КОМП'ЮТЕРНІ НАУКИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

УДК 637.5.02
DOI https://doi.org/10.32782/tnv-tech.2024.2.1

ОБРОБКА ТА АНАЛІЗ ДАНИХ НА ПРИКЛАДІ НАБОРУ SPAMBASE З ВИКОРИСТАННЯМ БІБЛІОТЕК ДЛЯ МАШИННОГО НАВЧАННЯ

***Балвак А. А.** – аспірант*
Державного університету інформаційно-комунікаційних технологій
ORCID ID: 0000-0002-6441-8225

***Лемешко А. В.** – доктор філософії, доцент,*
доцент кафедри комп'ютерної інженерії
Державного університету інформаційно-комунікаційних технологій
ORCID ID: 0000-0001-8003-3168

***Антоненко А. В.** – кандидат технічних наук, доцент,*
доцент кафедри стандартизації та сертифікації сільськогосподарської продукції
Національного університету біоресурсів і природокористування України
ORCID ID: 0000-0001-9397-1209

***Зіняр Д. А.** – аспірант*
Державного університету інформаційно-комунікаційних технологій
ORCID ID: 0000-0002-7282-2682

***Бурачинський А. Ю.** – аспірант*
Державного університету інформаційно-комунікаційних технологій
ORCID ID: 0009-0003-7913-2152

***Приходько А. П.** – магістр*
Міжрегіональної академії управління персоналом
ORCID ID: 0009-0001-2365-9903

У статті проаналізовано набір даних Spambase з даними про електронні листи, які класифіковано як спам та не спам. Наведено детальний аналіз цього датафрейма з інформацією про дані в стовпчиках (властивості) та записи набору. Дані набору було завантажено в середовище для розробки програмного забезпечення Google CoLab для програмування та подальшого аналізу. Для наукових обчислень та аналізу даних в Python використано бібліотеки NumPy, Pandas, Matplotlib, Sklearn, Imblearn. Їх комбінації дозволяють розробникам та дослідникам ефективно працювати зі структурованими даними, виконувати різні операції з ними, візуалізувати результати та розв'язувати складні завдання аналізу та обробки даних. Для кращого розуміння матеріалу розглянуто деякі основні теоретичні відомості щодо прогнозування даних. Надано визначення машинного навчання, штучного інтелекту та науки про дані. Також описано категорії машинного навчання, такі як кероване, некероване навчання та навчання з підкріпленням. Розглянуті основні типи ознак, які використовуються в моделях машинного навчання: якісні, впорядковані та кількісні ознаки. Також був представлений набір даних про хвороби серця Heart Disease на якому описано та позначено важливі визначення, такі як матриця ознак X, вектор ознак, властивості, вектор цільових значень Y. Описано необхідність розбиття набору даних на навчальний, перевірочний та тестовий для коректної оцінки та перевірки моделі. Також пояснено використання функцій втрат L_1 та L_2 для оцінки продуктивності моделі, вказано переваги та недоліки кожного підходу. Продовжено аналіз датасету Spambase в середовищі Google Colab. Побудовано гістограми, що представляють розподіл даних за різними властивостями для двох класів: спаму та не спаму. Проаналізовано гістограми для властивостей word_freq_credit, char_freq_! та capital_run_length_total. Функцією split() з бібліотеки NumPy розділено дані на навчальний, перевірочний та тестовий набори. Для набору даних на навчання виконано перебалансування класів за допомогою методу випадкової перевибірки (RandomOverSampler). В результаті було створено нові екземпляри для менш представленого класу листів які містять спам.

Ключові слова: *набір даних, машинне навчання, штучний інтелект, властивості, матриця ознак, вектор ознак, вектор цільових значень.*

Balvak A. A., Lemeshko A. V., Antonenko A. V., Ziniar D. A., Burachynskyi A. Yu., Prykhodko A. P. Data processing and analysis on the example of the spambase dataset using machine learning libraries

The article analyzes the Spambase dataset with data on e-mails classified as spam and non-spam. A detailed analysis of this data frame is provided with information about the data in the columns (properties) and records. The dataset was uploaded to the Google CoLab software development environment for programming and further analysis. NumPy, Pandas, Matplotlib, Sklearn, and Imblearn libraries are used for scientific calculations and data analysis in Python. Their combinations allow developers and researchers to effectively work with structured data, perform various operations, visualize results and solve complex data analysis and processing tasks. To better understand the material, reviewed basic theoretical information about data forecasting. Definitions of machine learning, artificial intelligence, and data science are provided. Machine learning categories such as supervised, unsupervised, and reinforcement learning are also described. The main types of features used in machine learning models are considered: qualitative, ordinal, and quantitative. The Heart Disease dataset was also presented, describing and labelling important definitions such as features matrix X, feature vector, properties, targets vector Y. The need to break up the whole dataset into training, validation and testing datasets for correct evaluation and model verification is described. The use of L_1 and L_2 loss functions to evaluate model performance is explained, and the advantages and disadvantages of each approach are indicated. The analysis of the Spambase dataset in the Google Colab environment continued. Histograms were constructed to represent the distribution of data by different properties for two classes: spam and non-spam. Analyzed histograms for properties word_freq_credit, char_freq_! and capital_run_length_total. The split() function from the NumPy library splits the data into training, validation, and testing sets. For the training dataset, classes were rebalanced using the random oversampling method (RandomOverSampler). As a result, new instances were created for the less-represented class of e-mails containing spam.

Key words: *dataset, machine learning, artificial intelligence, properties, features matrix, feature vector, targets vector.*

Вступ. У сучасному світі обробка та аналіз даних знаходяться в центрі уваги, особливо в контексті машинного навчання та штучного інтелекту. Ці технології знаходять широке застосування в багатьох галузях.

Наприклад, у телекомунікаціях [1], машинне навчання використовується для передбачення попиту на послуги зв'язку, аналізу великих обсягів даних для виявлення аномалій та уразливостей мереж, а також для покращення ефективності маркетингових кампаній та персоналізації пропозицій.

У транспортній та складській логістиці [2], машинне навчання допомагає розв'язувати проблеми маршрутизації та оптимізації доставок, передбачати попит на транспортні послуги, управляти запасами на складах тощо.

