle.load(f)
        model = BaggingRegressor(estimator=base_estimator, random_state=42)
    elif method == 'xgb':
        model = XGBRegressor(random_state=42)
    elif method == 'lgbm':
        model = LGBMRegressor(random_state=42)
    elif method == 'cat':
        model = CatBoostRegressor(random_seed=42, verbose=False)
    grid_search = GridSearchCV(model, param_grid,
scoring='neg_mean_squared_error', cv=5, n_jobs=-1)
    grid_search.fit(X, y)
    best_params = grid_search.best_params_
    str_params = {k: str(v) if isinstance(v, float) else v for k, v in
best_params.items()}
    df = pd.DataFrame.from_dict(str_params, orient='index',
columns=['Найкраще значення'])
    df.columns.name = 'Параметр'
    df = df.to_string(justify='center')
    print(df)
    best_model = grid_search.best_estimator_
    with
open(f'/content/drive/MyDrive/Diploma/Regression/best_model_{method}.pkl',
'wb') as f:
        pickle.dump(best_model, f)

def select_features(X1, X2, y, mode, per=1):
    with open('/content/drive/MyDrive/Diploma/Regression/best_model_rf.pkl',
'rb') as f:
        model = pickle.load(f)
    model.fit(X1, y)
    percentage = np.around(model.feature_importances_*100, decimals=2)
    plt.figure(figsize=(10, 6))
```

```
    if mode == 'thresh':
        mask = percentage >= per
        importance = [{p < per: "indianred", p >= per: "olivedrab"}[True] for
p in percentage]
        df1 = X1.loc[:, mask]
        df2 = X2.loc[:, mask]
        plt.bar(X1.columns, height=list(percentage), color=importance)
        plt.axhline(y=per, color='black', linestyle='--', label=f'поріг у
{per}%')
        plt.title("Відсоток значущості ознак")
        plt.legend()
    if mode == 'rec':
        rfecv = RFECV(estimator=model, scoring='neg_mean_squared_error',
cv=5, n_jobs=-1)
        rfecv.fit(X1, y)
        df1 = X1[X1.columns[rfecv.support_]]
        df2 = X2[X2.columns[rfecv.support_]]
        mask = ['indianred' if not selected else 'olivedrab' for selected in
rfecv.support_]
        plt.bar(X1.columns, height=list(percentage), color=mask)
        plt.title("Відсоток значущості рекурсивно обраних ознак")
    plt.xticks(rotation=90)
    plt.show()
    return df1, df2

def make_splits(X, y, number=2):
    if number == 2:
        X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
        return X_train, X_test, y_train, y_test
    elif number == 3:
        X_train, X_check, y_train, y_check = train_test_split(X, y,
test_size=0.3, random_state=42)
        X_test, X_val, y_test, y_val = train_test_split(X_check, y_check,
test_size=0.5, random_state=42)
        return X_train, X_test, X_val, y_train, y_test, y_val

def y_division(y):
    y1 = y["Anxiety"]
    y2 = y["Insomnia"]
    y3 = y["Depression"]
    y4 = y["OCD"]
    return [y1, y2, y3, y4]

def calculate_metrics(y_train, y_train_pred, y_test, y_test_pred):
    train_rmse = mean_squared_error(y_train, y_train_pred, squared=False)
    test_rmse = mean_squared_error(y_test, y_test_pred, squared=False)
    train_mae = mean_absolute_error(y_train, y_train_pred)
    test_mae = mean_absolute_error(y_test, y_test_pred)
    train_r2 = r2_score(y_train, y_train_pred)
    test_r2 = r2_score(y_test, y_test_pred)
    return [train_rmse, test_rmse, train_mae, test_mae, train_r2, test_r2]

def calculate_clf_metrics(y_train, y_train_pred, y_test, y_test_pred):
    train_accuracy = accuracy_score(y_train, y_train_pred)
```

```
    test_accuracy = accuracy_score(y_test, y_test_pred)
    return [train_accuracy, test_accuracy]

def average_metrics(m1, m2, m3, m4):
    train_rmse = np.mean(np.array([m1[0], m2[0], m3[0], m4[0]]))
    test_rmse = np.mean(np.array([m1[1], m2[1], m3[1], m4[1]]))
    train_mae = np.mean(np.array([m1[2], m2[2], m3[2], m4[2]]))
    test_mae = np.mean(np.array([m1[3], m2[3], m3[3], m4[3]]))
    train_r2 = np.mean(np.array([m1[4], m2[4], m3[4], m4[4]]))
    test_r2 = np.mean(np.array([m1[5], m2[5], m3[5], m4[5]]))
    total_time = np.mean(np.array([m1[6], m2[6], m3[6], m4[6]]))
    return [train_rmse, test_rmse, train_mae, test_mae, train_r2, test_r2,
total_time]

def average_clf_metrics(m1, m2, m3, m4):
    train_accuracy = np.mean(np.array([m1[0], m2[0], m3[0], m4[0]]))
    test_accuracy = np.mean(np.array([m1[1], m2[1], m3[1], m4[1]]))
    total_time = np.mean(np.array([m1[2], m2[2], m3[2], m4[2]]))
    return [train_accuracy, test_accuracy, total_time]

