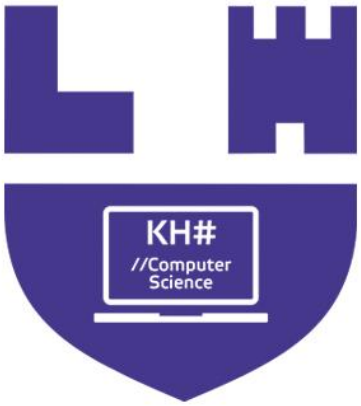
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерних наук



ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

Методичні вказівки до виконання лабораторних
для здобувачів першого (бакалаврського)
рівня вищої освіти
освітньої програми «Комп'ютерні науки»
галузі знань 12 Інформаційні технології
спеціальності 122 Комп'ютерні науки
денної та заочної форм навчання

2025

УДК 004.41

До друку
Голова вченої ради ФКІТ _____І. С. Кондіус

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ
директор бібліотеки _____Н. П. Поліщук

Затверджено вченою радою ФКІТ,
протокол №__ від «__» _____ 2025 року.

Розглянуто та схвалено на засіданні кафедри комп'ютерних наук ЛНТУ,
протокол №__ від «__» _____2025 року.

Завідувач кафедри КН_____В. О. Ліщина

Укладачі:_____В.О. Ліщина, кандидат технічних наук, доцент кафедри комп'ютерних наук ЛНТУ.
_____К.В. Вавринюк, асистент кафедри комп'ютерних наук ЛНТУ.
Рецензент:_____Н. М. Ліщина, кандидат технічних наук, доцент кафедри інженерії програмного забезпечення ЛНТУ

Інтелектуальний аналіз даних: методичні вказівки до виконання лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерні науки» галузі знань 12 Інформаційні технології спеціальності 122 Комп'ютерні науки денної та заочної форм навчання / В.О. Ліщина, К.В. Вавринюк. Луцьк: ЛНТУ. 2025. 49 с.

У методичних вказівках наведені лабораторні роботи з дисципліни «Інтелектуальний аналіз даних». Призначені для студентів спеціальності 122 «Комп'ютерні науки» денної форми навчання.

ЗМІСТ

ВСТУП

Курс «Інтелектуальний аналіз даних» є однією із навчально-професійних дисциплін спеціальності, яка формує знання та навички у сфері аналізу даних, машинного навчання та застосування сучасних інтелектуальних технологій. Вивчення курсу забезпечує практичне ознайомлення з методами обробки інформації, алгоритмами Data Mining, а також з інструментами аналізу, такими як середовище WEKA (Waikato Environment for Knowledge Analysis), що є популярною платформою для вирішення задач інтелектуального аналізу даних.

Головною метою дисципліни є формування у здобувачів освіти практичних навичок застосування теоретичних знань у розв'язанні прикладних задач аналізу даних. Особлива увага приділяється роботі з реальними наборами даних, використанню алгоритмів машинного навчання та оцінці результатів аналізу. Також розглядаються питання інтеграції методів Data Mining у програмні системи, зокрема через виклики алгоритмів з Java-додатків.

Завдання курсу «Інтелектуальний аналіз даних» полягає у забезпеченні теоретичної та практичної підготовки здобувачів вищої освіти до застосування сучасних методів аналізу та обробки даних, засвоєнні принципів побудови моделей та систем обчислювального інтелекту, у тому числі на основі статистичних, нейромережевих і нечітких методів.

Компетентності, які формуються в результаті вивчення дисципліни:

Інтегральна компетентність: здатність вирішувати складні спеціалізовані задачі та практичні проблеми у сфері комп'ютерних наук із застосуванням теорій та методів інформаційних технологій, з урахуванням комплексності та невизначеності умов.

Загальні компетентності:

- уміння застосовувати знання у практичних ситуаціях;
- здатність до пошуку, аналізу та обробки інформації з різноманітних джерел;
- навички прийняття обґрунтованих рішень;
- здатність до самостійного навчання та засвоєння нових знань;
- розвинене абстрактне мислення, здатність до аналізу та синтезу.

Фахові компетентності:

- уміння виявляти статистичні закономірності у даних, працювати з недетермінованими явищами;
- навички застосування методів машинного навчання, обчислювального інтелекту, включно з нейромережевими, нечіткими та генетичними алгоритмами;
- здатність до аналізу великих, погано структурованих даних, їх обробки та візуалізації результатів для підтримки прийняття рішень.

Лабораторна робота № 1
Знайомство з програмою інтелектуального аналізу даних weka та підготовка даних

Мета роботи: ознайомитися та отримати навички роботи з бібліотекою data mining алгоритмів WEKA. На практиці вивчити методи попередньої обробки даних для задач інтелектуального аналізу даних.

Теоретичні відомості

WEKA (Waikato Environment for Knowledge Analysis) – бібліотека алгоритмів машинного навчання для вирішення завдань інтелектуального аналізу даних (data mining). Основні можливості GUI інтерфейсу програми WEKA наведено в додатку А.

Програма дозволяє завантажити та провести попередню обробку даних (Preprocess), вирішити задачу класифікації або регресії (Classify), кластеризації (Cluster), пошуку асоціативних правил (Associate), відбору атрибутів (Select Attributes) та візуалізації (Visualize).

Дані для аналізу в WEKA можуть бути завантажені з файлу, із віддаленого джерела, з бази даних або дані можна згенерувати за допомогою математичної моделі.

Основний формат файлів даних, який використовується в WEKA – ARFF. У каталозі data, який знаходиться в каталозі встановленої програми, можна знайти приклади arff-файлів.

ARFF файл є ASCII текстовим файлом, який описує список об'єктів із загальними ознаками (атрибутами). Структурно такий файл розділяється на дві частини: заголовок і дані [1].

У заголовку описується ім'я даних та їх метадані (імена атрибутів і їх типи), а у другій частині представлені самі дані. Приклад ARFF файлу наведений на рисунку 1.1.

```
@relation 'football-2ndlevel-trainning'

@attribute vectorlengthL0 numeric
@attribute residual16x16 numeric
@attribute MBtypeAVC {0,1,2,3,8,9,10,11}
@attribute meansofvariances4x4 numeric
@attribute varianceofmeans4x4 numeric

@attribute class {0,1}

@data
16.40,1687.00,8,33.80,57.87,1
0.71,1715.00,2,95.19,68.55,1
1.00,132.00,1,1.86,0.73,0
```

Рисунок 1.1 – Приклад ARFF файлу

Заголовок містить інформацію про ім'я файлу і метадані про представлені у ньому даних. Ім'я описується в наступному форматі: @relation <ім’я>.

Іменем може бути будь-яка послідовність символів – @relation weather, але якщо ім'я містить пробіли, то воно має бути взято в лапки – @relation ‘weather nominal’.

Метадані описують атрибути представлених у файлі даних. Інформація про кожний атрибут записується в окремому рядку і включає ім'я атрибуту і його тип. Очевидно, що всі імена повинні бути унікальними. Порядок їх опису повинен збігатися з порядком колонок в описі самих даних. Загальний формат опису атрибуту наступний: @attribute <ім’я атрибута> <тип атрибута> Наприклад: @attribute temperature real.

Поле <тип> може мати одне з таких значень:

− real;
− integer;
− <категорія>;
− string;
− date [<формат дати>].

Типи real і integer є числовими. Категоріальні типи описуються переліком категорій (можливих значень). Наприклад: @attribute outlook {sunny, overcast, rainy}

Дані представляються в ARFF форматі у вигляді списку значень атрибутів об'єктів після тегу @ data. Кожен рядок списку відповідає одному об'єкту, кожна колонка – атрибуту, описаному в заголовку. Часто в термінології data mining такі рядки називають векторами [1].

Дані можуть містити припущення (невідомі) значення. У ARFF вони представляються символом «?» (рис 1.2).

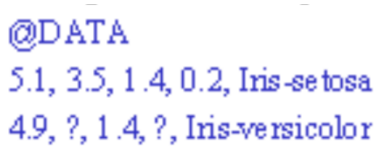


Рисунок 1.2 – Приклад даних з пропущеними значенями

Строкові дані, у разі якщо вони містити символи, що розділяють, повинні братися в лапки, а при описі дати можна вказати формат, в якому вона записується: @attribute timestamp DATE "yyyy-MM-dd HH:mm:ss". Дати також повинні братися в лапки: "2001-04-03 12:12:12".

Кожен набір даних, який використовується в лабораторних роботах, представлений у форматі ARFF. На початку кожного файлу міститься вичерпна інформація про задачу, представлену цим набором даних.

Якість даних є критично важливою характеристикою, яка включає такі

параметри, як точність, повнота, несуперечність, своєчасність, достовірність та інтерпретованість. Щоб підвищити якість даних і забезпечити їхню готовність до аналізу методами інтелектуального аналізу, застосовуються різні технології попередньої обробки.

До основних задач попередньої обробки даних належать: очищення, інтеграція, проріджування, стиснення та перетворення даних.

Очищення даних – включає виявлення та заповнення пропущених значень, усунення шумів і суперечностей, а також виявлення й видалення викидів.

Інтеграція даних – об'єднання інформації з різних джерел (бази даних, куби даних, файли) в одне узгоджене сховище. Передбачає усунення дублікатів, неузгодженостей та конфліктів між даними.

Проріджування та стиснення даних – спрямоване на зменшення обсягу даних з мінімальними втратами інформації. Включає зниження розмірності (відбір релевантних атрибутів) та чисельне стиснення (моделювання значень атрибутів за допомогою математичних моделей).

Перетворення даних – охоплює такі операції, як нормалізація, дискретизація, квантування, згладжування, агрегація та відображення даних за допомогою ядерних функцій. Ці дії забезпечують узгодженість і придатність даних для подальшого аналізу.

Крім того, до попередньої обробки даних можна віднести перетворення задачі багатокласової класифікації у серію бінарних задач, що полегшує побудову моделей у деяких методах машинного навчання.

Відбір атрибутів. У більшості практичних ситуацій набори даних містять занадто багато атрибутів, що збільшує час навчання алгоритмів. При цьому деякі з атрибутів є незначущими чи надмірними. Таким чином дані повинні бути попередньо оброблені з метою відбору деякої мінімальної підмножини атрибутів для навчання.

Для вибору хорошої підмножини атрибутів існує два підходи. Перший з них заснований на незалежній оцінці статистичних чи якихось інших характеристиках набору даних. Він називається фільтрацією і відбувається до початку безпосереднього аналізу даних.

У другому підході відбір підмножини атрибутів виконується всередині методів інтелектуального аналізу. Такий підхід називається методом обгортки (wrapper method), тобто алгоритм навчання «обгорнутий» в процедуру відбору атрибутів.

Методи інтелектуального аналізу даних можуть бути ефективно використані не лише для побудови моделей, але й для відбору інформативних атрибутів. Такий підхід дозволяє зменшити розмірність задачі та підвищити ефективність подальшого аналізу, особливо при використанні інших методів.

Один з найпростіших способів відбору атрибутів – побудова дерева рішень на повному наборі даних. Після побудови можна залишити лише ті атрибути, які

були використані в структурі дерева. Хоча це не обов'язково покращить якість нового дерева, відібрані атрибути можуть бути дуже корисними для інших алгоритмів аналізу.

Ще один підхід полягає у застосуванні алгоритмів, що будують лінійні моделі (наприклад, метод опорних векторів або логістична регресія). У такому випадку атрибути можна ранжувати за абсолютними значеннями коефіцієнтів моделі: атрибути з найменшими вагами можна вважати малозначущими й поступово вилучати. Цей процес можна повторити кілька разів для стабілізації вибору [1].

