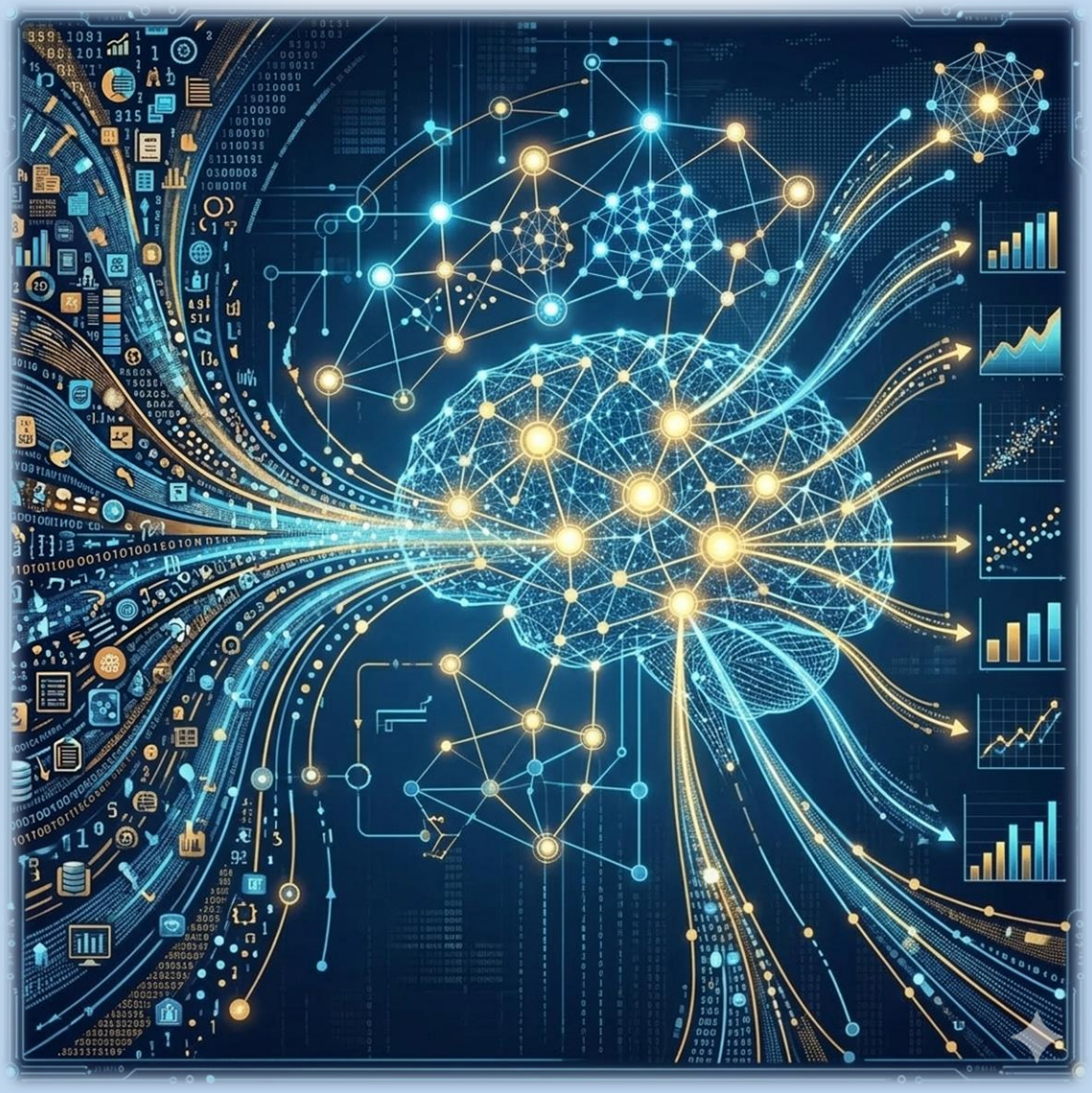
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ МЕНЕДЖМЕНТУ