В освіті [3] алгоритми машинного навчання застосовуються для різноманітних цілей, включаючи індивідуалізоване навчання, автоматизацію процесів оцінювання, розробку персоналізованих навчальних програм, аналіз даних для виявлення успішних педагогічних підходів та покращення методів викладання.

Ці приклади демонструють широкий спектр можливостей, які надає машинне навчання в різних галузях, тому важливо розуміти та використовувати ці технології для досягнення успіху в бізнесі.

Постановка проблеми. Дані стають все більшим цінним ресурсом у різних сферах життя, тому для багатьох виникає необхідність здобуття знань про машинне навчання. Багато людей, попри їхні потенційні можливості, можуть відчувати себе відчуженими від цих технологій через складність та недоступність. Для досягнення успіхів в науці, а також для розвитку суспільства в цілому, важливо зробити матеріали про машинне навчання доступними та зрозумілими широкому загалу. Крім того, доступ до розуміння машинного навчання має важливе значення для забезпечення рівних можливостей у сфері освіти та кар'єрного зростання. Збільшення числа людей, які матимуть розуміння машинного навчання сприятиме розвитку та появі нових інноваційних рішень.

Отже, існує необхідність у створенні доступних та зрозумілих матеріалів про машинне навчання, які дозволять більшій кількості громадян різного віку від школярів до дорослих різних професій та навіть пенсіонерів, які бажають розвиватись та переорієнтуватись, здобувати цінні знання та використовувати їх на практиці.

Мета дослідження. Метою дослідження є пошук відповідних датасетів у репозитаріях даних, аналіз інформації про них та подальше завантаження для детального розбору за допомогою інструментів мови Python.

Об'єктом дослідження є відповідні датасети з репозитаріїв даних.

Предметом дослідження є процес опрацювання набору даних, включаючи створення гістограм та аналіз їхніх характеристик, а також розбиття набору на навчальний, перевірочний та тестовий для подальшого використання у моделях машинного навчання.

Методи дослідження полягають у пошуку та вивченні наукових статей, книг, Youtube-відео, що стосуються основ машинного навчання, вибору наборів даних та їх аналізу.

Аналіз останніх досліджень і публікацій. Дослідження в областях машинного навчання та штучного інтелекту проводяться протягом багатьох років і мають широкий спектр застосувань. Для початку ознайомлення з цими темами можна переглянути деякі книги [4-6] та відео на платформі YouTube [7, 8].

Завершивши перші кроки в освоєнні машинного навчання та штучного інтелекту, можна продовжити поглиблювати знання та розуміння конкретних методів і алгоритмів, читаючи та вивчаючи відповідні академічні статті. Наприклад, статті в журналах “Journal of Machine Learning Research”, “IEEE Transactions on Pattern Analysis and Machine Intelligence” тощо. Також можна пройти відповідні онлайн-курси для поглиблення вивчення конкретних методів машинного

навчання. Наприклад, спеціалізовані курси на платформах Coursera, Udacity. Самостійне розроблення та виконання проєктів з машинного навчання допоможе закріпити знання і розуміння практичного застосування методів. Участь у конкурсах на платформі Kaggle допоможе максимально ефективно реалізувати майбутні проєкти завдяки можливості взяти участь у змаганнях з машинного навчання, що забезпечують широкий спектр завдань і доступ до різноманітних наборів даних для аналізу та моделювання.

Виклад основного матеріалу дослідження. Для аналізу з онлайн-ресурсу UCI Machine Learning Repository, що містить набори даних для машинного навчання використаємо датафрейм Spambase [9].

У цьому датасеті містяться дані про електронні листи які класифікуються як спам або не спам (значення в стовпчику *Class = 1,* то лист містить спам, а якщо значення в стовпчику *Class = 0,* то лист не містить спаму). Визначення **спам** може варіюватися залежно від контексту та критеріїв, встановлених користувачем чи організацією. Часто **спам-повідомленнями** вважаються такі повідомлення, що розсилаються без згоди одержувачів та рекламують товари та послуги або містять обманливу інформацію, наприклад, лотерейні шахрайства, запити на фінансову допомогу або спроби отримати особисту інформацію. Електронними листами **без спаму** можна вважати листи, які надіслані від імені офіційних джерел або організацій і мають легітимний характер.

Набір даних містить 57 властивостей та вектор цільових значень. З них 48 властивостей типу *word_freq_WORD* – це відсоток слів в електронному листі, які відповідають *WORD*, де *WORD* – слова, кількості яких обчислюються в листах. Наприклад, для властивостей *word_freq_make, word_freq_address, word_freq_all* – це відсотки входження слів *“make”, “address”, “all”* відповідно. *word_freq_WORD* визначається за формулою:

$$word\_freq\_WORD = \frac{N_{word}}{N_{заг.}} \times 100\% \quad (1)$$

де N_{word} – кількість слів WORD в електронному листі;

$N_{заг.}$ – загальна кількість слів в електронному листі.

Перелік властивостей типу word_freq_WORD – *word_freq_make, word_freq_address, word_freq_all, word_freq_3d, word_freq_our, word_freq_over, word_freq_remove, word_freq_internet, word_freq_order, word_freq_mail, word_freq_receive, word_freq_will, word_freq_people, word_freq_report, word_freq_addresses, word_freq_free, word_freq_business, word_freq_email, word_freq_you, word_freq_credit, word_freq_your, word_freq_font, word_freq_000, word_freq_money, word_freq_hp, word_freq_hpl, word_freq_george, word_freq_650, word_freq_lab, word_freq_labs, word_freq_telnet, word_freq_857, word_freq_data, word_freq_415, word_freq_85, word_freq_technology, word_freq_1999, word_freq_parts, word_freq_pm, word_freq_direct, word_freq_cs, word_freq_meeting, word_freq_original, word_freq_project, word_freq_re, word_freq_edu, word_freq_table, word_freq_conference*.

Датасет містить 6 наступних властивостей *char_freq_;, char_freq_(, char_freq_[, char_freq_!, char_freq_$, char_freq_#*, які відповідно визначають відсоток символів *“;”, “(”, “[”, “!”, “$”, “#” в* електронному листі.

Інші властивості – *capital_run_length_average, capital_run_length_longest, capital_run_length_total. capital_run_length_average* – середня довжина безперервних послідовностей великих літер; *capital_run_length_longest* – довжина найдовшої безперервної послідовності великих літер; *capital_run_length_total* – загальна кількість великих літер в електронному листі.