Існують також підходи, які ґрунтуються на аналізі схожості екземплярів вибірки. При цьому порівнюються сусідні зразки одного класу та різних класів, якщо значення певного атрибута сильно відрізняються у екземплярів одного класу, то цей атрибут, імовірно, незначущий, і його вагу слід зменшити та навпаки, якщо атрибут має різні значення у зразків різних класів, він вважається інформативним, і його вагу слід збільшити. Цю процедуру можна повторювати, поки не буде сформовано остаточний набір атрибутів із найбільшими вагами. До недоліків такого підходу належить те, що методи на основі близькості не виявляють надлишкових атрибутів, які мають високу кореляцію між собою. У таких випадках бажано додатково застосовувати статистичні методи виявлення мультиколінеарності.

Зазвичай пошук в просторі атрибутів відбувається в одному з двох напрямків: зверху вниз (починаючи з повного набору атрибутів і відкидаючи на кожному кроці найгірший з них) або знизу вгору (починаючи з порожньої множини атрибутів і додаючи найкращий з решти) (рис. 1.3).

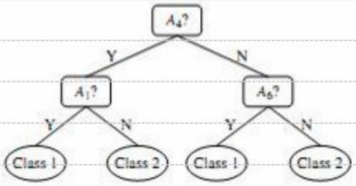
Прямий вибір (forward selection)	**Зворотне виключення (backward elimination)**	**Застосування дерев рішень**
Початкова множина атрибутів {A1, A2, A3, A4, A5}		
{}	{A1, A2, A3, A4, A5}	
=> {A1}	=>{A1, A2, A4, A5}	
=> {A1, A4}	=>{A1, A4, A5}	
=> {A1, A4, A5}		
		=>{A1, A4, A5}

Рисунок 1.3 – Пошук в просторі атрибутів

У деяких випадках для поліпшення точності класифікації та кращого розуміння атрибутів для вирішення поставленого завдання можлива побудова нового атрибуту на основі існуючих.

При роботі з даними, в яких є пропущені значення атрибутів для деяких

екземплярів, існують наступні стратегії поведінки:

- відкинути екземпляри з пропущеними значеннями, такий підхід застосовується насамперед для даних, у яких відсутнє значення цільового атрибута (для задач класифікації).
- заповнити пропущені значення вручну.
- застосувати глобальну константу (наприклад, «unknown»).
- використати деяке статистично розраховане по всій вибірці значення (середнє арифметичне, медіану, моду).
- використати статистичне значення, розраховане для примірників, що відносяться до того ж класу, як і розглянутий екземпляр.
- використати найбільш ймовірне значення для атрибуту, яке може бути розраховане за допомогою регресії, дерева рішень або інших математичних підходів.

Одиниці виміру, що використовуються в деякому атрибуті, можуть вплинути на результати аналізу. Так, наприклад, перетворення одиниць вимірювання з метрів в дюйми для атрибуту «висота» або перетворення з кілограмів у фунти для атрибуту «вага» можуть призвести до різних результатів. У загальному випадку, вираз деякої атрибуту в дрібніших одиницях виміру приведе до більш широкого діапазону значень для цього атрибуту, що може призвести до більшої значущості або ж ваги даного атрибуту.

Щоб уникнути залежності від вибору одиниць вимірювання та надати всім атрибутам однакову вагу дані повинні бути нормалізовані або нормовані. Нормалізація передбачає перетворення даних таким чином, щоб діапазон значень, прийнятих атрибутом, зменшився або став рівним [-1; 1] або [0; 1]. Нормалізація найбільш корисна в задачах з застосуванням нейронних мереж та задачах, алгоритми яких засновані на обчисленні відстаней.

Існує багато методів нормалізації даних, розглянемо деякі з них. Нехай у нас є числовий атрибут A з виміряними значеннями a_1, a_2, …, a_n.

Мінімаксна нормалізація. Цей метод дозволяє перетворити значення атрибутів у заданий діапазон, найчастіше – від 0 до 1. Для цього враховуються мінімальне та максимальне значення атрибуту. У результаті всі значення масштабуються так, щоб залишатись у межах цього діапазону (рис 1.4).

$$a'_i = \frac{a_i - \min A}{\max A - \min A} \cdot (\text{new}_{\max} - \text{new}_{\min}) + \text{new}_{\min}$$

Рисунок 1.4 – Мінімаксна нормалізація

Нормалізація до нульового середнього. У цьому підході значення атрибутів перетворюються так, щоб їх середнє значення стало нулем, а розкиданість значень (варіація) була однаковою. Це досягається за допомогою

статистичних показників – середнього та стандартного відхилення. Метод особливо корисний для алгоритмів, які чутливі до масштабу, зокрема методів, що базуються на відстані (рис 1.5).

$$a'_i = \frac{a_i - \mu_A}{\sigma_A}$$

Рисунок 1.5 – Нормалізація до нульового середнього

Дискретизація числових атрибутів.

Дискретизація числових атрибутів є обов'язковою і необхідною у разі застосування алгоритмів інтелектуального аналізу, що працюють тільки з категоріальними атрибутами. Крім того, алгоритми, що працюють з числовими атрибутами часто дають кращі результати або ж працюють швидше, якщо значення атрибутів попередньо приведені до дискретної форми.

Методи дискретизації можуть бути класифіковані за двома параметрами: чи використовується в них інформація про класи:

- дискретизація з учителем (supervised discretization) або дискретизація без вчителя (unsupervised discretization);
- в якому напрямку відбувається дискретизація: зверху-вниз (дискретизація починається з однієї або декількох точок поділу, а далі отримані інтервали рекурсивно розбиваються; метод розбиття) чи знизу-вгору (спочатку всі значення атрибуту розглядаються як потенційні точки поділу, а далі сусідні значення рекурсивно об'єднуються, утворюючи інтервали; об'єднання).

Вибірка (sampling). Вибірка застосовується в якості методу зменшення початкового набору даних з метою представлення великої вихідної множини екземплярів вибірки набагато меншою за розміром підмножиною.

Припустимо, що вихідний набір даних D містить N примірників.

Розглянемо найбільш загальні шляхи зменшення його розміру.

Проста випадкова вибірка без повернення: з вихідного набору D випадковим чином вибирається S примірників (S<N), при цьому ймовірність вибору кожного примірника рівноймовірна.

Проста випадкова вибірка з поверненням: схожа на попередню, однак з тією відмінністю, що після вибору примірника, він повертається у вихідну вибірку і згодом знову може бути вибраний.

Кластерна вибірка: якщо вихідна вибірка згрупована в деякі роз'єднаним «кластери» (наприклад, сторінки з бази даних або дані з різних географічних джерел), то до кожного з таких кластерів може бути застосована проста випадкова вибірка.

Стратифікована вибірка: якщо вихідна вибірка несиметрична щодо розподілу класів і може бути розділена на страти, то проста випадкова вибірка

застосовується до кожної страти окремо (наприклад, якщо дані представляють відомості про покупців різних вікових груп і при цьому кількість представників різних груп не однакова, такий підхід дозволить не втратити відомості про рідкісні групи покупців).

Завдання для самостійного виконання

- Побудуйте потік даних в модулі Knowledge flow як описано в додатку А.
- В додатку Б оберіть вибірку для аналізу.
- Використовуючи вкладки попередньої обробки даних та візуалізації, проведіть детальний опис вибірки даних. Вкажіть:
  a) яке практичне завдання вирішується; о скільки примірників у вибірці;
  b) атрибути, які характеризують екземпляри вибірки, їхні типи та опис;
  c) чи є екземпляри з відсутніми значеннями, чи є викиди в даних;
  d) який атрибут є цільовим, які значення він приймає, скільки екземплярів кожного класу в вибірці;
  e) подайте дані в графічному вигляді; о наведіть перші 5 екземпляри вибірки.
- Встановіть додатковий пакет scatterPlot3D та візуалізуйте дані.

Контрольні питання

1. Що таке інтелектуальний аналіз даних?
2. Що таке розвідувальний аналіз?
3. Для чого використовується програма WEKA, які її можливості.
4. Яке призначення модулів Explorer, Knowledge Flow, Experimenter, Command-Line Interface?
5. Опишіть формат arff файлу.
6. Опишіть призначення вкладок в модулі Explorer: Preprocess panel, Classify, Cluster, Associate, Select Attributes, Visualize.
7. Що таке генеральна сукупність і вибірка? Якими властивостями повинні володіти дані? Що таке репрезентативна вибірка?
8. Що розуміють під фільтрацією в Weka? В чому різниця між фільтрами атрибутів та фільтрами екземплярів? В чому різниця між unsupervised та supervised фільтрами?
9. Що таке якість даних? Яка мета підготовки даних до аналізу? Які завдання входять в підготовку даних?
10. Який атрибут даних називають цільовим?
11. Що таке значимий та незначимий атрибут? Що таке відбір атрибутів?
12. За допомогою яких фільтрів можна виконати наступні завдання підготовки даних:
    - перетворити тип атрибута;

- нормалізувати значення числового атрибута; о найти та замінити відсутні значення в даних;
- видалити всі екземпляри даних з заданим значенням атрибута;
- створити новий атрибут; о виконати відбір атрибутів;
- знайти викиди в даних;
- створити підвибірку даних.

Лабораторна робота № 2-3
Набори даних. Шкали. Попередня обробка даних

Мета роботи: закріплення знань про проведення попередньої обробки даних. Формування умінь та навичок очищення набору даних, кодування числових та якісних ознак, здійснення перетворення даних та генерації ознак.

Теоретичні відомості

Бібліотека Pandas є однією з ключових бібліотек, яка дозволяє зробити повноцінний процес аналізу даних, а саме:

- завантаження даних. Бібліотека Pandas дозволяє завантажувати дані різних форматів. Бібліотека підтримує завантаження текстових файлів, бінарних, а також є можливість підключення до баз даних і роботи з ними напряму;
- подання даних. Pandas оперує двома основними структурами даних: Series і DataFrame. Series – індексований масив деякого типу: числовий, бінарний або категоріальний. DataFrame – двовимірна структура даних (сукупність Series) або – це таблиця;
- обробка даних. Після завантаження та формування Series або DataFrame виконання фільтрації, індексування, об'єднання різних DataFrame один з одним, є можливість написання власних обчислювальних методів і застосовування їх на DataFrame;
- побудова графіків. За обраними даними з DataFrame можна будувати гістограми значень, можна подивитися як значення в даному стовпці змінюється в часі. Візуалізація можлива завдяки інтеграції Pandas з іншою бібліотекою – Matplotlib. Загалом Pandas інтегрується й з такими бібліотеками, як NumPy або SciPy.

Робота з Series. Series – це одновимірна структура, яка зберігає дані разом з індексами. Індекси можуть бути задані вручну:

```
import pandas as pd
s = pd.Series([1, 2, 3.4], index=['a', 'b', 'c'])
```

або автоматично:

```
d = {'Kyiv': 1000, 'London': 300, 'Barcelona': None}
cities = pd.Series(d)
```

Операції з Series:

- доступ до елементів: cities['Kyiv'], cities[cities < 1000];
- зміна значень: cities['London'] = 500, cities[cities < 1000] = 3;
- арифметичні операції: cities * 3;
- перевірка на пропущені значення: cities.isnull(), cities.notnull().

Робота з DataFrame. DataFrame – це основна структура для роботи з табличними даними. Створити її можна, наприклад, так:

```
import pandas as pd
df = pd.read_csv('citibike.csv')
```

Основні методи:

- df.head(3) – доступ до перших трьох рядків;
- df.tail(3) – доступ до останніх трьох рядків;

- df.shape – розмір (рядки, стовпці);
- df.columns – назви стовпців;
- df.dtypes – типи даних у стовпцях;
- df[['start time', 'start station name']] – вибір окремих стовпців.

Індексація:

- df.iloc[-1] – останній рядок;
- df.iloc[-1, 4] – значення в останньому рядку 5-го стовпця;
- df.loc[1, ['tripduration']] – доступ за міткою індексу.