Кафедра економічної кібернетики,
комп'ютерних наук та інформаційних технологій

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

Методичні рекомендації до лабораторних робіт
для здобувачів першого (бакалаврського) рівня вищої освіти
ОПП «Комп'ютерні науки» спеціальності F3 (122) «Комп'ютерні науки»
денної форми здобуття вищої освіти



Миколаїв – 2025

УДК 004.6
І-73

Друкується за рішенням науково-методичної комісії факультету менеджменту Миколаївського національного аграрного університету (протокол №1 від 28 серпня 2025 року)

Укладачі:

О. Ю. Пархоменко – к.ф.-м.н., доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

С. І. Тищенко – в.о. завідувача кафедри, к.п.н., доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

С. І. Ємельянов – PhD, старший викладач кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О. О. Жебко – асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О. Є. Богатєнкова – асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету.

Рецензенти:

В. В. Базаренко – заступник начальника Миколаївської обласної військової адміністрації з питань цифрового розвитку, цифрових трансформацій і цифровізації (CDTO);

Д. Л. Кошкін – к.т.н., доцент, доцент кафедри електроенергетики, електротехніки та електромеханіки Миколаївського національного аграрного університету.

Інтелектуальний аналіз даних : методичні рекомендації до лабораторних І-73 робіт для здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Комп'ютерні науки» спеціальності F3 (122) «Комп'ютерні науки» денної форми здобуття вищої освіти / уклад. О. Ю. Пархоменко, С. І. Тищенко, С. І. Ємельянов, О. О. Жебко, О. Є. Богатєнкова . Миколаїв: МНАУ, 2025. 59 с.

УДК 004.6

ЗМІСТ

ПЕРЕДМОВА

Сучасний світ неможливо уявити без стрімкого зростання обсягів інформації. Кожну секунду генеруються терабайти даних: транзакції в банках, історії покупок в інтернет-магазинах, показники датчиків, пости в соціальних мережах, медичні картки пацієнтів. Однак самі по собі дані – це лише сировина. Справжню цінність вони набувають лише тоді, коли ми вміємо з них вилучати приховані знання, виявляти неочевидні закономірності та будувати прогнози. Саме цьому присвячений пропонований навчальний посібник з дисципліни «Інтелектуальний аналіз даних».

Сьогодні Data Mining перестав бути екзотичною наукою для вузького кола фахівців – це невід'ємна складова діяльності сучасного IT-спеціаліста. Володіння методами інтелектуального аналізу даних дозволяє не лише обробляти великі масиви інформації, але й приймати обґрунтовані рішення в бізнесі, науці, медицині, маркетингу та багатьох інших сферах. Саме формуванню цих прикладних компетенцій присвячений цей посібник, який ви тримаєте в руках.

Навчальний посібник є логічним продовженням теоретичного курсу з інтелектуального аналізу даних і містить чотирнадцять лабораторних робіт, що охоплюють ключові аспекти діяльності сучасного аналітика даних. Структура посібника побудована за принципом «від простого до складного» – від знайомства з базовими процедурами підготовки даних до створення складних прогностичних моделей та нейронних мереж. Такий підхід дозволяє студентам поступово занурюватися в професію, закріплюючи теоретичні знання через виконання конкретних практичних завдань.

Перші лабораторні роботи закладають фундамент, необхідний для подальшої роботи. Ви ознайомитеся з основними задачами Data Mining, опануєте базові інструменти – мову програмування Python з її потужними бібліотеками та табличний процесор Google Sheets, навчитеся готувати дані до аналізу: нормалізувати, стандартизувати, очищати від шумів, обробляти пропуски та кодувати категоріальні ознаки. Окрема увага приділяється первинному статистичному аналізу, який дозволяє глибоко зрозуміти структуру даних ще до застосування складних алгоритмів.