У векторі цільових значень *Class* вказується, чи вважався електронний лист спамом *(1)* чи ні *(0)*.

У наборі даних Spambase міститься 4601 запис з них 1813 записів (39.4%) про листи які містять спам та 2788 записів (60.6%) про листи які не містять спаму.

Тепер ми використаємо всі ці властивості, щоб мати можливості розрізняти закономірності та передбачувати чи містять листи спам, чи ні.

Спочатку в Google CoLab створюємо новий блокнот, називаємо його "Spambase dataset" й завантажуємо файл *"spambase.data"* з набором даних. Імпортуємо бібліотеки *NumPy*, *Pandas*, *Matplotlib*, *Sklearn*, *Imblearn* (рис. 1).

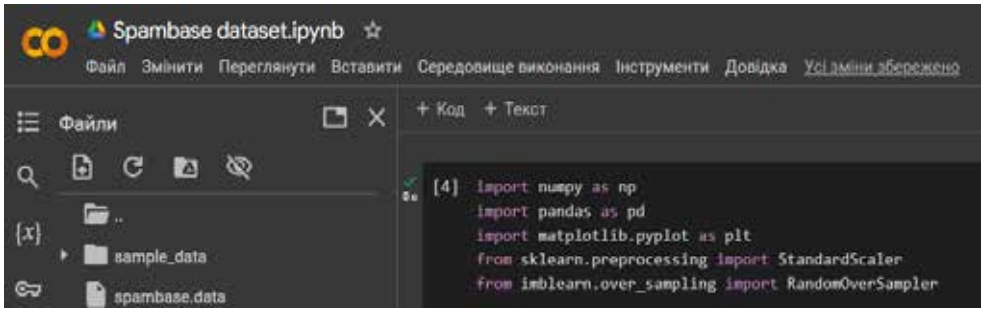


Рис. 1. Імпорт бібліотек NumPy, Pandas, Matplotlib, Sklearn, Imblearn

Короткий огляд бібліотек для аналізу даних та машинного навчання [10-12]. *Бібліотека NumPy надає підтримку для масивів та матриць, а також велику кількість математичних функцій для роботи з ними.*

Pandas – це бібліотека для обробки та аналізу даних в Python. Вона надає структури даних високого рівня, такі як DataFrame, які дозволяють зручно та ефективно працювати з табличними даними. Pandas також має багато функцій для фільтрації, групування, агрегації та візуалізації даних.

Matplotlib – це бібліотека для візуалізації даних в Python. Вона надає широкий спектр інструментів для створення різних типів графіків, включаючи лінійні графіки, гістограми, діаграми розсіювання та інші. Matplotlib дозволяє налаштовувати вигляд графіків та елементів їхнього оформлення.

Бібліотека Scikit-learn (Sklearn) містить реалізації багатьох алгоритмів машинного навчання для класифікації, регресії, кластеризації тощо. Sklearn також має інструменти для попередньої обробки даних, валідації моделей та оцінки їхньої ефективності.

Imbalanced-learn (Imblearn) – це бібліотека для роботи з незбалансованими даними в задачах класифікації в Python. Вона надає методи та інструменти для ресемплінгу даних, такі як збільшення, зменшення та синтетичне створення прикладів, які допомагають покращити роботу класифікаторів на даних з незбалансованими класами.

Щоб виконати код у комірці натискаємо клавішу *«Відтворити»* ліворуч від коду або використовуємо комбінацію клавіш *Ctrl+Enter*. Щоб імпортувати наш набір даних, необхідно перетягнути файл *"spambase.data"* в папку (рис. 1).

За допомогою функції pandas *read_csv()* прочитаємо файл *"spambase.data"*. Спочатку створили список *cols* з елементами *word_freq_make, word_freq_address, word_freq_all* та ін., тобто це назви стовпців (властивості набору даних). Параметр

names приймає список *cols* елементи якого будуть використовуватись як заголовки для створеного набору даних *df*. В результаті виконання команди *df.head()* буде виведено перші п'ять рядків даних (рис. 2).

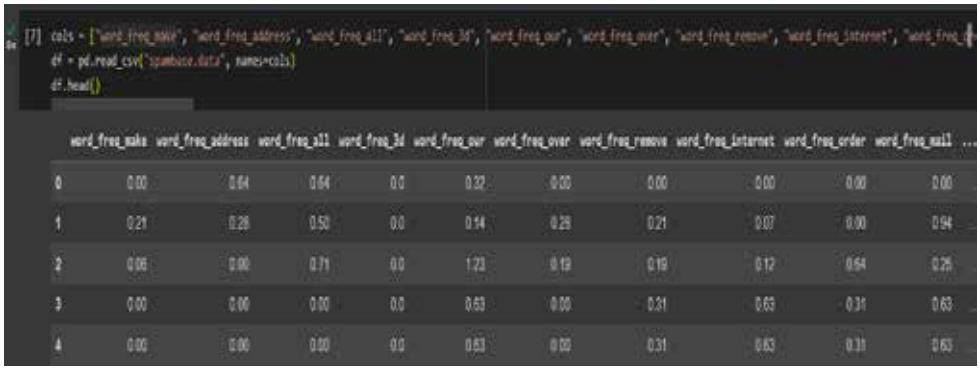


Рис. 2. Дані файлу "spambase.data"

Нам необхідно передбачити *для інших листів залежно від заданих властивостей* якими будуть значення в полі *Class чи 1, чи 0. Це називається* ***класифікацією****. Наприклад, для листа 0 на рис. 2. властивостями є значення word_freq_make = 0.00, word_freq_address = 0.64, word_freq_all = 0.64, word_freq_3d = 0.0, word_freq_our = 0.32 та інші. Властивості електронних листів будуть передані в певну модель, щоб передбачити значення міток, в цьому випадку це значення в стовпчику Class.*

Основні теоретичні відомості [4-8]. Ознайомимось з деякими визначеннями. **Машинне навчання** – це галузь інформатики, в якій вивчаються алгоритми, спрямовані на автоматичне навчання комп'ютерів на основі даних без прямого втручання програміста. Це відмінно від традиційного програмування, де людина вказує комп'ютеру конкретні дії. Штучний інтелект, машинне навчання і наука про дані – всі ці терміни пов'язані між собою, але мають свої відмінності.