Фільтрація даних:

- df[(df['tripduration'] < 1000) & (df['usertype'] == 'Subscriber')]

Описова статистика:

- df.describe() – базова статистика для числових стовпців;
- df.describe(include=[np.object]) – статистика для нечислових стовпців.

Аналіз категоріальних змінних:

- df['usertype'].value_counts(normalize=True) – розподіл значень;
- df['gender'].unique() – унікальні значення ознаки.

Кореляція:

- df.corr() – кореляційна матриця між числовими змінними.

Вибірка частини даних:

- df.sample(frac=0.1) – вибірка 10% випадкових рядків.

Збереження результатів:

- df.to_csv('path_to_file.csv') – збереження у CSV-файл.

Методи групування даних. Групування здійснюється методом groupby. Він дозволяє:

- об'єднати рядки за спільною ознакою;
- виконати агрегацію (mean, sum, min тощо);
- застосовувати до кожної групи власні функції.

З метою огляду методів групування та маніпуляцій даних в Pandas розглянемо деяку змінну, що може набувати два або кілька значень, і потрібно визначити всі рядки, в яких зустрічається тільки перше значення даної змінної, потім – тільки друге і т. д. Це можна здійснити за допомогою методу groupby. Після того, як у нас виділилася певна група, з нею можна окремо працювати, аналізувати розподіл різних ознак у даній групі, а також можна порівнювати ці групи між собою [2].

Приклади використання groupby та агрегацій:

- групування за типом користувача: df.groupby(['usertype']).groups;
- перші значення в кожній групі: df.groupby(['usertype']).first();
- середня тривалість поїздки для кожного типу користувача: df.groupby(['usertype'])[['tripduration']].mean();
- групування за кількома ознаками з агрегацією: df.groupby(['usertype']).agg({'tripduration': 'sum', 'starttime': 'first'});
- кілька агрегацій для однієї ознаки: df.groupby(['usertype']).agg({'tripduration': ['sum', 'min'], 'starttime': 'first'});
- власна функція для агрегації: df.groupby(['usertype']).agg({'tripduration': lambda x: max(x) + 1}).

Різноманітні датасети містять декілька таблиць даних. Для поєднання усіх таблиць, з метою аналізу і виявлення зв'язків між відповідними їх елементами, застосовують методи merge і join:

- pd.merge(df1, df2, on='key') – злиття по спільному стовпцю key (аналог INNER JOIN);
- pd.merge(df1, df2, on='key', how='left') – ліве злиття (всі рядки з df1, відповідні з df2);
- pd.merge(df1, df2, on='key', how='right') – праве злиття (всі рядки з df2, відповідні з df1);
- pd.merge(df1, df2, on='key', how='outer') – повне злиття (всі рядки з обох таблиць);
- df1.join(df2, lsuffix='_left', rsuffix='_right') – злиття по індексу;
- pd.merge(df1, df2, left_on='id1', right_on='id2') – злиття по різних назвами стовпців;
- pd.merge(df1, df2, on='key', indicator=True) – додавання стовпця з джерелом рядка (both, left_only, right_only).

Методи перетворення ознак даних.

Метод map для заміни значень у стовпці створюється словник, де ключі – це старі значення, а значення – нові. Наприклад:

```
usertype = {'Customer': 1, 'Subscriber': 2}
df['usertype'].map(usertype).head()
```

Метод apply з лямбда-функцією можна застосувати функцію до всього стовпця: df['tripduration'].apply(lambda x: x / 60).head() # перетворення секунд у хвилини

Метод apply до рядків (axis=1), якщо потрібно працювати з кількома стовпцями одночасно: df.apply(lambda x: x['tripduration'] / 60, axis=1).head()

Застосовуючи власні функцій з apply можна визначити функцію для складніших перетворень:

```
def transform(row):
    if row['usertype'] == 'Customer':
        return 1
    else:
        return 2
df['user_code'] = df.apply(transform, axis=1)
```

Завдання для самостійного виконання:

1. Підготуйте середовище для роботи з даними:
   - відкрийте Google Colab (https://colab.research.google.com) або запустіть Jupyter Notebook локально на своєму комп’ютері.
   - переконайтеся, що у середовищі встановлено бібліотеки Pandas, NumPy
2. На сайті https://www.kaggle.com/datasets підібрати будь-який датасет, що містить числові значення разом із текстовими даними. Із отриманим датасетом виконати всі операції, аналогічні до прикладу з файлом citibike.csv, що наведені в пункті 1.3.2 для роботи з об'єктом pandas.DataFrame.
3. Для обраного датасету за допомогою інструментів бібліотеки Pandas виконати операції:

- групування даних (наприклад, за категоріями або часовими мітками),
- перетворення ознак (наприклад, створення нових колонок, зміна типів, обчислення похідних значень тощо).

4. На сайті https://www.kaggle.com/datasets підібрати три будь-яких датасети, що містять деякі числові значення разом із текстовими даними. Із отриманими датасетами виконати всі операції, передбачені для роботи з кількома таблицями (наприклад, злиття, об’єднання, обробка спільних або зовнішніх ключів).

Контрольні питання

1. Що таке датасет і які відомі файлові формати їхнього подання ви знаєте?
2. Призначення бібліотеки NumPy.
3. Призначення бібліотеки Pandas.

Лабораторна робота № 4-5
Лінійна регресія. Багатофакторна регресія.

Мета роботи: надбання практичних навичок розв'язання задач регресії і класифікації, в лінійних моделях аналітичного аналізу даних за допомогою методів і засобів машинного навчання, реалізованих в бібліотеках Sklearn для ефективного розв'язання широкого кола задач в програмному середовищі Python.

Теоретичні відомості

Лінійна регресія вирішує задачу підходу «навчання з учителем». Це означає, що є певний розмічений набір даних, на яких виконується навчання. Після чого відбувається передбачення на нових даних, які ще невідомі.

«Лінійна регресія» передбачає якусь дійсну змінну, що дозволяє здійснювати моделювання лінійної залежності від заданих ознак. Тобто, існує певна змінна, яка залежить, як очікується, від інших ознак і для якої виконується пошук залежності.

Наприклад, можна спробувати передбачити заробітну плату фахівця залежно від його віку, досвіду, статі тощо. Можна очікувати, що заробітна плата фахівця залежить від віку і досвіду, але не залежить, наприклад, від статі. Цю залежність і потрібно змоделювати.

Проте в реальному світі простір даних є простором набагато більшої розмірності (площини або гіперплощини).

Таким чином, лінійна регресія – це цільова функція, для якої потрібно визначити різні вагові коефіцієнти (характеристика різних ознак), які найкраще описують вихідні дані [2].

Також цільова функція лінійної регресії може бути інтерпретована як зважена сума різних ознак моделі у вигляді вагових коефіцієнтів, що будуть підбиратися в нашій моделі. За значеннями підібраних вагових коефіцієнтів визначається, які з ознак є найбільш важливими.

Узагальненням лінійної регресії є поліноміальна регресія, яка виконує аналогічні функції. У даній цільовій функції аналогічно підбираються вагові коефіцієнти характеристик ознак моделі, проте змінні характеристики ознак можуть бути піднесені у степеневу функцію.

Функціонал якості і градієнтний спуск. При роботі з лінійною регресією використовується підхід навчання «з учителем», де на розмічених даних виконується процес навчання моделі З метою передбачення на нових даних. Для визначення характеристики якості опису даних тренувальної вибірки, на якій відбувається процес навчання, потрібно функція помилки, яку необхідно оптимізувати, Як функція помилки може бути використана «Mean Absolute Error», або середня абсолютна помилка, яка вказує, як далеко середні прогнози лежать від коректних відповідей. Ця залежність достатньо логічна і добре

інтерпретована функція втрат, проте вона не диференційовна, тому не може бути використана, наприклад, для методу градієнтного спуску або методів градієнтної оптимізації взагалі. Тому, зазвичай, використовуються інші функції. Наприклад, функція середньої квадратичної помилки («Mean Squared Error»), з аналогічною функцією, але з усередненням квадратів відстаней передбачень від реальних відповідей. Дана функція диференційовна і дуже часто використовується в методах градієнтного спуску. Однак лінійна регресія з такою функцією втрат може вирішувати проблему й аналітично, без використання градієнтного спуску. Але в просторах великої розмірності доводиться оперувати складними матрицями, й, відповідно, не завжди це працює. Тому дуже часто легше використовувати градієнтні методи, які дають результат практично завжди.

Отже, функція втрат може бути подана поверхнею, по якій потрібно спуститися. Тобто, потрібно мінімізувати функцію, визначити її мінімум. Саме там модель і буде працювати найкраще. Загальновідомо, що градієнт показує напрямок найшвидшого зростання функції, а антиградієнт - напрямок найшвидшого зменшення функції. Тому можна взяти якийсь набір параметрів (вектор) у нашій моделі, визначити вектор антиградієнта і по ньому спуститися. Використовуючи вектор антиградієнта параметри моделі оновлюються до тих пір, поки відбудеться збіжність, або якість нашої моделі не буде влаштовувати задані характеристики. Тобто, послідовно оновлюються параметри моделі, наприклад, вагові коефіцієнти лінійної регресії відповідно до вектора антиградієнта.

Існує величезна кількість модифікацій градієнтного спуску: можна змінювати швидкість ітераційного кроку, можна оперувати підвибірками об'єктів. Наприклад, існує стохастичний градієнтний спуск, який використовує не всю вибірку для ітераційних кроків за антиградієнтом, а тільки один об'єкт, або градієнтний спуск «mini-batch» (набір об'єктів). Усі види модифікацій і методи оптимізації, які використовуються як в лінійній регресії, так і, наприклад, у нейромережах, зазвичай, це певні різновиди градієнтного спуску.

Логістична регресія. У задачах класифікації передбачається якась дискретна відповідь, для чого використовується підхід машинного навчання «з учителем». Тобто, процес навчання відбувається на різних даних, виконується прогноз нових даних і передбачається дискретна відповідь (0 або 1 в разі бінарної класифікації) [2].

Наприклад, можна класифікувати спам або виконувати більш складну багатокласову класифікацію (тварини, рослини тощо). У разі бінарної класифікації можна використовувати лінійну регресію і дивитися на знак отриманого числа. Лінійна регресія передбачає дійсне число, і, наприклад, якщо у нас значення менше нуля, тоді вважається, що це нуль, а якщо більше нуля, тоді це вважається одиниця. Таким чином, для лінійно роздільних класів виконується побудова деякої роздільної поверхні, де усі точки, що лежать зверху,

будуть відноситись до класу «1», а усі точки, що лежать знизу, будуть відноситись до класу «0».

Але, якщо необхідно передбачати ймовірність належності до якогось класу, тоді потрібно використовувати логістичну регресію. Логістична регресія в своїй основі використовує сигмоїд-функцію, в яку відправляється лінійний класифікатор, а сигмоїд-функція, у свою чергу, повертає число від «0» до «1», яке може бути віднесене відповідно до деякої ймовірності. Чим ближче число до одиниці, тим вища ймовірність належності до певного класу. Також можна побудувати деякий класифікатор, вибираючи певне відсікання «decision boundary».

Розв'язання задачі лінійної регресії. Для роботи з лінійними моделями для розв'язання двох задач регресії і класифікації потрібно використати засоби бібліотеки sklearn.