Наступний блок лабораторних робіт присвячено фундаментальним методам навчання без учителя – кластеризації. Ви детально вивчите ієрархічні алгоритми, навчитеся обирати оптимальні міри близькості та методи зв'язку кластерів, будувати дендрограми та інтерпретувати результати. Окремо розглядаються найпопулярніші алгоритми квадратичної похибки – чіткий k-means та нечіткий c-means, які ви навчитеся застосовувати для аналізу даних.

Значну увагу в посібнику приділено задачам класифікації та прогнозування. Ви познайомитеся з широким спектром методів: від класичного лінійного дискримінантного аналізу до сучасних підходів – методу k-найближчих сусідів, наївного баєсівського класифікатора, дерев рішень, методу

опорних векторів та нейронних мереж. Кожен метод супроводжується практичними прикладами реалізації мовою Python.

Особливе місце посідають завдання з пошуку асоціативних правил, що дозволяють виявляти приховані закономірності в транзакційних даних. Ви детально розберете роботу класичного алгоритму Apriori та навчитеся оцінювати знайдені правила за допомогою підтримки, достовірності, ліфту та інших метрик.

Завершальні лабораторні роботи знайомлять із найпотужнішими інструментами сучасного аналізу – методами виявлення зв'язків між змінними (кореляційний та дисперсійний аналіз), регресійним аналізом для прогнозування та факторним аналізом (методом головних компонент) для зменшення розмірності простору ознак. Ви навчитеся не лише будувати ці моделі, але й оцінювати їх якість та інтерпретувати отримані результати.

Кожна лабораторна робота містить чітко сформульовану мету, короткі теоретичні відомості, деталізовані практичні завдання (частина з яких виконується в Python, частина – в Google Sheets), а також контрольні питання для самоперевірки. Така структура дозволяє використовувати посібник як для аудиторних занять під керівництвом викладача, так і для самостійного опанування дисципліни.

Сподіваємось, що цей посібник стане для вас надійним провідником у світ інтелектуального аналізу даних, допоможе здобути необхідні теоретичні знання та практичні навички, які дозволять вам впевнено почуватися на реальних проектах. Пам'ятайте: аналіз даних – це не просто робота з числами, це мистецтво знаходити істину в хаосі інформації, і саме ви творите світ, у якому технології допомагають людям приймати правильні рішення.

Бажаємо успіхів і цікавих відкриттів!

Лабораторна робота №1
Основи роботи з даними в Python та Google Sheets. Нормалізація та стандартизація

1. Мета роботи

Формування та закріплення практичних навичок первинної обробки даних з використанням мови програмування Python (бібліотеки pandas, numpy, matplotlib) та табличного процесора Google Sheets. Оволодіння методами нормалізації та стандартизації числових даних, їх візуалізації та інтерпретації отриманих результатів.

2. Короткі теоретичні відомості

У задачах інтелектуального аналізу даних якість вхідної інформації безпосередньо визначає якість отриманих результатів. Реальні дані, зібрані з різних джерел, майже ніколи не бувають ідеальними. Вони можуть мати різні масштаби, одиниці вимірювання, містити пропуски або аномальні значення. Тому попередня обробка даних є критично важливим етапом будь-якого дослідження, який, за різними оцінками, займає до 80% часу аналітика.

Двома фундаментальними процедурами підготовки числових даних є нормалізація та стандартизація. Їхня мета – привести всі числові ознаки до порівнянного вигляду, щоб жодна з них не домінувала над іншими через більший діапазон значень. Це особливо важливо для алгоритмів, що базуються на обчисленні відстаней (наприклад, кластеризація, метод k-найближчих сусідів) або використовують градієнтні методи навчання (нейронні мережі).

Нормалізація – це процес перетворення значень ознаки з одного діапазону в інший, найчастіше в інтервал [0, 1] або [-1, 1]. Найпоширенішим методом є min-max нормалізація, яка виконує лінійне масштабування даних. В результаті мінімальне значення ознаки стає 0, максимальне – 1, а всі проміжні значення пропорційно розташовуються між ними.