def prediction(X_train, X_test, y_train, y_test, model, clf, rounded,
flag=False):
    if clf:
        with
open(f'/content/drive/MyDrive/Diploma/Classification/best_model_{model}.pkl',
'rb') as f:
            model = pickle.load(f)
    else:
        with
open(f'/content/drive/MyDrive/Diploma/Regression/best_model_{model}.pkl',
'rb') as f:
            model = pickle.load(f)
    if clf == True and model == 'st' and flag == True:
        pass
    else:
        start_time = time.time()
        model.fit(X_train, y_train)
        end_time = time.time()
        total_time = end_time - start_time
    if rounded:
        y_train_pred = np.round(model.predict(X_train)*2)/2
        y_test_pred = np.round(model.predict(X_test)*2)/2
    else:
        y_train_pred = model.predict(X_train)
        y_test_pred = model.predict(X_test)
    if clf:
        metrics = calculate_clf_metrics(y_train, y_train_pred, y_test,
y_test_pred)
    else:
        metrics = calculate_metrics(y_train, y_train_pred, y_test,
y_test_pred)
    metrics.append(total_time)
    return metrics
```

```
def prediction_average(X_train, X_test, y_div_train, y_div_test, model, clf,
rounded):
    m1 = prediction(X_train, X_test, y_div_train[0], y_div_test[0], model,
clf, rounded)
    m2 = prediction(X_train, X_test, y_div_train[1], y_div_test[1], model,
clf, rounded, flag=True)
    m3 = prediction(X_train, X_test, y_div_train[2], y_div_test[2], model,
clf, rounded, flag=True)
    m4 = prediction(X_train, X_test, y_div_train[3], y_div_test[3], model,
clf, rounded, flag=True)
    if clf:
        metrics = average_clf_metrics(m1, m2, m4, m4)
    else:
        metrics = average_metrics(m1, m2, m4, m4)
    return metrics

def make_prediction(X_train, X_test, y_train, y_test, y_div_train,
y_div_test, clf=False, rounded=False):
    if clf:
        m1 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'svc', clf=clf, rounded=rounded)
        m2 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'et', clf=clf, rounded=rounded)
        m3 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'bg', clf=clf, rounded=rounded)
        m4 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'xgb', clf=clf, rounded=rounded)
        m5 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'lgbm', clf=clf, rounded=rounded)
        m6 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'cat', clf=clf, rounded=rounded)
        m7 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'st', clf=clf, rounded=rounded)
        models = ['SVC', 'ExtraTreesClassifier', 'BaggingClassifier',
'XGBClassifier', 'LGBMClassifier', 'CatBoostClassifier',
'StackingClassifier']
        metrics = ['Accuracy']
    else:
        m1 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'svr', clf=clf, rounded=rounded)
        m2 = prediction(X_train, X_test, y_train, y_test, 'et', clf=clf,
rounded=rounded)
        m3 = prediction(X_train, X_test, y_train, y_test, 'bg', clf=clf,
rounded=rounded)
        m4 = prediction(X_train, X_test, y_train, y_test, 'xgb', clf=clf,
rounded=rounded)
        m5 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'lgbm', clf=clf, rounded=rounded)
        m6 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'cat', clf=clf, rounded=rounded)
        m7 = prediction_average(X_train, X_test, y_div_train, y_div_test,
'st', clf=clf, rounded=rounded)
        models = ['SVR', 'ExtraTreesRegressor', 'BaggingRegressor',
'XGBRegressor', 'LGBMRegressor', 'CatBoostRegressor', 'StackingRegressor']
        metrics = ['RMSE', 'MAE', 'R2']
```

```
    data = np.array([m1, m2, m3, m4, m5, m6, m7])
    names = ['train', 'test']
    columns = pd.MultiIndex.from_product([metrics, names])
    df = pd.DataFrame(data[:, :-1], columns=columns)
    df.insert(0, 'Модель', models)
    df["Час навчання"] = data[:, -1]
    df["Час навчання"] = df["Час навчання"].round(2)
    df.iloc[:, 1:-1] = df.iloc[:, 1:-1].round(3)
    df = df.to_string(justify='center', index=False)
    print(df)

def cross_validation(X, y, scoring, model):
    with
open(f'/content/drive/MyDrive/Diploma/Regression/best_model_{model}.pkl',
'rb') as f:
        model = pickle.load(f)
    cv_results = cross_validate(model, X, y, cv=5, scoring=scoring,
return_train_score=True)
    train_rmse = cv_results['train_RMSE'].mean()
    test_rmse = cv_results['test_RMSE'].mean()
    train_mae = cv_results['train_MAE'].mean()
    test_mae = cv_results['test_MAE'].mean()
    train_r2 = cv_results['train_R2'].mean()
    test_r2 = cv_results['test_R2'].mean()
    total_time = cv_results['fit_time'].mean()
    return [train_rmse, test_rmse, train_mae, test_mae, train_r2, test_r2,
total_time]