Для початку потрібно імпортувати бібліотеку для візуалізації Matplotlib і seaborn. Також потрібно імпортувати бібліотеку pandas для роботи з бібліотекою sklearn і її модулем datasets. У datasets міститься певний набір стандартних базових датасетів, на яких можна перевіряти роботу моделей, виконувати аналіз і тренування.

```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from seaborn import datasets.
```

Розв'яжемо задачу регресії за допомогою dataset boston, в якому міститься інформація про вартість і характеристики будинків в районах міста Бостон, США. Якщо подивитися на опис даного dataset, то у ньому є різні характеристики в районах міста, і цільова змінна – це середнє значення вартості будинку в одиницях тис. доларів

```
boston = datasets.load _boston()
boston.keys()
print(boston. DESCR[100:1300]).
```

Тобто, є певна цільова змінна MEDV і різні характеристики типу CRIM rate, тобто рівень злочинності в даному районі, вік, кількість кімнат тощо.

Загалом, використовуючи відповідні дані потрібно вміти передбачувати цільове значення, тобто середню вартість будинку. Для початку потрібно візуально проаналізувати дані. Завжди потрібно дивитися на dataset, щоб приблизно уявляти як він виглядає (наявність пропусків, ознак у вигляді простих чисел тощо)

```
boston_df = pd.DataFrame(boston.data, columns=boston.feature_names)
boston_df.head().
```

Оскільки у нас дійсні числа, то можна розв'язувати задачу передбачення

дійсної змінної за допомогою лінійної регресії. Визначаємо середню ціну будинку за допомогою distplot із seaborn

```
plt figure(figsize—(6, 4))
sns.distplot(boston. target)
plt.xlabel('Price (in thousands)')
plt.ylabel('Count')
plt.tight_layout()
plt_show().
```

Середня ціна будинку коливається в районі 20-30 тисяч доларів. У принципі, якщо передбачання відбувається завжди за 20 або 30 тисяч доларів за цим dataset, то, швидше за все, ми вже можемо непогано вгадувати, Оскільки навчання лінійній регресії викликає інтерес, то для цього потрібно імпортувати з модуля liner_model відповідний клас LinearRegression і отримати instance даного класу – модель, яка буде навчатися. Для того, щоб навчити модель у sklearn, потрібно викликати метод fit. Метод fit - це стандартний інтерфейс, який використовується в переважній більшості моделей. У метод fit передаються дані: Boston.data – це набір об'єктів (набір інформації про ознаки); Target - це цільова змінна (набір, список цільових змінних кожного будинку, тобто середня ціна будинку в цій області).

```
from sklearn.linear_model import LinearRegression
linear regression = LinearRegression()
model = linear regression. fit(boston.data, boston.target).
```

Модель пройшла навчання, і стан зберігся в змінній model. Для початку потрібно проаналізувати процес навчання. Відомо, що лінійна регресія «розкидає» вагові коефіцієнти з кожною ознакою для того, щоб оцінити те, який вноситься вклад в передбачення дійсної змінної. Таким чином, якщо перемножити усі вагові коефіцієнти з відповідними ознаками, то буде отримана цільова змінна, тобто середня вартість будинку:

```
feature_weight df = pd. DataFrame(list(zip(boston.feature_names,
model.coef_)))
feature _weight_df.columns = ['Feature', Weight']
print(feature_weight_df).
```

Видно, що є різні вагові коефіцієнти у різних ознак, а саме: великі і маленькі ваги. Оцінювати їх абсолютне значення не варто, тому що їх ніяк не можна нормалізувати. Проте, якщо б їх можна було б привести до якогось одного порядку, можна було б визначити їхній дійсний внесок в передбачувану здатність нашої моделі.

Оскільки лінійна регресія за функцією (4.3) перемножує вагові коефіцієнти з ознаками, то можна виконати передбачення. Причому потрібно не забути вільний коефіцієнт intercept, який завжди є у навченій моделі лінійної регресії:

```
import operator
```

```
first_predicted = sum(map(lambda pair: operator.mul(* pair),
zip(model.coef_. boston.data[0])))
first_predicted += model intercept_
print(first_predicted).
```

Отримано результат – біля 30 тисяч доларів, – що вже ближче до правди, тому що середнє значення дорівнює 20 або 30 тисяч доларів.

Для того, щоб кожен раз вручну не перемножати всі вагові коефіцієнти зі змінними ознак, використовується стандартний метод predict, який викликається у моделі на даних, щоб отримати конкретне передбачення. Даний метод predict автоматизує вищевказану процедуру множення вагових коефіцієнтів. Це також стандартний інтерфейс – в усіх моделях sklearn, крім методу /її, є також метод predict, який дозволяє передбачати значення.

```
predicted = model predicf(boston.data)
print(predicted]:10]).
```

Для того, щоб оцінити якість виконаної роботи (відповідність отриманого передбачення реальним значенням), можна вивести таку табличку:

```
predictions_ground_truth_df =
pd.DataFrame(list(zip(predicted,boston.target)))
predictions_ground_truth_df.columns =[‘Prediction’, ‘Ground truth’]
predictions_ground_truth_df.head()
print(predictions_ground_truth.df).
```

Оскільки табличні значення є менш інформативними, то можна використовувати графічне подання, на якому буде відображатись передбачуване і дійсне значення.

```
plt figure(figsize—(6, 4))
plt.scatter(predicted, boston.target)
plt.xlabel(’ Predicted’)
plt.ylabel(‘Ground truth’)
pleplo([0, 50], [0, 50], color="red")
plt.tight_layour()
plt_show().
```

У дійсності на графіку всі значення мають розміщуватись на лінії під 45°. Більш того, є певна похибка, якийсь очікуваний розкид, тому що лінійна модель (модель лінійної регресії) досить проста і слабка модель.

Завдання для самостійного виконання:

1. На сайті https://www.kaggle.com/datasets або на інших сайтах підібрати будь-який датасет, що містить числові характеристики певних ознак досліджуваного об'єкта (економічні, технічні або соціальні характеристики будь-яких об'єктів статистичних досліджень). Визначіть цільову змінну, яка залежить від певних ознак, описаних в обраному датасеті. Із отриманим

датасетом виконати усі операції розв'язання задачі лінійної регресії (передбачення значення цільової змінної від заданих ознак).

Контрольні питання

1. У чому полягає задача регресії? Наведіть практичний приклад?
2. Чим задача регресії схожа і чим відрізняється від задачі класифікації?
3. Що таке навчання з учителем і без учителя? До якого типу належить завдання регресії?
4. Задача регресії є описовою або прогнозуючою і чому?
5. Опишіть один з розглянутих методів, що вирішують завдання

Лабораторна робота № 6
Задача кластеризації

Мета роботи: На практиці вивчити роботу алгоритмів кластеризації, навчитися інтерпретувати результати їх роботи і вибирати найкращий метод для розв'язуваної прикладної задачі.

Теоретичні відомості

У лабораторній роботі розглядаються наступні методи кластеризації (у дужках наведено назву на WEKA):

- поділяючий метод кластеризації K-середніх (SimpleKMeans);
- ієрархічний метод кластеризації (HierarchicalClusterer)
- імовірнісний метод кластеризації EM (EM)
- ієрархічний метод кластеризації COBWEB (COBWEB)
- метод заснований на щільності розташування об'єктів DBSCAN

Розглянемо параметри налаштування використовуваних алгоритмів кластеризації в WEKA.

Параметри налаштування кластеризатора SimpleKMeans:

- displayStdDevs – відображати значення стандартного відхилення для числових атрибутів і підрахунки для номінальних атрибутів;
- distanceFunction – функція відстані;
- dontReplaceMissingValues – не замінювати пропущені значення середнім значенням або модою;
- maxIterations – максимальна кількість ітерацій алгоритму;
- numClusters – кількість кластерів;
- preserveInstancesOrder – зберігати порядок примірників у вибірці;
- seed – випадковий сид для рандомізації вибірки.

Параметри налаштування кластеризатора Hierarchical:

- distanceFunction – функція відстані;
- distanceIsBranchLength – у дендрограмі висота лінії, що зв'язує кластери, буде показувати відстань між ними;
- linkType – тип зв'язку для розрахунку відстані між двома кластерами;
- numClusters – кількість кластерів;
- printNewick – виводити кластери в форматі Newick.

Параметри налаштування кластеризатора EM:

- displayModelInOldFormat – використовувати старий формат представлення моделі (у випадках великої кількості кластерів);
- maxIterations – максимальна кількість ітерацій алгоритма.;
- minStdDev – мінімальне значення стандартного відхилення;
- numClusters – кількість кластерів (встановити значень -1 для автоматичного вибору кількості кластерів);

- seed – випадковий сід для рандомізації вибірки.

Параметри налаштування кластеризатора DBSCAN:

- database_Type – використовувана база даних;
- database_distanceType – функція відстані;
- epsilon – радіус пошуку;
- minPoints – мінімальна кількість об'єктів усередині радіуса.

Параметри налаштування кластеризатора COBWEB:

- acuity – мінімальне значення стандартного відхилення для числових атрибутів;
- cutoff - встановити поріг до якого відсікати вузли дерева;
- saveInstanceData – зберегти інформацію про примірники для візуалізації;
- seed – випадковий сід для рандомізації вибірки.

Секція «Model and evaluation» містить інформацію про кількісний розподіл екземплярів по кластерах. При цьому буде вказано, скільки об’єктів було кластеризовано (Clustered Instances), а скільки не увійшли в жоден з кластерів (Unclustered instances) [3].

Якщо було обрано опцію «Classes to clusters evaluation» (порівняння попередньої заданих класів з кластерами), то ця секція також буде містити результати оцінки якості кластеризації. Буде вказано, який з побудованих кластерів відповідає якому класу, буде побудовано матрицю помилок та вказана кількість невірно кластеризованих екземплярів.

Завдання на лабораторну роботу

Завдання 1. Виконайте наступні завдання для набору даних ‘bank.arff’.

1. Запустіть алгоритм кластеризації SimpleKMeans, задаючи значення параметра K (кількість кластерів) від 1 до 12 .
2. Запишіть в таблицю значення сум квадратичних помилок, одержуваних при різних значеннях K. Що означає цей параметр і як змінюються його значення?
3. Для значення K=5 укажіть:
   - скільки кластерів було створено;
   - скільки примірників потрапило в кожен з кластерів (вказати кількість і відсоток);
   - скільки ітерацій знадобилося для кластеризації даних; о складіть таблицю з характеристиками центроїдів.
4. Для значення K=5 візуалізуйте результати кластеризації (по осі абсцис відкласти назву (номер) кластера, по осі ординат – номер примірника в кластері) та дайте оцінку отриманим результатам:
   - чи є значна відмінність у значеннях атрибуту «вік» (age) між кластерами?
   - у яких кластерах домінують жінки (female), а в яких чоловіки (male)?
   - що можна сказати про значення атрибута «регіон» (region) у кожному кластері?

- що можна сказати про розкид значень атрибуту «дохід» (income) між кластерами?
- у яких кластерах домінують сімейні люди (married), а в яких холості (unmarried)?
- у якій кластер потрапило найбільше людей з машинами? о у яких кластерах переважають люди з ощадними рахунками (savings accounts)?
- що можна сказати про розкид значень атрибуту «поточний банківський рахунок» (current account) між кластерами?
- що можна сказати про розкид значень атрибуту «іпотека» (mortgage holdings) між кластерами?
- які кластери в основному складаються з людей, які придбали PEP (особистий план купівлі акцій), і які з людей, які не придбали його?

5. Запустіть алгоритм кластеризації EM та оцініть результати.

Завдання 2.Виконайте наступні завдання для набору даних ‘iris.arff’

1. Запустіть алгоритм кластеризації SimpleKMeans з K=3 та оцініть якість кластеризації, порівнюючи кластери з попередньо заданими класами:
   - запишіть значення суми квадратичних помилок, кількість об'єктів в кластерах та характеристики кожного центроїду; о проаналізуйте як співвідносяться кластери та значення цільового атрибуту, скільки екземплярів було віднесено до «невірних» кластерів, який клас виявився «складним» для виділення;
   - візуалізуйте результати, використовуючи різні атрибути для осі ординат (при візуалізації екземпляри, позначені квадратами були віднесені до «невірного» кластеру);
   - визначте, на що впливає параметр «seed» і чому він є важливим при кластеризації методом k-середніх; для цього проведіть експерименти з різними значеннями параметру і порівняйте отримані результати.