Штучний інтелект (ШІ) – це галузь інформатики, що спрямована на створення систем, які можуть виконувати завдання, аналогічні до тих, що виконуються людиною, і моделювати людську поведінку.

Машинне навчання в сучасному розумінні є підгалуззю штучного інтелекту, яка спрямована на розв'язання конкретних завдань та формування прогнозів на основі аналізу та використання відповідних даних.

Наука про дані є дисципліною, спрямованою на виявлення закономірностей в накопичених даних. Це може включати застосування методів машинного навчання. Отже, ці області взаємопов'язані між собою, і в них можуть використовуватись методи машинного навчання.

Існує декілька категорій машинного навчання, серед яких **кероване навчання**. При керованому навчані використовуються відомі вхідні дані, що означає, що кожному набору вхідних даних відповідають відомі вихідні значення. Це дозволяє нам тренувати моделі та аналізувати вихідні дані залежно від вхідних.

Наприклад, маємо такі зображення (рис. 3), які для комп'ютера є наборами пікселів певних кольорів. При керованому навчанні всі ці вхідні дані мають мітки пов'язані з ними, необхідно щоб комп'ютер міг передбачити, що, наприклад, на

цій картинці зображена кішка, на цій картинці – собака, а на цій картинці – ящірка (рис. 3).

Рис. 3. Приклад коректного передбачення зображень при керованому навчанні

При **некерованому навчанні** алгоритмові не надається міток, залишаючи самостійно знаходити закономірності в даних.

Наприклад, маємо наступні вхідні дані (рис. 4), які для комп'ютера є лише зображеннями, лише пікселями.

Рис. 4. Вхідні зображення при некерованому навчанні

При некерованому навчанні комп'ютер не зможе повідомити як результат хто є кішкою, собакою та ящіркою. Але він зможе згрупувати всі ці зображення й повідомити які групи мають щось спільне, тобто знайти певні структури в немаркованих даних (рис. 5).

І, нарешті, розглянемо **навчання з підкріпленням** при якому комп'ютерна програма навчається в якомусь інтерактивному середовищі на основі винагород і покарань. Наприклад, процес дресирування собаки (рис. 6), але якщо уявити що пес – це комп'ютер. По суті, те, що ми робимо, полягає в тому, що надаємо зворотний зв'язок, аналогічний винагородам, комп’ютеру й говоримо: «Привіт, це добре, продовжуй так робити».

Рис. 5. Групування подібних зображень при некерованому навчанні

Рис. 6. Процес дресирування пса

Розглянемо кероване навчання. Модель машинного навчання виглядає так, ніби є набір вхідних даних, які передаються певній моделі. Після обробки даних модель видає результат, який є нашим прогнозом, передбаченням. Отже, усі ці вхідні дані ми називаємо вектором ознак (feature vector).

У нас можуть бути якісні ознаки. Якісні означає категоріальні дані, тобто є кінцева кількість категорій або груп. Одним із прикладів якісної характеристики може бути стать чоловіча або жіноча. Іншими прикладами категоріальних даних є різні національності, території тощо. Але в них немає властивого порядку, тобто не можна оцінити Україну як одиницю, Францію як двійку тощо. Тому ми називаємо такі дані номінальними.

Номінальні дані ми можемо передати в комп’ютер за допомогою підходу One-Hot Encoding. Наприклад, є набір вхідних даних з України, Польщі, Німеччини та Франції. Щоб комп'ютер міг розпізнати такі дані можна використати One-Hot Encoding. Якщо дані відповідають певній категорії, то це одиниця, якщо ні, то

нуль. Наприклад, для України [1, 0, 0, 0], Польщі [0, 1, 0, 0], Німеччини [0, 0, 1, 0], Франції [0, 0, 0, 1] (табл. 1).

Таблиця 1

Відповідності категоріям (країнам) бінарних змінних

Країна	Бінарна змінна
Україна	[1, 0, 0, 0]
Польща	[0, 1, 0, 0]
Німеччина	[0, 0, 1, 0]
Франція	[0, 0, 0, 1]

Є ще інший тип якісних ознак. На рис. 7 ліворуч зображено різні вікові групи: немовлята, малюки, підлітки, молоді люди, дорослі та інші. А з правого боку ми можемо мати різні оцінки, а саме: погано, не дуже добре, так собі, добре та чудово. Це вже впорядковані дані. Наприклад, малюк набагато ближче до немовляти, ніж до літньої людини, а добре ближче до чудово, ніж до погано. І тому для таких типів наборів даних можна ввести позначення від одного до п’яти або присвоїти номери. Це матиме сенс і для комп'ютера.

Рис. 7. Різні вікові групи та настрій

Існують також **кількісні дані** – це дані з числовими значеннями, які можуть бути **дискретними** тобто цілими числами або **неперервними**, тобто дійсними числами. Наприклад, довжина або температура є кількісними характеристиками. А кількість яєць у кошику – приклад **дискретної кількісної характеристики.**

Числа, які входять до вектора ознак – це дані, які передаються в модель, тому що комп’ютери дуже добре та правильно розуміють числа (рис. 8).

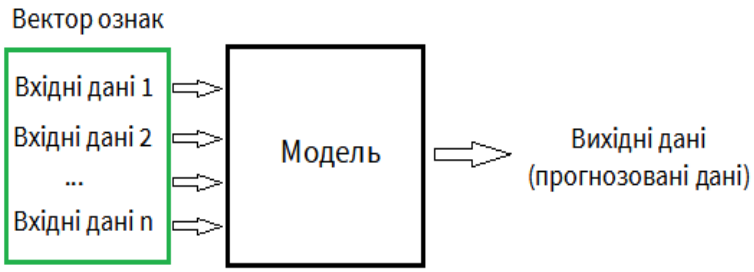


Рис. 8. Схематичне зображення передачі даних в модель для навчання

Які типи прогнозів може виводити наша модель? При керованому навчанні буває необхідно передбачити дискретні класи серед більше ніж двох варіантів, наприклад, чи це хот-дог, чи піца, чи морозиво. Це багатокласова класифікація.

Існує також бінарна класифікація, наприклад, чи це хот-дог, чи не хот-дог. Отже, є лише дві категорії, з якими працюють: чи щось є чимось, чи не є чимось.

В табл. 2 наведено різні приклади бінарної та багатокласової класифікацій.