Стандартизація переслідує дещо іншу мету: привести розподіл ознаки до вигляду з нульовим середнім та одиничним стандартним відхиленням. Найпоширенішим методом є z-стандартизація (або z-перетворення). В результаті цього перетворення одиницею виміру стає одне стандартне відхилення, що дозволяє інтерпретувати значення ознаки як відхилення від середнього.

Для реалізації цих перетворень ми будемо використовувати два інструменти:

Python з бібліотеками pandas (для роботи з табличними даними), numpy (для математичних обчислень) та matplotlib (для візуалізації). Це промисловий стандарт для аналізу даних, який надає максимальну гнучкість та потужність.

Google Sheets – хмарний табличний процесор, який дозволяє швидко виконувати базові розрахунки, будувати графіки та ділитися результатами. Він є чудовим інструментом для швидкого прототипування та візуалізації.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно створити невеликий набір даних, що складається з 15–20 рядків (об'єктів) та 3–4 числових стовпців (ознак). Тематика набору даних обирається самостійно. Наприклад:

- Інформація про студентів (вік, середній бал, кількість пропусків, дохід сім'ї).
- Дані про продажі (ціна товару, кількість продажів, витрати на рекламу, залишок на складі).
- Характеристики автомобілів (потужність двигуна, об'єм двигуна, витрата палива, ціна).

Завдання 1. Підготовка даних та робота в Google Sheets

Створіть нову таблицю в Google Sheets. Сформуйте набір даних відповідно до обраної тематики. Перший рядок має містити назви ознак.

Для кожної числової ознаки за допомогою вбудованих функцій розрахуйте:

- Середнє арифметичне (=AVERAGE())
- Медіану (=MEDIAN())
- Мінімальне та максимальне значення (=MIN(), =MAX())
- Стандартне відхилення (=STDEV())

Створіть нові стовпці для нормалізованих (min-max) та стандартизованих (z-score) значень однієї з ознак на ваш вибір. Використовуючи посилання на комірки з мінімумом, максимумом, середнім та стандартним відхиленням, виконайте відповідні розрахунки для всіх рядків.

Побудуйте два графіки (точкові діаграми) на одному полі, що відображають:

- залежність нормалізованих значень від початкових (x – початкові, y нормалізовані)
- залежність стандартизованих значень від початкових (x – початкові, y стандартизовані)

Додайте легенду, підписи осей та заголовок.

Завдання 2. Робота в Python. Імпорт, аналіз, візуалізація

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot.

З експортуйте створений у Google Sheets набір даних у формат CSV. Завантажте цей файл у середовище виконання (наприклад, у теку з проєктом або в Google Colab). Використовуючи pandas, прочитайте дані з CSV-файлу.

За допомогою методів pandas виведіть перші 5 рядків датафрейму, загальну інформацію про нього (info()), та основні статистичні показники для всіх числових стовпців (describe()).

Створіть окрему функцію min_max_normalize(series) та окрему функцію z_score_standardize(series), які приймають об’єкт pandas.Series та повертають серію з нормалізованими/стандартизованими значеннями. Функції повинні обчислювати необхідні параметри (min, max, mean, std) безпосередньо на основі вхідної серії.

Застосуйте створені функції до всіх числових стовпців вашого набору даних. Збережіть результати в новому датафреймі.

Використовуючи matplotlib, створіть фігуру з двома підграфіками (subplots), розташованими поруч.

На першому підграфіку відобразіть теплову карту (heatmap) початкових даних. Оберіть та налаштуйте кольорову палітру на власний розсуд. Для створення теплової карти в Python використовуйте функцію imshow().

На другому підграфіку відобразіть теплову карту стандартизованих даних.

Додайте заголовки до підграфіків та підписи осей.

Завдання 3. Порівняльний аналіз

У звіті наведіть фрагменти коду створених функцій.