Завдання 3. Ієрархічна кластеризація

1. Завантажте набір даних ‘flagdata.arff. Цей файл представляє атрибути прапорів деяких європейських країн. Виконайте наступні завданні:
2. Запустіть алгоритм COBWEB з параметрами C=0,4 (0,35), saveInstanceData = True, cluster mode = Use training set;
   - візуалізуйте отриману дендрограму та запишіть її, вкажіть, які країни потрапили в який кластер;
   - укажіть, що спільного у прапорів, що опинилися в одному кластері.
   - Завантажте набір даних ‘zoo.arff’ і виконайте наступні завдання:
   - оберіть з вибірки частину тварин на власний розсуд (наприклад, ссавців);
   - запустіть алгоритм Hierarchical Clusterer (тип тварини не використовувати в кластеризації, а назву за допомогою фільтру перетворити на рядковий тип – NominalToString);

- проекспериментуйте з налаштуванням алгоритму та візуалізуйте результати його роботи;
- оцініть, чи є логічний сенс в створюваних кластерах.

Завдання 4. Алгоритм DBScan

1. Для використання алгоритму щільнісної кластеризації згенеруйте набор даних за допомогою алгоритму BIRCHCluster. В наборі згенеруйте також флаг класу.
2. За допомогою налаштувань методу DBScan досягніть найкращої кластеризації даних.
3. Кластеризуйте набір даних за допомогою інших алгоритмів. Який з алгоритмів виявився найбільш ефективним?

Контрольні питання

1. У чому полягає задача кластеризації? Наведіть практичний приклад?
2. Що таке навчання з учителем і без учителя? До якого типу належить задача кластеризації?
3. Задача кластеризації є описовою або прогнозуючою і чому?
4. Чим визначається «схожість» об'єктів при вирішенні задачі кластеризації?
5. Що таке однорівнева і ієрархічна кластеризація?
6. Що таке чітка і нечітка кластеризація?
7. Які є підходи до розрахунку відстані між кластерами?
8. Що таке алгомеративна і дівізімна ієрархічна кластеризація?
9. Опишіть один з розглянутих методів, що вирішують завдання кластеризації?
10. Як оцінити якість побудованої моделі для завдання кластеризації?

Лабораторна робота № 7-8
Задача класифікації

Мета роботи: на практиці вивчити роботу алгоритмів класифікації, навчитися інтерпретувати результати роботи класифікаторів і вибирати найкращий метод для вирішення поставленого завдання.

Теоретичні відомості

У лабораторній роботі розглядаються наступні методи класифікації (у дужках наведено назву в програмі WEKA, при цьому перше слово перед крапкою означає тип алгоритму класифікації і також вказує назву папки, в якій знаходиться метод):

- 0R чи Zero Rule класифікація, тобто прогнозування середнього значення для числового класу та моди для категоріального (rules.ZeroR);
- класифікація за одним правилом 1R чи One Rule (rules.OneR);
- покриваючий метод PRISM (rules.Prism);
- наївна Байєсова класифікація (bayes.NaiveBayes);
- методи побудови дерев рішень CART (trees.simpleCart) та C4.5 (trees.J48);
- метод опорних векторів (functions.SMO);
- метод k найближчих сусідів kNN (lazy.IBk).

Для додавання найпростіших методів в WEKA необхідно встановити пакети simpleEducationalLearningSchemes і simpleCART.

Параметри налаштування алгоритму OneR:

− MinBucketSize – використовується для дискретизації числових атрибутів.

Параметри налаштування алгоритму NaiveBayes:

− displayModelInOldFormat – відображення побудованої моделі у старому форматі, що підходить, коли атрибут класу приймає багато значень;

− useKernelEstimator – для оцінки числових атрибутів використовувати оціночну функцію відмінну від нормального розподілу; useSupervisedDiscretization – використовувати дискретизацію з учителем для перетворення числових атрибутів у номінальні.

Параметри налаштування алгоритму J48:

− binarySplits – використовувати бінарний поділ на категоріальних атрибутах для побудови дерев;

− confidenceFactor – довірчий рівень, використовується для відсікання гілок (малі значення - сильніше відсікання);

− minNumObj – мінімальна кількість примірників у листі;

− reducedErrorPruning – який алгоритм відсікання гілок використовувати;

− saveInstanceData – чи зберігати навчальну інформацію для візуалізації;

− subtreeRaising – чи використовувати операцію підняття піддерев при відсіканні

гілок;

− unpruned – чи залишити дерево повним;

− useLaplace – використовувати оціночну функцію Лапласа для підрахунку ймовірностей в листках.

Параметри налаштування алгоритму SMO:

− buildLogisticModels – чи застосовувати логістичні моделі до виходів (для належної оцінки ймовірностей). c – параметр складності C;

− kernel – функція ядра;

− epsilon – параметр точності (не змінювати);

− filterType – чи буде змінена початкова інформація і яким чином (нормалізація або стандартизація);

− toleranceParameter – допустиме відхилення (не змінювати);

Параметри налаштування алгоритму IBk:

− KNN – кількість сусідів;

− crossValidate – чи буде використовуватися для вибору оптимальної кількості сусідів крос-перевірка hold-one-out;

− distanceWeighting – метод вибору вагових коефіцієнтів для відстаней;

− meanSquared – чи використовується середньоквадратична помилка, чи середня абсолютна помилка для крос- перевірки під час вирішення завдання регресії;

− nearestNeighbourSearchAlgorithm – алгоритм пошуку найближчих сусідів;

− windowSize – максимальна кількість примірників, дозволених в навчальному пулі. Додавання додаткових примірників понад цього значення призведе до видалення старих екземплярів. Значення 0 означає відсутність межі

Для алгоритмів Prism, Id3, ZeroR параметрів, що настроюються, немає.

Додаткову інформацію про алгоритми, їх параметри і вимоги до оброблюваних даних можна отримати у вікні налаштувань алгоритмів на панелі «About» в програмі WEKA і в джерелах [5].

Методи оцінки помилок класифікації. Розглянемо параметри оцінки побудованої моделі класифікації. Матриця помилок – це матриця розміру L × L, де L – число класів, ij-й елемент матриці (i – рядок, j – стовпець) дорівнює числу об'єктів з i-го класу, які були віднесені до j-го. Число вірно класифікованих об'єктів дорівнює сумі елементів, що стоять на головній діагоналі.

Результати добре навченого класифікатора покажуть матрицю помилок, в якій найбільші значення стоять на діагоналі матриці, а невеликі значення (в ідеалі нулі) – на інших позиціях.

Розглянемо матрицю помилок для двох класів «так» та «ні» (рис. 8.1) .

		Передбачений клас		
		Так	**Ні**	
Реальний клас	**Так**	істинно позитивні (TP)	хибно негативні (FN)	P=TP+FN
	Ні	хибно позитивні (FP)	істинно негативні (TN)	N=FP+TN
		P'=TP+FP	N'=FN+TN	

Рисунок 8.1 – Матриця помилок для двох класів

На діагоналі матриці знаходяться істинно позитивні (true positive, TP) і істинно негативні (true negative, TN) екземпляри. Примірники, які відносяться до класу «так», але були віднесені класифікатором до класу «ні» називаються хибно негативними (false negative, FN). Примірники, які відносяться до класу «ні», але були віднесені класифікатором до класу «так» називаються хибно позитивними (false positive, FP).

Розглянемо вирази для розрахунку параметрів точності класифікації для кожного з класів. Для випадків класифікації, в яких кількість класів більше двох, для розрахунків приймається, що клас, що розглядається, є класом «так», а всі інші класи об'єднуються та утворюють клас «ні».

Параметр «TP rate» (чутливість, sensitivity) або «recall» (ефективність) розраховується за формулою зображеною на рисунку 8.2.

$$TP\ rate = recall = \frac{TP}{TP+FN} = \frac{TP}{P}.$$

Рисунок 8.2 – Параметр «TP rate»

Для класу, що розглядається, значення цього параметру дорівнює відсотку вірно класифікованих об'єктів класу (отримується діленням діагонального елементу матриці помилок на суму елементів в його рядку). Іншими словами, параметр чутливості показує долю позитивних екземплярів, які були вірно розпізнані.

Параметр «FP rate» розраховується за формулою зображеною на рисунку 8.3.

$$FP\ rate = \frac{FP}{FP+TN} = \frac{FP}{N}.$$

Рисунок 8.3 – Параметр «FP rate»

Його значення дорівнює відсотку об'єктів інших класів, які помилково були занесені в клас, що розглядається (якщо з матриці викреслити рядок класу,

що розглядається, то значення дорівнюватиме сумі елементів стовпця цього класу, поділеній на суму всіх елементів).

Параметр «TN rate» (специфічність, specificity) розраховується за формулою зображеною на рисунку 8.4. та показує частину негативних екземплярів, які були вірно розпізнані.

$$TN\ rate = \frac{TN}{TN+FP} = \frac{TN}{N}.$$

Рисунок 8.4 – Параметр «TN rate» (специфічність, specificity)

Параметр «precision» (точність) дорівнює відсотку вірно класифікованих об’єктів із всіх об’єктів, віднесених алгоритмом до класу, що розглядається (відношення діагонального елемента до суми елементів стовпця).

Параметр «F-measure» – це середнє гармонійне Precision і Recall.

Параметри success rate (accuracy, recognition rate – частка успішних спроб, точність, коефіцієнт розпізнавання) та error rate (misclassification – частка помилок, помилкова класифікація).

Також буває важливо оцінити втрати і виграші (costs and benefits), пов'язані з класифікаційної моделлю. Втрати, пов'язані з хибно позитивними передбаченнями (як наприклад, невірне передбачення, що хворий пацієнт є здоровим), набагато більші, ніж хибно негативні передбачення (здоровий пацієнт віднесений до хворих). В таких випадках ми можемо віддати перевагу одному типу помилки над іншим шляхом призначення їм різних значень втрат, пов’язаних з перейменуванням класів.

Наприклад, розглянемо задачі видачі кредиту банком. Втрати при видачі кредиту неплатнику набагато вище, ніж втрати від невидачі кредиту особі, яка зуміла би виплатити свій борг [5].

Розглянемо тепер проблему нерівномірного розподілу класів, коли важливий для задачі клас є рідкісним в вибірці. Це означає, що розподіл набору даних відображає значну більшість негативних примірників і меншість позитивних примірників.

Наприклад, у задачі розпізнавання шахрайства цікавим для нас класом (позитивним) є клас «шахрайство», який з'являється набагато рідше, ніж негативний клас «не шахрайство». У медичній задачі до такого рідкісного класу може бути віднесений клас «злоякісна пухлина».

Припустимо, що ми навчили класифікатор класифікувати медичний набір даних, в якому цільовим атрибутом є атрибут «злоякісна пухлина», який може приймати значення «так» і «ні».

Параметр точності success rate рівний 97% може показати, що класифікатор досить точний. Однак якщо у всій вибірці було тільки 3%

примірників, що відносяться до злоякісних пухлин? Ясно, що в такому випадку точність розпізнавання в 97% не може бути прийнятною. У цьому випадку класифікатор може правильно розпізнавати більшість негативних екземплярів (не злоякісна) і помилково класифікувати всі позитивні примірники (злоякісна).

Таким чином нам необхідні інші параметри, що дозволяють оцінити наскільки добре класифікатор розпізнає позитивні примірники і негативні. Для цієї задачі можуть бути використані параметри sensitivity та specificity.