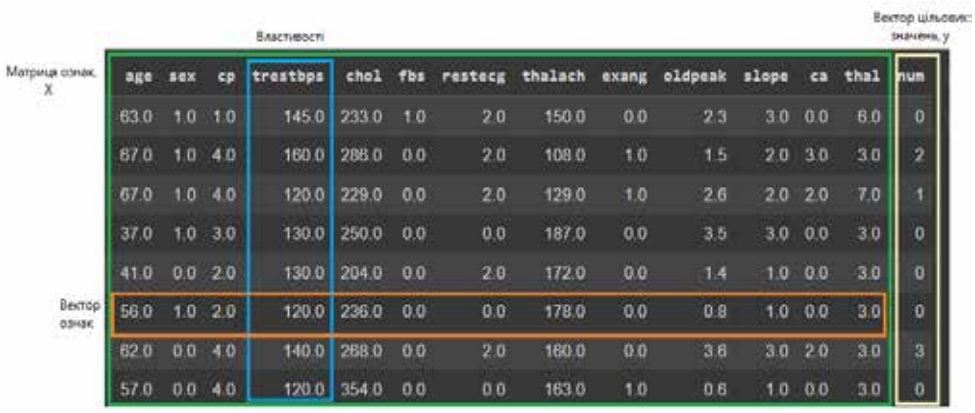
Таблиця 2

Приклади бінарної та багатокласової класифікацій

Бінарна класифікація	Багатокласова класифікація
Позитивне/негативне	Ведмідь/ящірка/акула/кит
Кішка/собака	Мандарин/лимон/груша
Спам/не спам	Троянда/фіалка/лілія

Розглянемо регресійні моделі при керованому навчанні, які використовують при прогнозуванні безперервних значень. Тобто замість того, щоб просто намагатись спрогнозувати різні категорії, ми намагаємось передбачити число, яке має певну шкалу. Наприклад, якою буде найближчими днями температура повітря або ціни на нерухомість. Ми намагаємось передбачити числа, які є якомога ближчими до справжніх значень.

Розглянемо набір даних Heart Disease [13] (рис. 9), який можна завантажити з онлайн-репозитарію UCI Machine Learning Repository. Датасет про хвороби серця містить клінічні параметри пацієнтів, що можуть бути використані для прогнозування наявності або відсутності хвороб серця. Основні характеристики датафрейму включають *вік, стать, артеріальний тиск, рівень холестерину* тощо. Дані про наявність захворювання серця у пацієнта містяться в стовпці *num*, де *0* – захворювання немає, *1, 2, 3, 4* – є захворювання.



age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0
56.0	1.0	2.0	120.0	236.0	0.0	0.0	178.0	0.0	0.8	1.0	0.0	3.0	0
62.0	0.0	4.0	140.0	268.0	0.0	2.0	160.0	0.0	3.6	3.0	2.0	3.0	3
57.0	0.0	4.0	120.0	354.0	0.0	0.0	163.0	1.0	0.6	1.0	0.0	3.0	0

Рис. 9. Позначення визначень на наборі даних Heart Disease

В одному рядку даних міститься інформація про одну людину (рис. 9). В кожному стовпчику містяться дані про властивості, наприклад про артеріальний тиск. У векторі ознак містяться вхідні дані з яких буде прогнозуватись чи має

пацієнт хвороби серця, чи ні. Все це є матрицею ознак X. Вектор цільових значень y – це значення які ми будемо намагатись передбачити (хворе серце у пацієнта чи ні).

Уявимо що набір даних [14] це плитка шоколаду (рис. 10), де *X* – матриця ознак, *y* – мітка, значення які необхідно передбачити. Кожен рядок даних буде передано в модель, яка зробить прогноз. Після будуть порівняні передбачені значення з фактичними значеннями *y*, які містяться в нашому наборі даних. В цьому є суть керованого навчання. Оцінивши невідповідності між прогнозованими та фактичними даними, ми зможемо переглянути й відповідно змінити модель.

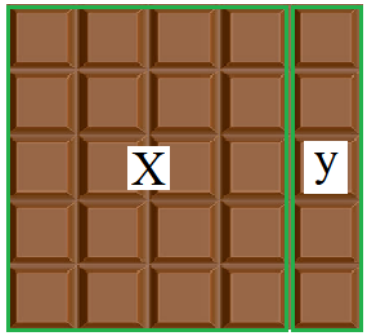


Рис. 10. Представлення набору даних плиткою шоколаду

Поступово ми будемо наближатись до справжніх значень, це є процесом навчання моделі. Виникає питання чи варто взяти всі дані для навчання (цілу плитку шоколаду)? Ні, тому що якщо ми так зробимо, то не будемо знати що наша модель буде коректно опрацьовувати нові дані. Тому розбиваємо весь набір даних на три різні набори даних на навчання, перевірку та тестування (рис. 11).

Рис. 11. Розбиття набору даних на навчання, перевірку та тестування

Залежно від того скільки є статистичних даних часто обирають наступні співвідношення між даними на навчання, перевірку та тестування *60%*, *20%*, *20%* або *80%*, *10%*, *10%*. Ми передаємо обрані для навчання дані в модель й обчислюємо різниці (втрати) між передбаченими та дійсними даними.

В процесі навчання вносяться корективи [15] й виконуються порівняння передбачених даних з даними обраними для перевірки. Це дозволяє після кожного тренування оцінювати втрати. Наприклад, необхідно перевірити якість прогнозування для певних вхідних даних на 4 різних моделях: *моделі A*, *моделі B*, *моделі C* та *моделі D* (рис. 12). З рис. 12 слідує, що *модель C* дає найнижчі втрати *0.5*.

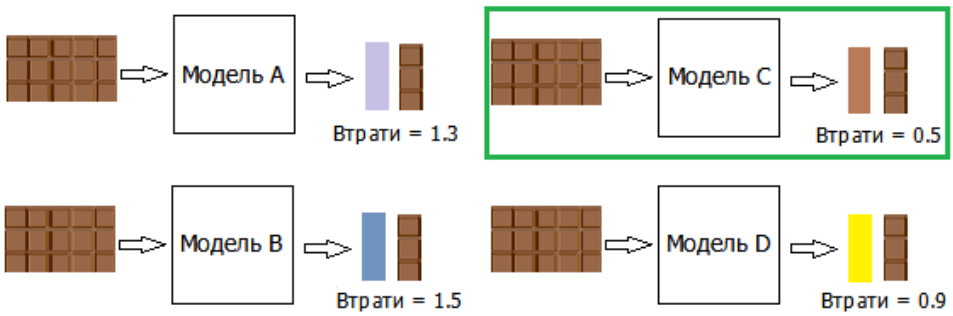


Рис. 12. Рівень втрат для різних моделей

Тестовий набір даних містить дані, які не були використані при навчаннях та перевірках. Тому тестові дані використовуються для остаточної перевірки, щоб побачити наскільки коректною є обрана модель. Отримані втрати свідчать про продуктивність моделі (рис. 13).