Порівняйте значення нормалізованої та стандартизованої ознаки, отримані в Google Sheets та в Python. Чи збігаються вони? Поясніть можливі причини розбіжностей (якщо вони є).

Проаналізуйте та опишіть, як змінився розподіл даних після стандартизації. Яке тепер середнє значення та стандартне відхилення для стандартизованих стовпців? Чи відповідає це теоретичним очікуванням?

4. Контрольні питання

1. Поясніть, чому нормалізація та стандартизація є важливими етапами підготовки даних. Наведіть приклад, коли їхнє незастосування може призвести до некоректних результатів.
2. У чому полягає принципова різниця між min-max нормалізацією та z-стандартизацією? В яких випадках доцільніше застосовувати кожен з цих методів?
3. Як впливають на результат нормалізації наявні у даних викиди (аномальні значення)? Чому z-стандартизація вважається стійкішою до викидів?
4. Які функції бібліотеки pandas використовувалися для первинного аналізу даних? Що дозволяє дізнатися метод describe()?
5. Як створити власну функцію в Python і застосувати її до стовпця датафрейму pandas?
6. Поясніть, що відображає теплова карта. Яку інформацію про набір даних можна з неї отримати?

7. Чому теплові карти до та після стандартизації можуть виглядати по-різному? Які висновки про структуру даних можна зробити на основі цих відмінностей?
8. Якими формулами в Google Sheets можна розрахувати min-max нормалізацію та z-стандартизацію, використовуючи один раз обчислені статистичні показники?
9. Як експортувати дані з Google Sheets у формат CSV та імпортувати їх у Python за допомогою pandas?
10. Яку інформацію надає спільний графік, що поєднує початкові, нормалізовані та стандартизовані значення? Про що може свідчити лінійний характер залежності на цьому графіку?

Лабораторна робота №2
Аналіз типів даних, шкал вимірювання та попередня обробка даних у Python та Google Sheets

1. Мета роботи

Формування практичних навичок ідентифікації типів даних та шкал вимірювання, виконання очищення даних, обробки пропусків та викидів, кодування категоріальних ознак (зокрема, dummy-кодування) та створення похідних ознак з використанням мови програмування Python (бібліотеки pandas, numpy) та табличного процесора Google Sheets.

2. Короткі теоретичні відомості

Будь-яке дослідження в галузі інтелектуального аналізу даних починається з наявності набору даних, який найзручніше представити у вигляді двовимірної таблиці "об’єкт-ознака". Рядки цієї таблиці відповідають окремим об’єктам (наприклад, клієнтам, товарам, результатам спостережень), а стовпці – їхнім характеристикам, або атрибутам (ознакам, змінним).

Критично важливим для коректного аналізу є розуміння типу шкали, за якою виміряна та чи інша ознака. Це визначає, які математичні та логічні операції можна з нею виконувати. Виділяють:

Номінальна шкала: значення є лише мітками категорій, їх не можна впорядкувати (наприклад, стать, місто проживання, колір очей).

Порядкова шкала: значення можна впорядкувати, але відстань між ними не визначена (наприклад, рівень освіти, військове звання, оцінки за шкалою "погано/добре/відмінно").

Інтервальна шкала: має впорядковані значення з визначеною відстанню між ними, але нуль є умовним (наприклад, температура за Цельсієм).

Шкала відношень: має всі властивості інтервальної, але нуль є абсолютним і означає відсутність ознаки (наприклад, вік, дохід, зріст).

Реальні дані майже ніколи не бувають ідеальними. Вони можуть містити:

- Пропуски: відсутні значення в деяких комірках.
- Викиди: значення, які різко відрізняються від основної маси даних.
- Помилки та суперечності: логічно некоректні значення.