def cross_validation_average(X, y, scoring, model):
    cv1 = cross_validation(X, y[0], scoring, model)
    cv2 = cross_validation(X, y[1], scoring, model)
    cv3 = cross_validation(X, y[2], scoring, model)
    cv4 = cross_validation(X, y[3], scoring, model)
    metrics = average_metrics(cv1, cv2, cv3, cv4)
    return metrics

def make_cv_prediction(X, y, scoring):
    m1 = cross_validation_average(X, y, scoring, 'svr')
    m2 = cross_validation_average(X, y, scoring, 'et')
    m3 = cross_validation_average(X, y, scoring, 'bg')
    m4 = cross_validation_average(X, y, scoring, 'xgb')
    m5 = cross_validation_average(X, y, scoring, 'lgbm')
    m6 = cross_validation_average(X, y, scoring, 'cat')
    m7 = cross_validation_average(X, y, scoring, 'st')
    models = ['SVR', 'ExtraTreesRegressor', 'BaggingRegressor',
'XGBRegressor', 'LGBMRegressor', 'CatBoostRegressor', 'StackingRegressor']
    data = np.array([m1, m2, m3, m4, m5, m6, m7])
    metrics = ['RMSE', 'MAE', 'R2']
    names = ['train', 'test']
    columns = pd.MultiIndex.from_product([metrics, names])
    df = pd.DataFrame(data[:, :-1], columns=columns)
    df.insert(0, 'Модель', models)
    df["Час навчання"] = data[:, -1]
    df["Час навчання"] = df["Час навчання"].round(2)
    df.iloc[:, 1:-1] = df.iloc[:, 1:-1].round(3)
```

```
    df = df.to_string(justify='center', index=False)
    print(df)

def drop_small_classes(X, y):
    data = pd.concat([X, y], axis=1)
    unique_values = {}
    for col in y:
        unique_values[col] = data[col].value_counts()
    filtered_values = {}
    for col, value_counts in unique_values.items():
        filtered_values[col] = value_counts[value_counts >= 10].index
    for col in y:
        data = data[data[col].isin(filtered_values[col])]
    return get_X_y(data)

def make_ranges(y):
    bins = [-float('inf'), 3, 7, float('inf')]
    labels = ['low', 'medium', 'high']
    y_mod = y.copy()
    for col in y_mod:
        y_mod[col] = pd.cut(y_mod[col], bins=bins, labels=labels,
right=False)
    return y_mod

def find_association(X, y, issue):
    y_mod = y.applymap(lambda x: 'High' if x >= 5 else 'Accepted')
    X_encoded = pd.get_dummies(X)
    numerical_features = ['Age', 'Hours per day', 'BPM']
    X_encoded.drop(numerical_features, axis=1, inplace=True)
    if issue == 'anx':
        y_encoded = pd.get_dummies(y_mod[['Anxiety']])
        condition = ['Anxiety_High']
        min_support = 0.22
        min_threshold = 0.75
        columns=['Чинники тривожності']
    elif issue == 'dep':
        y_encoded = pd.get_dummies(y_mod[['Depression']])
        condition = ['Depression_High']
        min_support = 0.19
        min_threshold = 0.6
        columns=['Чинники депресії']
    elif issue == 'ins':
        y_encoded = pd.get_dummies(y_mod[['Insomnia']])
        min_support = 0.129
        min_threshold = 0.415
        condition = ['Insomnia_High']
        columns=['Чинники безсоння']
    elif issue == 'ocd':
        y_encoded = pd.get_dummies(y_mod[['OCD']])
        condition = ['OCD_High']
        min_support = 0.1
        min_threshold = 0.275
        columns=['Чинники ОКР']
    combined_data = pd.concat([X_encoded, y_encoded], axis=1)
```

```
    frequent_itemsets = apriori(combined_data, min_support=min_support,
use_colnames=True)
    rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=min_threshold)
    filtered_rules = rules[(rules['consequents'].apply(lambda x:
set(condition).issubset(x))) & ((rules['antecedents'].apply(lambda x: len(x)
>= 4)))]
    print(filtered_rules[['antecedents']].to_string(index=False,
header=columns))

def custom_research(path):
    df_init = pd.read_csv('drive/My Drive/Diploma/custom_survey.csv',
encoding='utf-8')
    X = df_init.loc[:, "Age":"Music effects"]
    y = np.array(df_init.loc[:, "Anxiety":"OCD"])
    X_stand = scale(X, mode='standard')
    X_stand_enc = encode(X_stand)
    with open(f'/content/drive/MyDrive/Diploma/Regression/best_model_bg.pkl',
'rb') as f:
        model = pickle.load(f)
    y_pred = np.round(model.predict(X_stand_enc)*2)/2
    data = np.empty((y.shape[0], y.shape[1] * 2))
    for i in range(y.shape[1]):
        data[:, i * 2] = y_pred[:, i]
        data[:, i * 2 + 1] = y[:, i]
    names1 = ['Тривожність', 'Депресія', 'Безсоння', 'ОКР']
    names2 = ['прогнозовані', 'вказані']
    columns = pd.MultiIndex.from_product([names1, names2])
    results = pd.DataFrame(data, columns=columns)
    results.insert(0, 'ID респондента', range(1, y_pred.shape[0] + 1))
    print(results.to_string(index=False, justify='center'))