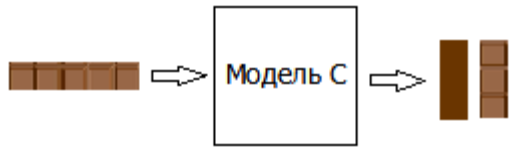


Рис. 13. Перевірка на тестових даних загальної коректності моделі

Втрати можна описати функцією L_1 (найменші абсолютні відхилення), значення якої являються абсолютними значеннями різниць між прогнозованими ($y_{predicted}$) та фактичними даними (y_{true}).

$$L_1 = \sum_{i=1}^{n} \left| y_{true} - y_{predicted} \right| \tag{2}$$

Наприклад, якщо реальне значення відрізняється від прогнозованого на 10 або на -10, то втрати становлять 10.

Також втрати можна описати функцією найменших квадратів L_2, яка є сумою квадратів різниць між прогнозованими та фактичними даними.

$$L_2 = \sum_{i=1}^{n} \left(y_{true} - y_{predicted} \right)^2 \tag{3}$$

Функція втрат L_2 має деякі математичні властивості, які роблять її зручною для обчислень, оскільки вона використовує квадратичні значення, що спрощує деякі аспекти оптимізації. Але якщо в наборі даних присутні великі відхилення, то *функція L_2 не дає добрих результатів, тому що значні квадратичні різниці в результаті дають великі помилки. Тому в таких випадках функція L_1 може бути кращим вибором.*

Також використовують показники точності чи продуктивності. Припустімо, що фотографії яблука та апельсина передаються в модель, а передбаченнями можуть

бути що це або яблуко, або апельсин (табл. 3). З таблиці слідує, що три передбачення правильні, а одне неправильне. *Отже, точність цієї моделі становить три чверті або 75%.*

Таблиця 3

Точності передбачення для фруктів (яблуко, апельсин)

Зображення	Фрукт, який передбачено	Фрукт, який є насправді
	Яблуко	Яблуко
	Апельсин	Апельсин
	Апельсин	Яблуко
	Яблуко	Яблуко

Подальший аналіз датасету Spambase. Продовжимо працювати в Google CoLab з нашим набором даних Spambase (рис. 2). Створимо графічні представлення цих табличних даних та проаналізуємо результати. Код наведений на рис. 14 дозволяє побудувати гістограми, тобто графічні представлення розподілу даних, зображені у вигляді стовпчиків.

```
for label in cols[:-1]:
    plt.hist(df[df["Class"]==1][label], color='red', label='spam', alpha=0.7)
    plt.hist(df[df["Class"]==0][label], color='green', label='non-spam', alpha=0.7)
    plt.title(label)
    plt.ylabel("Count")
    plt.xlabel(label)
    plt.legend()
    plt.show()
```

Рис. 14. Команди для побудови гістограм

Виконавши цей код побудуємо гістограми для кожної властивості набору даних з розділенням на 2 класи: клас *1* для листів які містять спам та клас *0* для листів які не містять спаму. Побудовані гістограми відображають реальні частоти (кількості реєстрацій листів як спам або не спам) в кожному інтервалі (стовпці). Проаналізуємо деякі команди коду рис. 14. *for label in cols[:-1]:* – це цикл, який проходиться по всіх властивостях (колонках) набору даних, за винятком останньої, яка містить мітки класів. Функція *hist* з бібліотеки *Matplotlib* викликається для побудови гістограм для значень класів *1* та *0* (*спам* чи *не спам* відповідно). Параметри *color*, *label* та *alpha* використовуються для налаштування вигляду гістограм. Вираз *plt.title(label)* встановлює заголовок графіку, який відповідає назві поточної властивості, яка зберігається у змінній *label*. *plt.ylabel("Count"), plt.xlabel(label)* встановлюють підписи для осей *y* та *x* відповідно. *plt.ylabel()* та *plt.xlabel()* – це функції з бібліотеки *Matplotlib* у *Python*, які використовуються для встановлення підписів осей *y* та *x* відповідно. Функція *plt.legend()* додає легенду, щоб показати,

який колір прямокутника гістограми відповідає класам *1* та *0*. Функція *plt.show()* відображає побудовану гістограму.

Проаналізуємо деякі гістограми (рис. 15). З першої гістограми слідує, що для властивості *word_freq_credit (відсоток слів credit в електронному листі), якщо є слова credit, то більшість листів буде спамом. Якщо в листах невелика кількість символів "!" (властивість char_freq_! визначає відсоток символів "!" в електронному листі), то більшість листів буде належати до спаму. При зростанні загальної кількості великих літер в електронному листі, яку визначає властивість capital_run_length_total, більшість листів буде містити спам (рис. 15).*

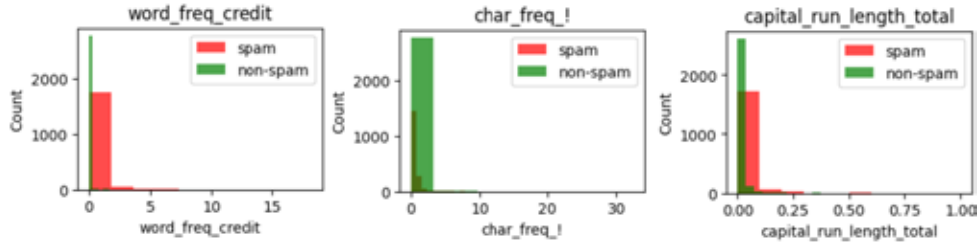


Рис. 15. Гістограми для властивостей word_freq_credit, char_freq_!, capital_run_length_total