Тому етап попередньої обробки даних (Data Preparation) є обов'язковим. Він включає очищення даних, обробку пропусків та викидів, а також перетворення даних у формат, придатний для аналізу. Одним з ключових перетворень є кодування категоріальних змінних, оскільки більшість алгоритмів машинного навчання працюють лише з числами. Найпоширенішим методом є dummy-кодування, яке створює для кожної категорії окрему бінарну ознаку.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно самостійно обрати або створити набір даних, що містить не менше 8–10 рядків (об'єктів) та 5–6 стовпців (ознак), серед яких мають бути присутні числові та категоріальні змінні. Можна скористатися невеликим фрагментом будь-якого відкритого набору даних (наприклад, з платформи Kaggle) або створити власний (наприклад, інформація про студентів, працівників, автомобілі, фільми тощо). У набір даних слід свідомо внести декілька проблемних значень (пропуски, викиди, логічні помилки) для відпрацювання навичок очищення.

Завдання 1. Робота в Google Sheets. Ідентифікація та первинне очищення

Створіть нову таблицю в Google Sheets та внесіть до неї ваш набір даних. Перший рядок має містити назви ознак.

Для кожного стовпця визначте тип даних (числові/категоріальні) та тип шкали (номінальна, порядкова, інтервальна, відношень). Створіть окремий аркуш "Опис даних", де у вигляді невеликої таблиці зафіксуйте цю інформацію для кожної ознаки.

Проведіть візуальний аналіз таблиці. Знайдіть свідомо внесені проблеми (пропуски, логічні помилки, підозрілі значення). Створіть ще один аркуш "Проблеми та рішення", де опишіть кожну знайдену проблему та запропонуйте спосіб її вирішення (наприклад: "замінити на середнє", "видалити рядок", "замінити на моду").

За допомогою формул Google Sheets реалізуйте обробку пропусків для однієї числової та однієї категоріальної ознаки згідно з вашим планом. Для числової ознаки використайте функції =AVERAGE(), =MEDIAN() або =STDEV() для виявлення викидів; для категоріальної – =MODE(). Для виявлення викидів у числовому стовпці розрахуйте перший (Q1) та третій (Q3) квартилі за допомогою функцій =QUARTILE(), обчисліть міжквартильний розмах IQR та визначте межі для "нормальних" значень.

Завдання 2. Робота в Python. Просунуте очищення та кодування

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте бібліотеки pandas, numpy.

Експортуйте ваш набір даних з Google Sheets у формат CSV. Завантажте файл у середовище виконання Python та прочитайте його за допомогою pandas.

Виведіть перші 5 рядків датафрейму, загальну інформацію про нього (info()), а також основні статистичні показники для числових стовпців (describe()). На основі виведеної інформації (info()) перелічіть стовпці, які містять пропуски.

Для кожного числового стовпця визначте наявність викидів за допомогою правила міжквартильного розмаху (IQR). Виведіть на екран кількість потенційних викидів для кожного стовпця та самі значення-викиди. Прийміть рішення про їх подальшу долю (заміна на верхню/нижню межу, видалення, залишення) та реалізуйте його у коді.

Виконайте dummy-кодування всіх категоріальних стовпців вашого набору даних за допомогою функції pd.get_dummies(). Виведіть на екран отриманий датафрейм. Зверніть увагу, як змінилася кількість стовпців.

Створіть хоча б одну нову (похідну) ознаку на основі наявних. Наприклад, якщо є стовпці "вік" та "дохід", можна створити ознаку "дохід на рік віку" (дохід/вік) або "вікова категорія" (молодий, середній, старший). Додайте цю нову ознаку до датафрейму.

Збережіть очищений та перетворений датафрейм у новий CSV-файл з назвою, яка вказує на те, що дані пройшли попередню обробку (наприклад, data_cleaned.csv).

Завдання 3. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду, які використовувалися для виявлення викидів та dummy-кодування.

Опишіть, чому для кожної з категоріальних ознак було застосовано саме dummy-кодування, а не інший метод. Чи є серед ваших ознак такі, для яких dummy-кодування було б неефективним через велику кількість унікальних значень? Запропонуйте альтернативу.