"""# Підготовка датасету"""

path = "drive/My Drive/Diploma/mxmh_survey_results.csv"
df_init = pd.read_csv(path)
print(df_init.head())

df = df_init.loc[:, "Age":"Music effects"]
new_columns = {col: col.replace('[', '(').replace(']', ')') if '[' in col
else col for col in df.columns}
df.rename(columns=new_columns, inplace=True)
print(df.head())

"""# Попередня обробка даних

## Статистичне дослідження
"""

df_stats(df)

"""## Заповнення пропусків"""

plot_nan_percentage(df)
fill_nan(df)
```

```
"""## Видалення аномалій"""

anomaly_check(df)
df["BPM"].replace(df["BPM"].max(), np.nan, inplace=True)
df["BPM"].fillna(df["BPM"].median(), inplace=True)
anomaly_check(df, mode='check')

"""## Кодування та масштабування

"""

X, y = get_X_y(df)
X_norm = scale(X, mode='minmax')
X_stand = scale(X, mode='standard')
X_enc_norm_old = encode(X_norm)
X_enc_stand_old = encode(X_stand)

"""## Дослідження розподілів"""

plot_distribution(X)
plot_distribution(X_norm, 'minmax')
plot_distribution(X_stand, 'standard')
plot_variables(y)

"""## Grid Search для вибору значущих ознак"""

param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}
grid_search(X_enc_stand_old, y, param_grid, 'rf')

"""## Вибір значущих ознак"""

correlation(X_enc_stand_old)
correlation(y, mode='variables')

X_enc_stand_per, X_enc_norm_per = select_features(X_enc_stand_old,
X_enc_norm_old, y, mode='thresh')
X_enc_stand_rec, X_enc_norm_rec = select_features(X_enc_stand_old,
X_enc_norm_old, y, mode='rec')

"""# Grid Search для моделей"""

X_train, X_test, X_val, y_train, y_test, y_val = make_splits(X_enc_stand_per,
y, number=3)
y_div_train = y_division(y_train)
y_div_test = y_division(y_test)
y_div_val = y_division(y_val)

param_grid = {
```

```
    'kernel': ['linear', 'poly', 'rbf'],
    'C': [0.1, 1, 10],
    'gamma': ['scale', 'auto'],
    'epsilon': [0.1, 0.2, 0.3]
}
grid_search(X_val, y_div_val[0], param_grid, 'svr')

param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}
grid_search(X_val, y_val, param_grid, 'et')

param_grid = {
    'n_estimators': [50, 100, 150],
    'max_samples': [0.6, 0.8, 1.0],
    'max_features': [0.6, 0.8, 1.0],
    'bootstrap': [True, False],
    'bootstrap_features': [True, False],
}
grid_search(X_val, y_val, param_grid, 'bg')

param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.1, 0.01, 0.001],
    'max_depth': [3, 5, 7],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'gamma': [0, 0.1, 0.5],
    'reg_alpha': [0, 0.1, 0.5],
    'reg_lambda': [0, 0.1, 0.5]
}
grid_search(X_val, y_val, param_grid, 'xgb')

param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.1, 0.01, 0.001],
    'max_depth': [3, 5, 7],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'reg_alpha': [0, 0.1, 0.5],
    'reg_lambda': [0, 0.1, 0.5]
}
grid_search(X_val, y_div_val[0], param_grid, 'lgbm')

param_grid = {
    'iterations': [100, 200, 300],
    'learning_rate': [0.1, 0.01, 0.001],
    'depth': [3, 5, 7],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bylevel': [0.6, 0.8, 1.0],
    'reg_lambda': [0, 0.1, 0.5]
```

```
}
grid_search(X_val, y_div_val[0], param_grid, 'cat')

with open('/content/drive/MyDrive/Diploma/Regression/best_model_xgb.pkl',
'rb') as f:
    xgb = pickle.load(f)
with open('/content/drive/MyDrive/Diploma/Regression/best_model_lgbm.pkl',
'rb') as f:
    lgbm = pickle.load(f)
with open('/content/drive/MyDrive/Diploma/Regression/best_model_cat.pkl',
'rb') as f:
    cat = pickle.load(f)
estimators = [
    ('xgb', xgb),
    ('lgbm', lgbm),
    ('cat', cat)
]
with open(f'/content/drive/MyDrive/Diploma/Regression/best_model_et.pkl',
'rb') as f:
    final_estimator = pickle.load(f)
st = StackingRegressor(estimators=estimators,
final_estimator=final_estimator)
with open('/content/drive/MyDrive/Diploma/Regression/best_model_st.pkl',
'wb') as f:
    pickle.dump(st, f)

"""# Прогнози за найкращими моделями

## Пороговий вибір ознак
"""

X_train_stand_per, X_test_stand_per, y_train_stand_per, y_test_stand_per =
make_splits(X_enc_stand_per, y)
y_div_train_stand_per = y_division(y_train_stand_per)
y_div_test_stand_per = y_division(y_test_stand_per)

make_prediction(X_train_stand_per, X_test_stand_per, y_train_stand_per,
y_test_stand_per, y_div_train_stand_per, y_div_test_stand_per)