Розіб'ємо дані з набору Spambase на дані які будуть використані для навчання, перевірки та тестування (рис. 16). Наступний рядок коду *train, valid, test = np.split(df.sample(frac=1), [int(0.6*len(df)), int(0.8*len(df))])* розділяє набір даних на *train*, *valid*, *test*. *df.sample(frac=1)* випадковим чином перемішує всі рядки у наборі даних *df*. Це забезпечує задання рандомного розміщення даних й дозволяє уникати будь-яких упорядкованих залежностей у вихідному наборі даних. *np.split(…, […])* – це функція *split()* бібліотеки *NumPy*, яка розбиває дані на підмасиви вказаної довжини або розділяє масив за вказаними індексами. Перший аргумент *df.sample(frac=1)* – це випадковим чином перемішані дані з набору *df*. Другий аргумент *[int(0.6*len(df)), int(0.8*len(df))]* – це список, що містить два елементи. Перший елемент *int(0.6*len(df))* вказує на індекс, до якого будуть включені дані для навчального набору. У цьому випадку це *60%* від загальної кількості рядків у наборі даних *df*. Другий елемент *int(0.8*len(df))* вказує на індекс, до якого будуть включені дані в набір для перевірки. У цьому випадку це *80%* від загальної кількості рядків у наборі даних *df*. Решта, тобто дані після другого індексу, будуть використані для тестування. Отже, після виконання цього рядка коду, в змінних *train*, *valid* і *test* міститимуться відповідно навчальний, перевірочний й тестовий набори даних. На навчання виділяється *60%* з усіх даних набору, а на перевірку та тестування – по *20%*.

Визначимо функцію *scale_dataset()* (рис. 16), яка буде виконувати масштабування даних та, в разі потреби, перебалансування класів у наборі даних. У цій функції дані з набору перетворюються в масив *X*, який містить властивості, та масив *y*, який містить вектор цільової змінної (дані в колонці *Class*).

StandardScaler() – це об'єкт, який використовується для масштабування ознак, перетворюючи їх так, щоб вони мали середнє значення *0* і стандартне відхилення *1*. *scaler.fit_transform(X)* – це метод, який застосовує масштабування до матриці ознак *X*.

```
[45] train, valid, test = np.split(df.sample(frac=1), [int(0.6*len(df)), int(0.8*len(df))])

def scale_dataset(dataframe, oversample=False):
    X = dataframe[dataframe.columns[:-1]].values
    y = dataframe[dataframe.columns[-1]].values

    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    if oversample:
      ros = RandomOverSampler()
      X, y = ros.fit_resample(X, y)

    data = np.hstack((X, np.reshape(y, (-1, 1))))

    return data, X, y
```

Рис. 16. Розбиття набору Spambase на дані для навчання, перевірки та тестування. Визначення функції scale_dataset()

При увімкненні коли *oversample=True* функцією *scale_dataset()* буде виконуватись перебалансування класів. Це процес зміни розподілу класів у наборі даних з метою покращення роботи моделі, особливо в умовах нерівності кількостей прикладів між класами. В нашому наборі даних Spambase міститься усього *4601* рядок з даними (рис. 17). На навчання виділяється *60%* з цих даних, тобто це становить *2760* рядків. Як було вказано вище дані на навчання, перевірку та тестування обираються випадковим чином. З рис. 17 слідує, що на навчання було виділено *1098* записів для *спаму* та *1662* записів для *не спамових листів*. Для набору даних на навчання параметр *oversample* встановлено *True*, тому виконано перебалансування класів за допомогою методу випадкової перевибірки (*RandomOverSampler*). Цей метод створює нові екземпляри менш представлених класів або видаляє екземпляри з надлишкових класів для створення збалансованого набору даних. В нашому випадку були створені нові екземпляри для менш представленого класу *листів які містять спам*. Кількість записів зі *спамом* була збільшена з *1098* до *1662* (рис. 17). Дані стали збалансованими. Але для даних, які виділяються на перевірку та тестування не потрібно виконувати балансування (рис. 17, *oversample=False*), тому що необхідно мати випадковий незмінний набір даних, щоб отримати фактичні результати по точності прогнозування.

ros.fit_resample(X, y) – це метод, який застосовує перебалансування класів до матриці ознак *X* та вектору цільової змінної *y*.

np.hstack((X, np.reshape(y, (-1, 1) – функція, яка об'єднує масив *X* (властивості) та масив, який містить цільові змінні *y* у єдиний двовимірний масив. В результаті кожний рядок цього масиву буде представляти один зразок даних, де перші *n* стовпців відповідають ознакам, а останній стовпець – цільовій змінній.

Функція *scale_dataset()* повертає масив *data*, який містить масштабовані та, за потреби, перебалансовані дані. Також ця функція повертає окремо масив *X* і вектор *y*, які містять відповідно масштабовані ознаки та цільову змінну.

```
[47] print(len(df))

     4601

[48] print(len(train[train["Class"]==1]))  # spam
     print(len(train[train["Class"]==0]))  # non-spam

     1098
     1662

[49] train, X_train, y_train = scale_dataset(train, oversample=True)
     valid, X_valid, y_valid = scale_dataset(valid, oversample=False)
     test, X_test, y_test = scale_dataset(test, oversample=False)

[50] len(y_train)

     3324

[51] sum(y_train == 1)

     1662

[52] sum(y_train == 0)

     1662
```

Рис. 17. Перебалансування класів з набору даних призначених на навчання

Висновки. У статті було розглянуто основи машинного навчання та дано основні визначення. В середовище для розробки програмного забезпечення Google CoLab було завантажено набір даних Spambase. Виконано обробку та аналіз даних з використанням мови програмування Python та бібліотек NumPy, Pandas, Matplotlib, Sklearn, Imblearn. З цим набором даних [4, 7] можна проводити більш глибокий аналіз із використанням методів машинного навчання: k-найближчих сусідів, Наївний Баєсів класифікатор, логістична регресія, метод опорних векторів та інші.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. Butakov, N. Exploring machine learning use cases in telecom. Ericsson – Helping to shape a world of communication. Режим доступу: https://www.ericsson.com/en/blog/2021/5/machine-learning-use-cases-in-telecom (дата звернення: 30.03.2024).

2. Tsolaki, K. (2023). Utilizing machine learning on freight transportation and logistics applications: A review. ICT Express, (9), 284–295. Режим доступу: https://doi.org/10.1016/j.icte.2022.02.001 (дата звернення: 30.03.2024).