Проаналізуйте, як змінився набір даних після створення похідної ознаки. Яку нову інформацію вона несе? Чи може вона бути корисною для подальшого аналізу (наприклад, для кластеризації або класифікації)?

4. Контрольні питання

1. Назвіть основні типи шкал вимірювання. Наведіть приклади ознак для кожної шкали та поясніть, чому ви обрали саме цю шкалу.
2. Чим відрізняються поняття "об’єкт", "атрибут" та "змінна" в контексті набору даних?
3. Які основні проблеми можуть бути виявлені в "сирих" даних? Охарактеризуйте кожну з них.

4. Поясніть суть правила міжквартильного розмаху (IQR) для виявлення викидів. Чому воно вважається стійкішим до викидів, ніж правило, засноване на середньому та стандартному відхиленні?
5. Які існують стратегії обробки пропущених значень? В яких випадках доцільно застосовувати кожну з них?
6. Що таке dummy-кодування і для чого воно використовується? Як створити dummy-змінні за допомогою бібліотеки pandas?
7. У чому полягає проблема "прокляття розмірності" при застосуванні dummy-кодування до ознак з великою кількістю категорій? Які існують способи її вирішення?
8. Як за допомогою функцій Google Sheets (=AVERAGE(), =MEDIAN(), =QUARTILE(), =MODE()) можна провести первинний аналіз даних та виявити проблемні значення?
9. Для чого виконується генерація нових ознак (feature engineering)? Наведіть приклад, як нова ознака може покращити якість моделі.
10. Яку інформацію надають методи info() та describe() бібліотеки pandas? Що можна дізнатися про набір даних за допомогою цих методів?

Лабораторна робота №3
Первинний статистичний аналіз даних у Python та Google Sheets

1. Мета роботи

Формування практичних навичок проведення первинного статистичного аналізу даних з використанням мови програмування Python (бібліотеки pandas, numpy, matplotlib, scipy.stats) та табличного процесора Google Sheets. Оволодіння методами побудови варіаційних рядів, обчислення основних статистичних характеристик, візуалізації розподілів, перевірки статистичних гіпотез та оцінки параметрів розподілу.

2. Короткі теоретичні відомості

Первинний статистичний аналіз є фундаментом будь-якого дослідження даних. Його мета – отримати узагальнене уявлення про набір даних, виявити його основні властивості, структуру та приховані закономірності ще до застосування складних алгоритмів машинного навчання.

Вихідні дані, отримані в результаті спостережень, зазвичай являють собою невпорядкований набір чисел. Для зручності аналізу на його основі будують варіаційний ряд – послідовність значень ознаки, розміщених у порядку зростання, де кожне унікальне значення зустрічається лише один раз. Таке унікальне значення називається варіантою. Кількість появи варіанти у вихідній вибірці називається частотою, а відношення частоти до загального обсягу вибірки – відносною частотою.

Залежно від типу даних розрізняють дискретні (для ознак з обмеженою кількістю значень) та інтервальні (для неперервних ознак) варіаційні ряди. Для візуалізації розподілу використовують гістограми (для інтервальних рядів) та полігони частот (для дискретних).

Для кількісного опису вибірки використовують числові характеристики, які поділяються на:

Міри центральної тенденції: середнє арифметичне, медіана, мода. Вони показують "типове" значення ознаки.

Міри мінливості (розсіювання): дисперсія, стандартне відхилення, розмах варіації, міжквартильний розмах. Вони показують, наскільки сильно значення розкидані навколо центру.

Оскільки ми майже ніколи не працюємо з усією генеральною сукупністю, а лише з її частиною – вибіркою, обчислені характеристики є лише оцінками істинних параметрів. Розрізняють точкові (одне число) та інтервальні (інтервал, який із заданою ймовірністю накриває істинне значення) оцінки. Наприклад, довірчий інтервал для середнього показує діапазон, у якому з певною ймовірністю (наприклад, 95%) знаходиться справжнє середнє значення генеральної сукупності.