Параметри precision і recall також широко використовуються в класифікації. Precision може бути розглянутий як міра точності / влучності (тобто який відсоток примірників віднесених до позитивних такими і є), тоді як recall – це міра повноти (який відсоток позитивних примірників віднесені до позитивних) що кожен екземпляр, який класифікатор відніс до класу C, насправді належить класу C. Однак, це нічого не говорить нам про кількість примірників клас C, які класифікатор неправильно

Існує тенденція зворотного взаємозв'язку між параметрами precision і recall, тобто існує можливість збільшити один параметр за рахунок зменшення іншого. Приміром, наш медичний класифікатор може досягти високого значення параметра precision відносячи всі екземпляри класу злоякісних пухлин до класу злоякісних, але при цьому може мати низьке значення параметра recall відносячи до класу злоякісних також негативні екземпляри. Зазвичай ці два параметри розглядаються сумісно. Альтернативний шлях застосування цих параметрів – це їх об'єднання в одному параметрі F-measure (F1 score або F-score) [6].

Критерії порівняння роботи класифікаторів

На додаток до параметрів, заснованих на точності, класифікатори можуть бути порівняні за такими наступними параметрами.

Швидкість роботи – іншими словами обчислювальні витрати, пов'язані з навчанням і застосуванням даного класифікатора.

Стійкість до помилок, робастність – можливість класифікатора робити правильні передбачення на зашумлених даних або даних з пропущеними значеннями. Даний параметр зазвичай оцінюється за допомогою синтетичних наборів даних із внесеними шумами і втраченими значеннями атрибутів.

Масштабованість – можливість будувати ефективний класифікатор на великих вибірках. Даний параметр оцінюється за допомогою наборів даних, що збільшуються.

Інтерпретованість – рівень розуміння та можливості проникнути в суть даних, які надає класифікатор. Інтерпретованість є суб'єктивним параметром і тому його важко оцінити. Дерева рішень та класифікаційні правила можуть бути легко інтерпретовані, проте їх інтерпретованість зменшується з їх ускладненням.

Розглянемо результати роботи класифікаторів в WEKA (Classifier output).

Секція «Run information» містить наступну інформацію:

- метод класифікації (scheme);
- назва набору даних, на якому проводилося навчання (relation);
- кількість примірників у вихідній вибірці (instances);
- атрибути, що характеризують об'єкти вибірки (attributes);
- відомості про тестову вибірку (test mode).

Секція «Classifier model» містить параметри налаштованого класифікатора і час, затрачений для побудови моделі. Залежно від типу класифікатора дана область буде містити різну інформацію:

- для алгоритмів, що будують правила, будуть відображені отримані правила;
- для байєсівських класифікаторів будуть перераховані розраховані ймовірності для всіх можливих комбінацій атрибут-значення-клас;
- для класифікаторів, заснованих на побудові дерев, відображається текстове представлення отриманого дерева; в дужках навпроти кожного листа вказана кількість примірників, які до нього віднесені; якщо в лист потрапляють екземпляри декількох класів, через слеш буде вказана кількість примірників, які відносяться до домішок;
- для функціональних методів виводяться значення коефіцієнтів побудованої функціональної моделі;
- для методу k найближчих сусідів відображаються налаштування класифікатора.

Секція «Predictions» буде відображена, якщо в налаштуваннях тестування класифікатора обрана опція «Output predictions». У ній для всіх примірників тестової вибірки будуть відображені результати класифікації, отримані за допомогою навченого класифікатора.

Секція оцінки побудованої моделі «Evaluation» містить кілька підпунктів. «Summary» містить загальну статистику роботи класифікатора:

- кількість та відсоток правильно і неправильно класифікованих примірників (Correctly and Incorrectly Classified Instances), загальна кількість примірників (Total Number of Instances);
- параметр Каппа (Kappa statistic);
- статистичні параметри помилки класифікації (Mean absolute error, Root mean squared error, Relative absolute error, Root relative squared error).

«Detailed Accuracy By Class» містить наступні параметри точності класифікації по кожному з класів: TP Rate, FP Rate, Precision, Recall, F-Measure, ROC Area [7].

«Confusion Matrix» містить матрицю помилок.

Завдання для самостійного виконання:

1. Для індивідуального завдання (додаток Б) вирішите задачу класифікації за допомогою всіх розглянутих в лабораторній роботі алгоритмів.
2. Змінюючи параметри налаштування алгоритмів, спробуйте досягти

найліпшої якості навчання класифікаторів.

3. Порівняйте отримані моделі за допомогою модуля Experimenter.

4. В звіті надайте результати роботи кожного алгоритму, його налаштування, а також результати порівняння.

Контрольні питання

1. У чому полягає задача класифікації? Наведіть практичний приклад.
2. Опишіть один із розглянутих методів класифікації.
3. Що таке навчання з учителем і без учителя? До якого типу належить задача класифікації?
4. Задача класифікації є описовою або прогнозуючою і чому?
5. Навіщо потрібні дві вибірки: навчальна і тестова?
6. Які існують підходи для поділу вихідної вибірки на навчальну і тестову?
7. Як оцінити якість побудованої моделі класифікації?
8. Що таке матриця помилок? Як її інтерпретувати?
9. Що означають параметри чутливість, специфічність, точність? Як їх розрахувати?
10. Що таке параметр Каппа? Що він показує?
11. Що таке аналіз втрати-виграші? Навіщо він?
12. Як порівняти роботу двох класифікаторів?
13. Що таке ансамблі класифікаторів і адаптивний бустінг? Для чого вони застосовуються?

Лабораторна робота № 9
Застосування нейронних мереж для розв'язання задач кластеризації та класифікації

Мета роботи: закріплення знань про сутність моделі нейронної мережі. Отримання навичок проведення класифікації шляхом створення та навчання нейромережі.

Теоретична частина

Штучна нейронна мережа (ШНМ) – це математична модель, натхненна структурою та функціонуванням біологічного мозку. Нейронні мережі здатні до навчання на основі вхідних даних і можуть вирішувати різні задачі машинного навчання: класифікацію, регресію, кластеризацію, прогнозування тощо.

Один з найпоширеніших методів кластеризації на основі нейронних мереж – це карти Кохонена або самоорганізовувані карти (SOM). Це неперервна (unsupervised) нейронна мережа, яка автоматично кластеризує вхідні дані за схожістю.

Основні характеристики SOM:

- мережа складається з входу та одного шару нейронів (вихідного шару);
- кожен нейрон у вихідному шарі має вектор ваг, розмір якого дорівнює розмірності вхідного вектора;
- у процесі навчання мережа зменшує відстань між вагами нейронів і поданими вхідними векторами;
- кожен нейрон відповідає певному кластеру.

Етапи роботи SOM:

- ініціалізація ваг нейронів (зазвичай випадкова);
- подання на вхід мережі вектора даних;
- обчислення відстані між вхідним вектором і вагами кожного нейрона;
- вибір нейрона-переможця (найближчого до вхідного вектора);
- коригування ваг нейрона-переможця та його сусідів;
- повторення кроків 2-5 для всіх вхідних векторів протягом кількох епох [8].

У середовищі GNU Octave (відкрита альтернатива MATLAB) є інструменти для створення і навчання нейронних мереж. Зокрема, для побудови мережі Кохонена використовується функція: newc(input_range, num_neurons, learning_rate), де:

- input_range – двовимірна матриця, яка визначає діапазон значень вхідних даних;
- num_neurons – кількість нейронів (кластерів);
- learning_rate – початкова швидкість навчання.

Інші корисні функції:

- train – навчання мережі;

- sim – подання вхідного вектора на мережу (опитування);
- IW{1} – матриця ваг першого шару.

Приклад побудови карти Кохонена для кластеризації штучно згенерованих даних наведено у лістингу 9.1.

Лістинг 9.1 – Функція реєстрації нових користувачів

```
#Параметри генерації точок
X = [0 1; 0 1];
clusters = 5;
points = 5;
std_dev = 0.01;
#Генерація вхідних даних
p = nngenc(X, clusters, points, std_dev);
% Створення нейронної мережі з шаром Кохонена
h = newc([0 1; 0 1], 5, 0.1);
h.trainParam.epochs = 50;
h = init(h);
h = train(h, p);
#Виведення кластерних центрів
w = h.IW{1};
plot(p(1,:), p(2,:), '^r'), grid on, hold on;
plot(w(:,1), w(:,2), 'ob');
xlabel('p(1)');
ylabel('p(2)');
#Опитування мережі новим вектором
A = 0.6;
B = 0.5;
plot(A, B, '^k');
p_test = [A; B];
y = sim(h, p_test)
% Вхідні дані: ріст і вага
p = [175 180 182 175 183 176 183 176 183 176 175 180 178 180 178 182
178 182 179 174 172 179;
     70  75 100  99  42  48  76  72  40  45  92  96  70  69  95  90
79  82  80  50  96  91];
#Створення мережі Кохонена з 3 кластерами
h = newc([0 200; 0 100], 3, 0.1);
h.trainParam.epochs = 500;
h = train(h, p);
#Виведення центрів кластерів
w = h.IW{1};
plot(p(1,:), p(2,:), '^r'), grid on, hold on;
plot(w(:,1), w(:,2), 'ob');
xlabel('Rist');
ylabel('Vaha');
#Тестування нового вхідного вектора
A = 181;
B = 65;
plot(A, B, '+k');
p_test = [A; B];
y = sim(h, p_test)
```

Лістинг 9.1 – кінець лістингу

Завдання для самостійного виконання:

1. Побудуйте нейронну мережу з шаром Кохонена, яка кластеризує згенеровані дані.
2. Побудуйте нейронну мережу, яка кластеризує людей за вагою та зростом.
3. Для обох мереж:
   - виведіть графік з результатами кластеризації;
   - подайте новий вхідний вектор і визначте, до якого кластеру його віднесено;
   - проінтерпретуйте отримані результати.

Лабораторна робота № 10-11
Пошук асоціативних правил

Мета роботи: на практиці вивчити роботу алгоритмів пошуку асоціативних правил і навчитися інтерпретувати результати їх роботи.

Теоретичні відомості

У лабораторній роботі розглядаються два методи пошуку асоціативних правил:

- алгоритм Apriori;
- алгоритм FPGrowth.

Параметри налаштування алгоритмів

Розглянемо параметри налаштування використовуваних алгоритмів пошуку асоціативних правил в WEKA.

Apriori:

- car – пошук класових (зі значенням цільового атрибута в правій частині) або звичайних асоціативних правил;
- classIndex – індекс цільового атрибута. Якщо встановлено значення -1, буде обраний останній атрибут;
- delta – ітеративно зменшувати значення порогу підтримки на дане значення. Зменшення буде відбуватися до тих пір, поки не буде досягнуто мінімальне значення підтримки чи не буде згенеровано задану кількість правил;
- lowerBoundMinSupport – нижня межа порогу підтримки.;
- metricType – встановлює тип метрики, за якою будуть ранжуватися правила (Confidence, Lift, Leverage, Conviction);
- minMetric – мінімальне граничне значення для обраної метрики;
- numRules – кількість правил, які необхідно знайти;
- outputItemSets – чи виводити часті набори;
- removeAllMissingCols – прибирати чи колонки (атрибути) в яких всі значення відсутні;
- significanceLevel – рівень значущості (тільки для достовірності);
- upperBoundMinSupport – верхня межа мінімальної підтримки. Ітеративне зменшення підтримки починається з цього значення [9].