3. Kuleto, V. (2021). Opportunities and Challenges of Artificial Intelligence and Machine Learning in Higher Education Institutions. Sustainability, (13), 10-24. Режим доступу: https://doi.org/10.3390/su131810424 (дата звернення: 30.03.2024).

4. Харченко, В. О. (2023). Основи машинного навчання : навч. посіб. Суми : Сум. держ. Університет. 264с. Режим доступу: https://essuir.sumdu.edu.ua/bitstreamdownload/123456789/92711/1/Kharchenko_mashynne_navchannia.pdf (дата звернення: 30.03.2024).

5. Могильний, С. Б. (2019). Машинне навчання з використанням мікрокомп'ютерів : Навч.-метод. посіб. Київ, 224 с. Режим доступу: https://api.man.gov.ua/api/assets/man/54c0ee59-b490-4ff3-a346-90a89fd67e30/ (дата звернення: 30.03.2024).

6. Burkov A. (2019). The Hundred-Page Machine Learning Book. p.160.

7. FreeCodeCamp.org. Machine Learning for Everybody – Full Course. (2022). YouTube. Режим доступу: https://www.youtube.com/watch?v=i_LwzRVP7bg (дата звернення: 30.03.2024).

8. Programming with Mosh. Python Machine Learning Tutorial (Data Science). (2020). YouTube. Режим доступу: https://www.youtube.com/watch?v=7eh4d6sabA0 (дата звернення: 30.03.2024).

9. Spambase / M. Hopkins та ін. UCI Machine Learning Repository. Режим доступу: https://archive.ics.uci.edu/dataset/94/spambase (дата звернення: 30.03.2024).

10. McKinney, W. Python for Data Analysis. Sebastopol, California : O'Reilly Media, Inc., 2012. p. 470. Режим доступу: https://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Python-for-Data-Analysis.pdf (дата звернення: 30.03.2024).

11. Лемешко, А.В., Антоненко, А.В., Петрик, А.В. (2023) Нейроморфні системи як інструмент реалізації штучного інтелекту. Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки, 34 (73), (3), 175-183.

12. Антоненко А., Пахомов, М., Калита, Т., Галета, В. (2023). Використання штучного інтелекту в автоматизованих системах. Вісник Хмельницького національного університету, (4), (323), 11-20.

13. Pykes K. Python Machine Learning: Scikit-Learn Tutorial. https://app.datacamp.com. Режим доступу: https://www.datacamp.com/tutorial/machine-learning-python (дата звернення: 30.03.2024).

14. Tamanna. Handling Imbalanced Datasets in Python: Methods and Procedures. Medium. Режим доступу: https://medium.com/@tam.tamanna18/handling-imbalanced-datasets-in-python-methods-and-procedures-7376f99794de (дата звернення: 30.03.2024).

15. Heart Disease / A. Janosi та ін. UCI Machine Learning Repository. Режим доступу: https://archive.ics.uci.edu/dataset/45/heart+disease (дата звернення: 30.03.2024).

REFERENCES:

1. Butakov, N. Exploring machine learning use cases in telecom. Ericsson – Helping to shape a world of communication. URL: https://www.ericsson.com/en/blog/2021/5/machine-learning-use-cases-in-telecom (date of access: 30.03.2024).

2. Tsolaki, K. (2023). Utilizing machine learning on freight transportation and logistics applications: A review. ICT Express, (9), 284–295. URL: https://doi.org/10.1016/j.icte.2022.02.001 (date of access: 30.03.2024).

3. Kuleto, V. (2021). Opportunities and Challenges of Artificial Intelligence and Machine Learning in Higher Education Institutions. Sustainability, (13), 10-24. URL: https://doi.org/10.3390/su131810424 (date of access: 30.03.2024).

4. Kharchenko, V. (2023). Fundamentals of machine learning: tutorial. Sumy : Sumy State University, 264 p. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/92711/1/Kharchenko_mashynne_navchannia.pdf (date of access: 30.03.2024) [in Ukrainian].

5. Mohylnyi, S. (2019). Machine learning with the use of microcomputers: Educational and methodological guide, Kyiv, 224 p. URL:https://api.man.gov.ua/api/assets/man/54c0ee59-b490-4ff3-a346-90a89fd67e30/ (date of access: 30.03.2024) [in Ukrainian].

6. Burkov, A. (2019). The Hundred-Page Machine Learning Book. 160.

7. FreeCodeCamp.org. Machine Learning for Everybody – Full Course. (2022). YouTube. URL: https://www.youtube.com/watch?v=i_LwzRVP7bg (date of access: 30.03.2024).

8. Programming with Mosh. Python Machine Learning Tutorial (Data Science), (2020). YouTube. URL: https://www.youtube.com/watch?v=7eh4d6sabA0 (date of access: 30.03.2024).

9. Hopkins, M. Spambase. UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/dataset/94/spambase (date of access: 30.03.2024).

10. McKinney, W. (2012). Python for Data Analysis. Sebastopol, California : O'Reilly Media, Inc., 470 p. URL: https://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Python-for-Data-Analysis.pdf (date of access: 30.03.2024).

11. Lemeshko, A.V., Antonenko, A.V., Petryk, A.V. (2023) Neiromorfni systemy yak instrument realizatsii shtuchnoho intelektu. Vcheni zapysky TNU imeni V.I. Vernadskoho. Seriia: Tekhnichni nauky. (34), (73), (3), 175-183.

12. Antonenko, A., Pakhomov, M., Kalyta, T., Haleta, V. (2023). Vykorystannia shtuchnoho intelektu v avtomatyzovanykh systemakh. Visnyk Khmelnytskoho natsionalnoho universytetu, (4), (323), 11-20.

13. Pykes, K. Python Machine Learning: Scikit-Learn Tutorial. https://app.datacamp.com. URL: https://www.datacamp.com/tutorial/machine-learning-python (date of access: 30.03.2024).

14. Tamanna. Handling Imbalanced Datasets in Python: Methods and Procedures. Medium. URL: https://medium.com/@tam.tamanna18/handling-imbalanced-datasets-in-python-methods-and-procedures-7376f99794de (date of access: 30.03.2024).

15. Janosi, A. Heart Disease et al. UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/dataset/45/heart+disease (date of access: 30.03.2024).