"""## Рекурсивний вибір ознак"""

X_train_stand_rec, X_test_stand_rec, y_train_stand_rec, y_test_stand_rec =
make_splits(X_enc_stand_rec, y)
y_div_train_stand_rec = y_division(y_train_stand_rec)
y_div_test_stand_rec = y_division(y_test_stand_rec)

make_prediction(X_train_stand_rec, X_test_stand_rec, y_train_stand_rec,
y_test_stand_rec, y_div_train_stand_rec, y_div_test_stand_rec)

"""## Нормалізовані дані"""

X_train_norm_rec, X_test_norm_rec, y_train_norm_rec, y_test_norm_rec =
make_splits(X_enc_norm_rec, y)
y_div_train_norm_rec = y_division(y_train_norm_rec)
y_div_test_norm_rec = y_division(y_test_norm_rec)
```

```
make_prediction(X_train_norm_rec, X_test_norm_rec, y_train_norm_rec,
y_test_norm_rec, y_div_train_norm_rec, y_div_test_norm_rec)

"""# Покращення моделей

## Крос-валідація
"""

y_div = y_division(y)
scoring = {'R2': 'r2',
           'RMSE': make_scorer(mean_squared_error, squared=False),
           'MAE': make_scorer(mean_absolute_error)}

make_cv_prediction(X_enc_stand_rec, y_div, scoring)

"""## Округлені прогнози"""

make_prediction(X_train_stand_rec, X_test_stand_rec, y_train_stand_rec,
y_test_stand_rec, y_div_train_stand_rec, y_div_test_stand_rec, rounded=True)

"""## Класифікація

### Збереження класифікаторів
"""

svc = SVC(kernel='rbf', C=1, gamma='auto', random_state=42)
with open('/content/drive/MyDrive/Diploma/Classification/best_model_svc.pkl',
'wb') as f:
  pickle.dump(svc, f)

et = ExtraTreesClassifier(n_estimators=100, max_depth=5, min_samples_leaf=4,
min_samples_split=10, max_features='log2', random_state=42)
with open('/content/drive/MyDrive/Diploma/Classification/best_model_et.pkl',
'wb') as f:
  pickle.dump(et, f)

with open('/content/drive/MyDrive/Diploma/Classification/best_model_et.pkl',
'rb') as f:
  base_estimator = pickle.load(f)
bg = BaggingClassifier(estimator=base_estimator, n_estimators=50,
max_samples=0.8, max_features=0.6, bootstrap=False, bootstrap_features=False,
random_state=42)
with open('/content/drive/MyDrive/Diploma/Classification/best_model_bg.pkl',
'wb') as f:
  pickle.dump(bg, f)

xgb = XGBClassifier(n_estimators=300, subsample=0.6, max_depth=7,
learning_rate=0.01, colsample_bytree=0.6, gamma=0.1, reg_alpha=0.5,
reg_lambda=0.1, random_state=42)
with open('/content/drive/MyDrive/Diploma/Classification/best_model_xgb.pkl',
'wb') as f:
    pickle.dump(xgb, f)
```

```
lgbm = LGBMClassifier(n_estimators=100, learning_rate=0.001, max_depth=3,
subsample=0.6, colsample_bytree=0.6, reg_alpha=0.1, reg_lambda=0.5,
random_state=42)
with
open('/content/drive/MyDrive/Diploma/Classification/best_model_lgbm.pkl',
'wb') as f:
  pickle.dump(lgbm, f)

cat = CatBoostClassifier(iterations=300, depth=5, subsample=0.8,
colsample_bylevel=0.6, learning_rate=0.1, reg_lambda=0.5,
bootstrap_type='MVS', random_seed=42, verbose=False)
with open('/content/drive/MyDrive/Diploma/Classification/best_model_cat.pkl',
'wb') as f:
  pickle.dump(cat, f)

with
open(f'/content/drive/MyDrive/Diploma/Classification/best_model_xgb.pkl',
'rb') as f:
    xgb = pickle.load(f)
with
open(f'/content/drive/MyDrive/Diploma/Classification/best_model_lgbm.pkl',
'rb') as f:
    lgbm = pickle.load(f)
with
open(f'/content/drive/MyDrive/Diploma/Classification/best_model_cat.pkl',
'rb') as f:
    cat = pickle.load(f)
estimators = [
    ('xgb', xgb),
    ('lgbm', lgbm),
    ('cat', cat)
]
with open(f'/content/drive/MyDrive/Diploma/Classification/best_model_et.pkl',
'rb') as f:
    final_estimator = pickle.load(f)
st = StackingClassifier(estimators=estimators,
final_estimator=final_estimator)
with open('/content/drive/MyDrive/Diploma/Classification/best_model_st.pkl',
'wb') as f:
    pickle.dump(st, f)

"""### Результат класифікації"""