FPGrowth:

- delta – ітеративно зменшувати значення порогу підтримки на дане значення. Зменшення буде відбуватися до тих пір, поки не буде досягнуто мінімальне значення підтримки чи не буде згенеровано задану кількість правил;
- findAllRulesForSupportLevel – знайти всі правила, які задовольняють нижній межі мінімального значення підтримки та мінімальному значенню метрики. Включення цього режиму скасує виконання ітеративного зменшення підтримки для знаходження заданого кількості правил;

- lowerBoundMinSupport - нижня межа порогу підтримки як частка кількості примірників;
- maxNumberOfItems – максимальна кількість примірників у частому наборі; значення -1 означає без обмежень;
- metricType – встановлює тип метрики, за якою будуть ранжуватися правила;
- minMetric – мінімальне граничне значення для метрики;
- numRulesToFind – кількість правил, які необхідно знайти;
- positiveIndex – встановлює індекс бінарного атрибуту, який буде розглядатися як позитивний;
- rulesMustContain – виводити правила, які містять задані об'єкти (список об'єктів, розділених комою);
- transactionsMustContain–для роботи алгоритму використовувати транзакції (примірники), які містять задані об'єкти;
- upperBoundMinSupport – верхня межа мінімальної підтримки. Ітеративне зменшення підтримки починається з цього значення;
- useORForMustContainList – використовувати логічний зв'язку «або» замість «і» для списків обов'язкових елементів у транзакціях і правилах.

Секція «Associator model» містить інформацію про побудовану модель.

Для алгоритму Apriori секція містить значення мінімальної підтримки, мінімальне значення обраної метрики (зазвичай достовірність), кількість циклів алгоритму. Далі розташовані знайдені часті набори даних та знайдені правила.

Для алгоритму FPGrowth буде вказана загальна кількість знайдених правил та задана кількість найліпших з них.

Представимо асоціативне правило у вигляді X => Y.

Для кожного з правил будуть відображені деякі числові параметри: число перед стрілкою вказує кількість екземплярів вибірки, для яких ліва умовна частина правила вірна (NX), а число після стрілки вказує кількість екземплярів вибірки, для яких вірна ліва і права частини правила (NXUY).

Завдання для самостійного виконання

Виконайте наступні завдання для набору даних ‘vote.arff’.

1. Запустіть алгоритм пошуку асоціативних правил Apriori.
2. Яке значення для порогу підтримки було використано в побудованій моделі? Яке значення для порогу достовірності було використано?
3. Запишіть 10 найкращих знайдених правил, вкажіть для них значення підтримки та достовірності.
4. Що позначають числа ліворуч і праворуч від стрілки в знайдених асоціативних правилах?
5. Запустіть алгоритм пошуку асоціативних правил FPGrowth.

6. Порівняйте списки десяти найкращих правил, отриманих двома алгоритмами. Поясніть відмінність в роботі двох алгоритмів.
7. Запустіть алгоритм Apriori задавши значення car = true. Які асоціативні правила були отримані? Що знаходиться в правій частині знайдених асоціативних правил?
8. Вирішіть задачу пошуку асоціативних правил для одного з наборів даних ‘bank-data.csv’, ‘marketbasket.arff’, ‘FoodMart.arff’.
9. Проаналізуйте та поясніть знайдені правила.
10. Застосуйте необхідні фільтри і вирішіть задачу пошуку асоціативних правил. Проаналізуйте знайдені асоціативні правила.
11. Спробуйте відшукати у власному індивідуальному завданні нові шаблони (асоціативні правила) за допомогою двох розглянутих алгоритмів. Згенеруйте також правила, в правій частині яких буде знаходитися ваш цільовий атрибут.

Контрольні питання

1. У чому полягає задача пошуку асоціативних правил? Наведіть практичний приклад?
2. Що таке частий набір?
3. Що таке сильне асоціативне правило?
4. З яких двох кроків складається пошук асоціативних правил?
6. У чому полягає принцип Apriori?
7. Як формуються правила зі знайдених частих наборів?
8. Опишіть алгоритм Apriori
9. Що означають параметри support, confidence, lift, conviction, що застосовуються в алгоритмі Apriori?
10. Опишіть алгоритм ECLAT.
11. Опишіть алгоритм FPGrowth.

Лабораторна робота № 13
Пошук асоціативних правил

Мета роботи: ознайомлення з принципами побудови та навчання самоорганізовуваних карт (Self-Organizing Maps, SOM) як інструменту для кластеризації багатовимірних даних.

Теоретичні відомості

Самоорганізуючі карти можуть використовуватися для вирішення таких задач як моделювання, прогнозування, пошук закономірностей у великих масивах даних, виявлення наборів незалежних ознак і стискування інформації.

Алгоритм функціонування самоорганізуючих карт (Self Organizing Maps - SOM) є одним з варіантів кластеризації багатовимірних векторів - алгоритм проектування із збереженням топологічної подібності. Прикладом таких алгоритмів може служити алгоритм k-найближчих середніх (c-means). Важливою відмінністю алгоритму SOM є те, що в ньому всі нейрони (вузли, центри класів) впорядковані в деяку структуру (зазвичай двовимірну сітку). При цьому, в ході навчання модифікується не лише нейрон-переможець (нейрон карти, який найбільшою мірою відповідає вектору входів і визначає до якого класу відноситься приклад), але і його сусіди, хоча і у меншій мірі. За рахунок цього SOM можна вважати одним з методів проектування багатовимірного простору в простір з нижчою розмірністю. При використанні цього алгоритму, вектора, близькі у вихідному просторі, виявляються поруч і на отриманій карті.

SOM здійснює використання впорядкованої структури нейронів. Зазвичай використовуються одно- і двовимірні сітки. При цьому кожен нейрон є n-мірним вектором-стовпцем, де n визначається розмірністю вхідного простору (розмірністю вхідних векторів). Вживання одно- і двовимірних сіток пов'язане з тим, що виникають проблеми при відображенні просторових структур більшої розмірності (при цьому знову виникають проблеми з пониженням розмірності до двовимірної, уявної на моніторі). Зазвичай, нейрони розташовуються у вузлах двовимірної сітки з прямокутними або шестикутними чарунками. При цьому, як було сказане вище, нейрони також взаємодіють один з одним. Величина цієї взаємодії визначається відстанню між нейронами на карті. При реалізації алгоритму SOM заздалегідь задається конфігурація сітки (прямокутна або шестикутна), а також кількість нейронів в мережі. Деякі джерела рекомендують використовувати максимально можливу кількість нейронів в карті. При цьому початковий радіус навчання (neighborhood в англомовній літературі) в значній мірі впливає на здатність узагальнення за допомогою отриманої карти. У разі, коли кількість вузлів карти перевищує кількість прикладів в навчальній вибірці, успіх використання алгоритму у великій мірі залежить від відповідного вибору початкового радіусу навчання. Проте, у разі, коли розмір карти складає десятки

тисяч нейронів, час, потрібний на навчання карти, зазвичай буває дуже велике для вирішення практичних задач. Таким чином, необхідно досягати допустимий компроміс при виборі кількості вузлів [10].

Підготовка навчальної вибірки. Відмінність в підготовці навчальної вибірки в цьому алгоритмі полягає в тому, що вихідні поля в такій вибірці можуть бути відсутніми зовсім. Навіть якщо в навчальній вибірці будуть присутні вихідні поля, вони не братимуть участь при навчанні нейромережі. Проте вони братимуть участь при відображенні карт. Нормалізація полів тут така ж, як для дерев рішень. Навчальна вибірка налаштовується так само, як і для нейромережі і дерева рішень.

Навчання. Перед початком навчання карти необхідне проініціалізувати вагові коефіцієнти нейронів. Вдало вибраний спосіб ініціалізації може істотно прискорити навчання і привести до здобуття якісніших результатів. Існують три способи ініціації початкових вагів:

- ініціалізація випадковими значеннями, коли всім вагам даються малі випадкові величини;
- ініціалізація прикладами, коли в якості початкових значень задаються значення випадково вибраних прикладів з навчальної вибірки;
- лінійна ініціалізація.

В цьому випадку ваги ініціюються значеннями векторів, лінійно впорядкованих уздовж лінійного підпростору, який проходить між двома головними власними векторами вхідного набору даних.

Візуалізація. Отриману в результаті навчання карту можна представити у вигляді прошарованого пирога, кожен шар якого є розфарбовуванням, породженим однією з компонент вхідних даних. Отриманий набір розфарбовувань може використовуватися для аналізу закономірностей, які є між компонентами набору даних. Після формування карти, отримується набір вузлів, який можна відображати у вигляді двовимірної картинки. При цьому кожному вузлу карти можна поставити у відповідність ділянку на малюнку (чотири або шестикутну), координати якої визначаються координатами відповідного вузла в гратах. Тепер для візуалізації залишається лише визначити колір чарунок цієї картинки. Для цього і використовуються значення компонент. Найпростіший варіант - використання градацій сірого. В цьому випадку чарунки, відповідні вузлам карти, в які попали елементи з мінімальними значеннями компонента або не попало взагалі жодного запису, будуть змальовані чорним кольором, а чарунки, в які попали записи з максимальними значеннями такого компонента, відповідатимуть чарункам білого кольору. В принципі можна використовувати будь-яку градієнтну палітру для розфарбовування. Отримані розфарбовування в сукупності утворюють атлас, що відображає розташування компонент, зв'язки між ними, а також відносне розташування різних значень компонент.

Завдання для самостійного виконання

1. Відповідно до заданого варіанту підготувати необхідні дані у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.
2. Виконати завантаження даних з таблиць MS Excel за допомогою Майстра імпорту відповідно до методики, представленої в розділі «Порядок виконання роботи».
3. Отримати самоорганізуючі карти Кохонена відповідно до методики, представленої в розділі «Порядок виконання роботи».
4. Виконати економічний аналіз різних ситуацій, змінюючи основні дані на картах як у бік збільшення, так і у бік їх зменшення відповідно до методики, представленої в розділі «Порядок виконання роботи».

Варіанти завдань:

1. Оцінка ефективності вкладень в рекламу
2. Оцінка ефективності вкладень в будівництво
3. Оцінка ефективності вкладень в курорти
4. Оцінка ефективності вкладень в промисловість
5. Аналіз ринку ЗМІ
6. Аналіз ринку нерухомістю
7. Аналіз ринку книжкової продукції
8. Аналіз ринку медичних препаратів
9. Аналіз ринку комп'ютерної техніки
10. Аналіз ринку техніки зв'язку
11. Створення профілів клієнтів (розділення на групи) по суспільних інтересах
12. Створення профілів клієнтів (розділення на групи) по спортивних інтересах
13. Створення профілів клієнтів (розділення на групи) по наукових інтересах

Лабораторна робота № 14
Методи та засоби візуалізації результатів аналізу даних

Мета роботи: надбання практичних навичок з візуалізації результатів аналітичного аналізу даних за допомогою бібліотек Matplotlib і Pandas в програмному середовищі Python для ефективного розв'язання широкого кола задач

Теоретичні відомості

Візуалізація даних засобами Pandas і Matplotlib.

Для створення графіків у Pandas використовується вбудований функціонал, заснований на бібліотеці Matplotlib. Спочатку потрібно створити структуру DataFrame з даними, які необхідно візуалізувати.

Щоб графіки відображалися безпосередньо в Jupyter Notebook, використовується спеціальна команда активації режиму виводу зображень у самому блокноті.

Побудова гістограми виконується за допомогою методу hist() для обраної колонки. Цей метод показує, скільки значень потрапляє у визначені діапазони.

Для більш гнучкого керування графіками доцільно використовувати функціональність модуля pyplot, який дозволяє додавати елементи оформлення графіка, змінювати стиль, кольори тощо.

Наприклад:

- для побудови лінійного графіка використовується функція plot();
- для побудови точкової діаграми – scatter(), де можна задати кольори і підписи;
- для додавання заголовка – title();
- для підписів осей – xlabel() та ylabel();
- для додавання легенди – legend();
- для налаштування позначок на вісі X – xticks();
- для виводу графіка – show();
- для збереження зображення у файл – savefig().