X_enc_stand_red, y_red = drop_small_classes(X_enc_stand_rec, y)
y_enc = encode(y_red, mode='all')
X_train_clf, X_test_clf, y_train_clf, y_test_clf =
train_test_split(X_enc_stand_red, y_enc)
y_div_train_clf = y_division(y_train_clf)
y_div_test_clf = y_division(y_test_clf)

make_prediction(X_train_clf, X_test_clf, y_train_clf, y_test_clf,
y_div_train_clf, y_div_test_clf, clf=True)

"""## Альтернативні задачі
```

```
### Модифікована класифікація
"""

y_mod = make_ranges(y)
y_mod_enc = encode(y_mod, mode='all')
X_train_clf_mod, X_test_clf_mod, y_train_clf_mod, y_test_clf_mod =
make_splits(X_enc_stand_rec, y_mod_enc)
y_div_train_clf_mod = y_division(y_train_clf_mod)
y_div_test_clf_mod = y_division(y_test_clf_mod)

make_prediction(X_train_clf_mod, X_test_clf_mod, y_train_clf_mod,
y_test_clf_mod, y_div_train_clf_mod, y_div_test_clf_mod, clf=True)

"""### Пошук асоціативних правил"""

find_association(X, y, 'anx')
find_association(X, y, 'dep')
find_association(X, y, 'ins')
find_association(X, y, 'ocd')

"""# Результат на власному опитуванні"""

custom_research('drive/My Drive/Diploma/custom_survey.csv')
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ

Слайд 1

ПРОГНОЗУВАННЯ ПСИХІЧНОГО ЗДОРОВ'Я НА ОСНОВІ ДАНИХ ПРО МУЗИЧНІ ВПОДОБАННЯ

Виконала: студентка IV курсу, групи КА-93
Ланько Анна Анатоліївна

Керівник: д.т.н., доцент
Недашківська Надія Іванівна

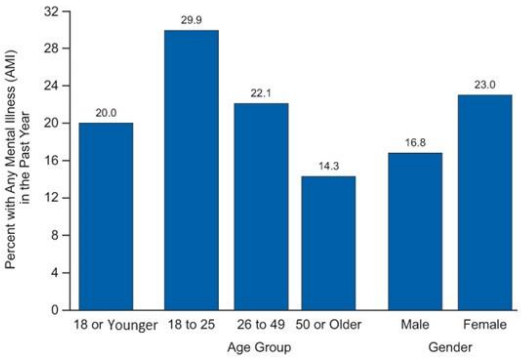
Слайд 2

АКТУАЛЬНІСТЬ ТЕМИ ДОСЛІДЖЕННЯ

Незначні ментальні порушення зазвичай виникають в умовах постійного стресу та ігноруються суспільством, у той час як **їх хронічна наявність послаблює нервову систему людини та поступово еволюціонує у більш серйозні розлади.**

Дане дослідження є інструментом попередження розвитку серйозних психічних аномалій шляхом своєчасного виявлення проблеми на основі загальнодоступних даних про музичні вподобання.

Вибір вихідних даних зумовлено науково обґрунтованим зв'язком між емоційним станом та характером музичних вподобань.



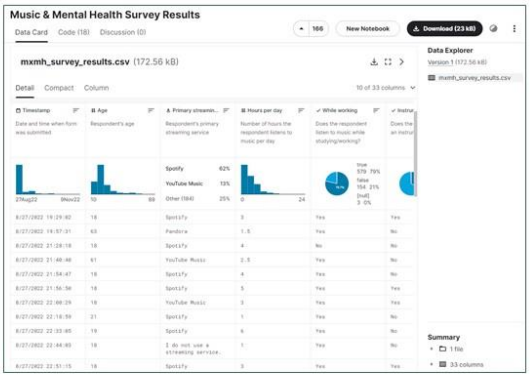
Відсоток людей з наявністю ментальних розладів за віковими та гендерними категоріями у 2022 р.

Слайд 3

ВИХІДНІ ДАНІ

Дані було зібрано користувачем відкритої платформи для обміну досвідом та матеріалами у галузі машинного навчання *Kaggle* двома способами:

1) **опитування у мережі** (посилання на гугл-форму в соціальних мережах, на форумах та Discord-серверах);
2) **«живе» опитування** (на вулицях, у громадських місцях).



Слайд 4

ПОСТАНОВКА ЗАДАЧІ

Суть задачі полягає в **прогнозуванні ступеня наявності тривожності, безсоння, депресії та обсесивно-компульсивного розладу в певної особи у вигляді оцінки в діапазоні від 0 до 10 на основі 27 категоріальних та числових ознак.**

Формалізована задача:
Нехай X – множина ознак, а Y – множина цільових змінних. Існує невідома цільова залежність (відображення) $y^*: X \to Y$, значення якої відомі лише для об'єктів скінченної вибірки $X^m = \{(x_1; y_1), \dots, (x_m; y_m)\}$. Потрібно побудувати функцію $f: X \to Y$, яка допоможе прогнозувати оцінки станів для довільного зразка $x \in X$.

Оскільки прогнозується числове значення, а не належність до класу, **маємо задачу регресії**.

X			
№	«Age»	…	«Music effect»
1			
…			
736			

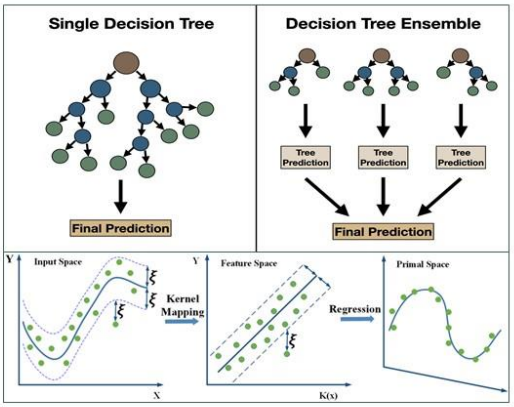
Y			
№	«Anxiety»	…	«OCD»
1			
…			
736			

Слайд 5

ОГЛЯД МЕТОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ

З огляду на складність вихідних даних, було розглянуто більш складні методи розв'язання задачі регресії:

1) **ансамблеві методи** – комбінація кількох дерев рішень (прогнозують значення за розбиттям даних на менші групи) для покращення точності прогнозування та уникнення перенавчання;
2) **метод опорних векторів** – регресія (SVR) на основі знаходження гіперплощини, яка найкраще розділяє дані у просторі ознак, підходить для даних високої розмірності.

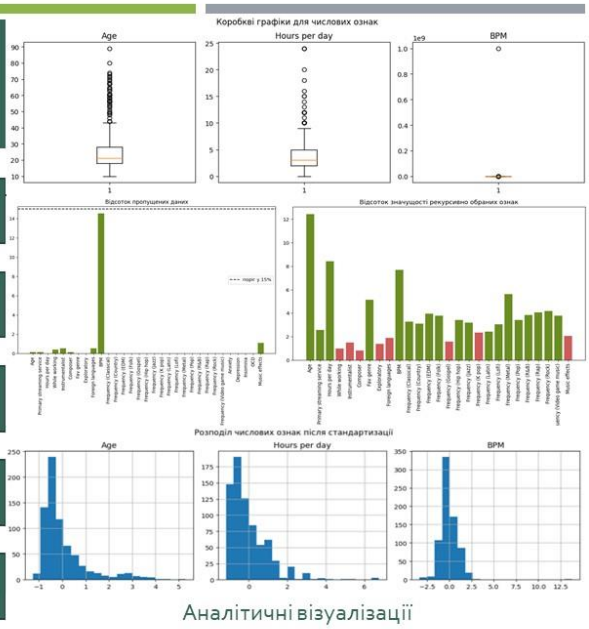


Концепція ансамблів (зверху) та SVR (знизу)

Слайд 6

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

- Попереднє форматування
- Видалення аномалій
- Заповнення пропусків
- Вибір значущих ознак
- Кодування та масштабування



Аналітичні візуалізації

Слайд 7

НАЛАШТУВАННЯ ГІПЕРПАРАМЕТРІВ МОДЕЛЕЙ

1. З наявного переліку гіперпараметрів для кожної моделі було обрано найбільш суттєві.

2. Для кожного з них було визначено найдоцільніші варіанти значень.

3. Таким чином отримали множину комбінацій для кожної моделі.

Результат:
За найкращий набір гіперпараметрів для кожної моделі було обрано конфігурацію з мінімальною середньоквадратичною похибкою (MSE) на валідаційній вибірці.

Назва гіперпараметра	Розглянуті значення	Найкраще значення
n_estimators	50, 100, 150	50
max_samples	0.6, 0.8, 1.0	0.8
max_features	0.6, 0.8, 1.0	0.8
bootstrap	True, False	False
bootstrap_features	True, False	False

Приклад підбору для найкращої моделі – **BaggingRegressor**

Слайд 8

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Найкраща модель – **BaggingRegressor**

Модель	Метрики якості						Час навчання, с
	RMSE		MAE		R2		
	train	test	train	test	train	test	
VR	2,565	3,149	1,865	2,513	0,212	-0,173	0,03
ExtraTreesRegressor	2,715	2,905	2,313	2,458	0,154	-0,012	0,16
BaggingRegressor	2,737	2,894	2,331	2,448	0,141	-0,004	12,38
XGBRegressor	1,402	3,008	1,147	2,511	0,774	-0,008	2,05
LGBMRegressor	2,878	2,979	2,426	2,485	0,013	-0,03	0,02
CatBoostRegreesor	2,38	3,031	1,983	2,517	0,323	-0,068	0,04
StackingCVRegressor	2,788	2,975	2,347	2,48	0,067	-0,029	6,14

Слайд 9

СПРОБИ ПОКРАЩЕННЯ МОДЕЛЕЙ

Можливі причини низької результативності:

1) використання некоректних методів розбиття даних;
2) неправильне застосування метрик;
3) неправильна інтерпретація задачі;
4) складність, слабка кореляція даних.

Висновок:
Результат експериментів довів, що низькі значення метрик викликані складністю вихідних даних та суб'єктивністю оцінок.