Розширена візуалізація засобами Matplotlib.

Для створення кількох графіків одночасно використовується функція subplots(), яка дозволяє створити "полотно" з кількома підграфіками (вікнами).

Ця функція повертає об’єкти фігури та вісей (figure та axes), що дозволяє керувати кожним окремим графіком. Кількість рядків і стовпчиків задається через параметри nrows та ncols, а розмір усього зображення – через figsize [11].

Для кожного підграфіка можна:

- створити окрему лінію графіка;
- задати заголовок;
- налаштувати вигляд.

Щоб уникнути накладання графічних елементів, використовується функція tight_layout(). Зображення також можна зберегти у файл за допомогою функції savefig().

Візуалізація результатів аналізу даних у Pandas.

Для практичного прикладу можна використовувати набір даних про пасажирів «Титаніка». Завантаження даних здійснюється через функцію читання CSV-файлу.

Щоб побудувати:

- гістограму вартості квитків – використовується метод plot.hist() на колонці з цінами;
- оцінку щільності розподілу – метод plot.kde();
- точкову діаграму залежності двох змінних – метод plot.scatter().

Також можливе групування даних за категоріями (наприклад, за класом пасажирів або фактом виживання) за допомогою методу groupby(). Для кожної групи можна будувати окрему лінію на графіку.

Для кастомізації графіків:

- можна задати межі осей через функцію xlim() або ylim();
- додати легенду – через legend();
- задати назву графіка – через set_title() на об’єкті вісі;
- керувати збереженням – через атрибут figure.savefig().

Усі ці методи дозволяють отримати гнучке представлення аналітичної інформації.

Інтерактивна візуалізація засобами Plotly.

Бібліотека Plotly забезпечує створення інтерактивних графіків, які дозволяють масштабувати, наводити курсор на елементи для перегляду деталей, а також вмикати та вимикати серії даних. Вона працює з різними типами графіків – лінійними, точковими, гістограмами, тепловими картами тощо.

Для роботи з Plotly у середовищі Jupyter Notebook використовується модуль plotly.express, який забезпечує простий синтаксис побудови графіків.

Основні функції:

Побудова графіка здійснюється за допомогою функцій з plotly.express, таких як:

- scatter() – для побудови точкових діаграм;
- line() – для лінійних графіків;
- bar() – для стовпчикових діаграм;
- histogram() – для гістограм;
- box() – для коробчастих діаграм;
- pie() – для кругових діаграм.

Параметри функцій дозволяють:

- визначити, які колонки даних відображати по осях X та Y;
- задавати кольорове групування даних;
- встановлювати розмір та прозорість точок;
- обирати назви підписів та легенд.

Відображення графіка здійснюється автоматично після створення об'єкта графіка або через метод show(), якщо потрібна додаткова контрольована візуалізація.

Оформлення:

- можна змінювати заголовок графіка;
- редагувати підписи осей;
- налаштовувати інтерактивні підказки;
- задавати фільтри або анімацію (наприклад, при візуалізації змін у часі).

Plotly інтегрується з Pandas і дозволяє напряму передавати DataFrame до функцій побудови графіків, забезпечуючи простоту використання й високу гнучкість.

Завдання для самостійного виконання:

1. Ознайомлення з базовими інструментами Matplotlib. У середовищі Jupyter Notebook, на основі наведених у теоретичній частині прикладів, опрацювати основні команди візуалізації з бібліотеки Matplotlib у процедурному та об’єктному стилях.
2. Робота з даними у Pandas та їх візуалізація Обрати на сайті https://www.kaggle.com/datasets будь-який набір даних, що містить числові та текстові ознаки. Застосувати до нього методи візуалізації з використанням Pandas.
3. Інтерактивна візуалізація з Plotly. Обрати на Kaggle або іншому відкритому ресурсі датасет із числовими та категоріальними ознаками (наприклад, дані про економічні, соціальні або технічні характеристики). Провести інтерактивну візуалізацію даних з використанням бібліотеки Plotly.

Контрольні питання

1. На основі яких системних додатків працюють вбудовані елементи бібліотеки візуалізації даних Plotly?
2. Яким чином виконується збереження результатів аналізу даних засобами бібліотек Matplotlib і Plotly?
3. Призначення бібліотеки Matplotlib,
4. Призначення бібліотеки Plotly.

Лабораторна робота № 15
Статистичний аналіз текстів (Text Mining). WebMining

Мета роботи: закріплення знань про аналіз текстової інформації та алгоритми роботи з мережевими даними

Теоретичні відомості

У задачах класифікації текстових даних можна виокремити три основні підходи.

Перший – ручна класифікація. Цей підхід не передбачає використання комп'ютерних засобів. Наприклад, у бібліотеці працівник, переглядаючи книги та читаючи анотації, самостійно визначає тематику кожної книги й присвоює відповідні рубрики. Цей спосіб є трудомістким, вимагає високої кваліфікації працівників і потребує багато часу. Через це він непридатний для обробки великої кількості документів у стислі терміни.

Другий підхід – класифікація на основі правил. У цьому випадку класифікація реалізується через створення спеціальних правил – алгоритмів, які визначають, до якого класу належить текст. Такі правила формулюють експерти галузі, які мають глибокі знання з відповідної тематики. Після створення правила автоматизуються засобами програмування.

Переваги:

- автоматизація процесу класифікації;
- відсутність обмежень на кількість документів;
- висока точність (за рахунок експертного формування правил).

Недолік:

- необхідність постійного оновлення та підтримки правил, що потребує залучення фахівців на тривалий час.

Третій підхід – класифікація на основі машинного навчання. Цей підхід заснований на використанні навчальних даних, з яких система самостійно формує правила класифікації.

Навчальні дані – це набір прикладів (документів), що вже віднесені до певних класів. Єдине завдання людини – визначити ці класи для початкових даних. Яскравим прикладом такого підходу є фільтрація спаму в електронній пошті, коли позначені як "спам" повідомлення слугують навчальними прикладами для системи [11].

Застосування алгоритмів обробки текстів (NLP). Алгоритми обробки природної мови (Natural Language Processing, NLP) активно використовуються в багатьох сферах, зокрема:

- пошук інформації (усний і письмовий запит);
- персоналізована онлайн-реклама;
- автоматичний переклад;

- маркетинговий аналіз і налаштування;
- розпізнавання мови, голосові помічники, чат-боти, автоматизована підтримка клієнтів.

Багато з цих технологій базуються на методах глибокого навчання (Deep Learning) – сучасного напряму машинного навчання, що активно розвивається. Сучасні моделі машинного навчання працюють на основі створених уявлень (representations) вхідних ознак і оптимізації ваг моделі.

Завдання для самостійного виконання:

1. Ознайомлення з підходами до класифікації текстових даних. На основі теоретичного матеріалу опрацювати відмінності між ручною, алгоритмічною (на основі правил) та автоматизованою (на основі машинного навчання) класифікацією текстів.
2. Реалізація автоматичної класифікації текстів. Використовуючи бібліотеки Python для обробки текстів (наприклад, scikit-learn, nltk, pandas), реалізувати алгоритм класифікації документів за тематичними категоріями. В якості датасету можна використати відкриті набори даних з Kaggle або інших ресурсів.
3. Кластеризація текстів. Реалізувати програму для автоматичного групування (кластеризації) текстів на основі їхньої семантичної схожості. Застосувати алгоритми кластеризації (наприклад, k-means, DBSCAN) у поєднанні з методами векторизації тексту (TF-IDF, Word2Vec або BERT).

Контрольні запитання:

1. Які існують основні підходи до класифікації текстових документів і які їхні переваги та недоліки?
2. У чому полягає різниця між класифікацією на основі правил та на основі машинного навчання?
3. Що таке навчальна вибірка в задачах текстової класифікації?
4. Яку роль у класифікації відіграє попередня обробка текстових даних (очищення, токенізація, лематизація)?
5. Що таке TF-IDF і як ця міра використовується для представлення тексту у вигляді числового вектору?
6. У чому полягає відмінність між класифікацією та кластеризацією текстових даних?
7. Які задачі можуть вирішуватись за допомогою методів NLP у реальному житті?
8. Яку роль відіграє глибоке навчання в сучасних NLP-системах?

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Viktoriia H., Bohdan D. Інтелектуальний аналіз даних з використанням weka explorer. International scientific and technical conference Information technologies in metallurgy and machine building. 2024. С. 287-289. URL: https://doi.org/10.34185/1991-7848.itmm.2023.01.077 (дата звернення: 08.05.2025).
2. Unsupervised Learning: Association Rules / K. J. Cios et al. Data Mining. Boston, MA. P. 289-306. URL: https://doi.org/10.1007/978-0-387-36795-8_10 (date of access: 09.04.2025).
3. Supervised Learning: Statistical Methods / K. J. Cios et al. Data Mining. Boston, MA. P. 307-379. URL: https://doi.org/10.1007/978-0-387-36795-8_11 (date of access: 08.04.2025).
4. Supervised Learning: Decision Trees, Rule Algorithms, and Their Hybrids / K. J. Cios et al. Data Mining. Boston, MA. P. 381-417. URL: https://doi.org/10.1007/978-0-387-36795-8_12 (date of access: 08.04.2025).
5. Knowledge Graph OLAP / C. G. Schuetz et al. Semantic Web. 2020. P. 1-35. URL: https://doi.org/10.3233/sw-200419 (date of access: 27.04.2025).
6. Data Science / Mr. Vasudev Shahapur et al. International Journal of Advanced Research in Science, Communication and Technology. 2022. P. 487-495. URL: https://doi.org/10.48175/ijarsct-2904 (date of access: 27.04.2025).
7. Classification of Landsat 8 Images Using Convolutional Neural Network Based on Minimum Noise Fraction Transform / V.O. Lishchyna et al. 2024 35th Conference of Open Innovations Association (FRUCT), Tampere, Finland, 24-26 April 2024. 2024. URL: https://doi.org/10.23919/fruct61870.2024.10516385 (date of access: 07.05.2025).
8. Guillod J. Data Science. Python Programming for Mathematics. Boca Raton, 2024. P. 189-219. URL: https://doi.org/10.1201/9781003565451-12 (date of access: 04.05.2025).
9. Jeyaraj R., Pugalendhi G., Paul A. Data Science. Big Data with Hadoop MapReduce. Includes bibliographical references and index., 2020. P. 357-369. URL: https://doi.org/10.1201/9780429321733-7 (date of access: 05.05.2025).
10. Ліщина Н., Ліщина В. Однопрохідний алгоритм аналітичного опису контурів об’єктів. Die wichtigsten vektoren für die entwicklung der wissenschaft im jahr 2020. 2020. URL: https://doi.org/10.36074/24.01.2020.v1.20 (дата звернення: 07.05.2025).
11. Krishna G. G. Multilingual NLP. International Journal of Advanced Engineering and Nano Technology. 2023. Vol. 10, no. 6. P. 9-12. URL: https://doi.org/10.35940/ijaent.e4119.0610623 (date of access: 09.04.2025).

Інтелектуальний аналіз даних: методичні вказівки до виконання лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерні науки» галузі знань 12 Інформаційні технології спеціальності 122 Комп'ютерні науки денної та заочної форм навчання / В.О. Ліщина, К.В. Вавринюк. Луцьк: ЛНТУ. 2025. 49 с.

Комп'ютерний набір і верстка: К. Вавринюк

Підписано до друку 2025 р.
Формат 60х 84/16. Гарнітура Times New Roman.
Папір офсетний 80 г/м2. Друк офсетний.
Ум. друк. арк. 7,5. Обл.-вид. арк. 75.
Наклад 50 прим. Зам. №

Інформаційно-видавничий відділ
Луцького національного технічного університету
43018, Луцьк-18, вул. Львівська,75.
Друк – ІВВ ЛНТУ