1. Перехресна перевірка
(результат мінімально покращився)

2. Округлення результатів регресії
(результат не змінився)

3. Класифікація оцінок
(низька результативність)

Слайд 10

АЛЬТЕРНАТИВНІ ЗАДАЧІ

З метою покращення результативності дослідження було розглянуто наступні альтернативні задачі:

1. **Класифікація ментального стану за рівнями наявності розладів:**
   1) 0-3 – «Низький»;
   2) 4-7 – «Середній»;
   3) 8-10 – «Високий».
2. **Пошук асоціативних правил.**

Висновок:
Точність такої класифікації за всіма моделями варіюється в межах 50%±3% і є практично прийнятною для складних даних, побудовані асоціативні правила є слабкими за змістом.

№	Набір чинників тривожності
1	«While working»: так, «Exploratory»: так, «Primary streaming service»: Spotify, «Music effects»: покращує
2	«While working»: так, «Primary streaming service»: Spotify, «Frequency (Gospel)»: ніколи, «Music effects»: покращує
3	«While working»: так, «Exploratory»: так, «Frequency (Gospel)»: ніколи, «Music effects»: покращує

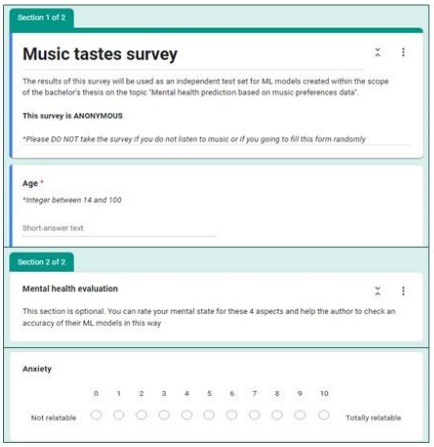
Приклад найпоширеніших умов (X) для асоціативного правила $X \rightarrow Y$, де Y – тривожність

Слайд 11

СТВОРЕННЯ ВЛАСНОГО ОПИТУВАЛЬНИКА

Анонімний опитувальник містить лише значущі для алгоритму питання та, на відміну від вихідних даних, має дві секції:

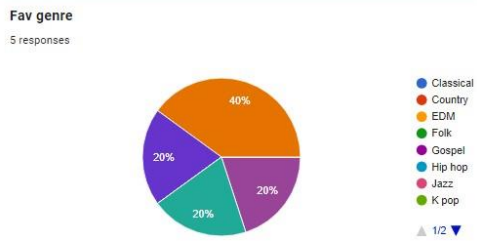
1) **обов'язкові питання** – ознаки, що будуть використані для прогнозу;
2) **вибіркові питання** – респондент за бажанням міг оцінити свій ментальний стан; ці відповіді допоможуть оцінити адекватність спрогнозованої оцінки.



Слайд 12

РЕЗУЛЬТАТИ НА ВЛАСНИХ ЗНАЧЕННЯХ

На опитування дало відповідь 5 осіб, кожній особі присвоєний ID від 1 до 5.



Приклад відповідей на питання про улюблений жанр

ID	Тривожність		Депресія		Безсоння		ОКР	
	pred	true	pred	true	pred	true	pred	true
1	5,5	5	4,5	3	3	2	2	2
2	5,5	6	4	2	3	1	2,5	1
3	5,5	4	4,5	3	3,5	4	2,5	0
4	5	3	3,5	4	3	2	0	1
5	5,5	7	4,5	5	3	0	2,5	5

Регресія для найкращої моделі – **BaggingRegressor**

Висновок:
Алгоритм не є усередненим, оскільки реагує на зміни значень між респондентами.
Результат є практично прийнятним

Слайд 13

ВИСНОВКИ

1. **Найкращою моделлю є бегінг ансамблів на основі особливо випадкового лісу.** Це пояснюється випадковістю розбиттів та незалежним навчанням дерев з подальшою агрегацією прогнозу, що попереджує перенавчання і допомагає знаходити складні нелінійні залежності між даними.
2. З огляду на гарний прогноз на власній вибірці та достатню якість модифікованої класифікації, **задачу можна вважати успішно розв'язаною, а розроблений алгоритм – практично прийнятним з точки зору результативності.**
3. **Недоліком розробленого алгоритму є не надто висока результативність,** що пояснюється складною залежністю всередині даних та суб'єктивністю визначення фактичних значень цільових змінних.
4. **Покращити його можна шляхом застосування нейронних мереж,** що не було реалізовано в даній роботі з огляду на замалий обсяг даних та масштаб предмету дослідження.

Слайд 14

ПЕРСПЕКТИВИ ДОСЛІДЖЕННЯ

Дані про музичні вподобання легко зібрати у великому обсязі на основі персональної інформації користувачів стрімінгових платформ.

Отриманий прогноз на основі повідомлень, рекомендацій та обмежень може **допомогти конкретній особі звернути увагу на свій ментальний стан і тим самим попередити розвиток психічних порушень**.

Щодо вищезгаданих ресурсів, розроблений програмний продукт допоможе **налагодити рекомендаційні системи та створити більш здорове середовище у спільноті**.

Слайд 15

ДЯКУЮ ЗА УВАГУ!