Для перевірки припущень про властивості розподілу формулюють статистичні гіпотези. Наприклад, гіпотезу про те, що розподіл ознаки відповідає нормальному закону. Рішення про прийняття або відхилення гіпотези приймають на основі статистичних критеріїв (наприклад, критерію згоди Пірсона хі-квадрат), порівнюючи розраховане за вибіркою емпіричне значення з критичним.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно самостійно обрати або створити набір даних, що містить не менше 50–100 значень однієї числової ознаки. Можна скористатися відкритими джерелами (наприклад, Kaggle) або згенерувати випадкові дані за допомогою Python. Приклади: дані про зріст або вагу людей, час виконання завдання, розмір заробітної плати, температура повітря тощо.

Завдання 1. Робота в Google Sheets. Побудова варіаційного ряду та візуалізація

Створіть нову таблицю в Google Sheets. Внесіть ваші дані (50–100 чисел) в один стовпець. Перший рядок має містити назву ознаки.

Побудуйте дискретний варіаційний ряд, якщо кількість унікальних значень невелика, або інтервальний варіаційний ряд, якщо значення неперервні. Для інтервального ряду:

Визначте кількість інтервалів (наприклад, за формулою Стерджеса: 1 + 3.322 * log10(N), де N – обсяг вибірки).

Обчисліть мінімальне та максимальне значення, а потім крок інтервалу.

За допомогою функції =FREQUENCY() (або комбінації COUNTIFS()) підрахуйте частоти для кожного інтервалу. Зверніть увагу, що FREQUENCY є функцією масиву.

На основі побудованого варіаційного ряду створіть гістограму за допомогою інструменту "Діаграма" (тип діаграми – "Стовпчаста діаграма"). Відформатуйте її: додайте заголовок, підписи осей, налаштуйте кольори.

Обчисліть основні статистичні показники для вашого набору даних за допомогою вбудованих функцій:

Середнє арифметичне (=AVERAGE())

Медіану (=MEDIAN())

Моду (=MODE())

Мінімум та максимум (=MIN(), =MAX())

Дисперсію (=VAR.S() – виправлена вибіркова)

Стандартне відхилення (=STDEV.S())

Квартилі (=QUARTILE())

Завдання 2. Робота в Python. Поглиблений статистичний аналіз

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, scipy.stats.

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) або згенеруйте його безпосередньо в Python за допомогою numpy.random.

Виведіть перші 10 рядків, загальну інформацію про серію даних (якщо використовуєте pandas.Series) та основні статистичні показники за допомогою методу .describe().

Створіть полігон частот. Для цього:

Побудуйте гістограму з певною кількістю інтервалів (bins).

Отримайте значення частот та межі інтервалів з об’єкта гістограми.

Обчисліть середини інтервалів.

Побудуйте лінійний графік, де вісь X – середини інтервалів, а вісь Y – частоти. Накладіть цей графік поверх гістограми.

Розрахуйте точкові оцінки параметрів розподілу: середнє, медіану, моду, дисперсію, стандартне відхилення. Порівняйте їх зі значеннями, отриманими в Google Sheets.

Розрахуйте довірчий інтервал для середнього з надійністю 95%. Для цього можна скористатися функцією scipy.stats.t.interval() або scipy.stats.norm.interval() (якщо відомо, що розподіл нормальний і обсяг вибірки великий). Виведіть на екран межі інтервалу.

Перевірте гіпотезу про відповідність розподілу нормальному закону за допомогою критерію згоди Пірсона (хі-квадрат). Використовуйте функцію scipy.stats.chisquare(). Для цього:

Вам знадобляться емпіричні частоти (з гістограми) та теоретичні частоти, розраховані для нормального розподілу з вашими середнім та стандартним відхиленням.

Порівняйте отримане p-value з рівнем значущості $\alpha = 0.05$. Зробіть висновок про те, чи можна вважати розподіл нормальним.

Завдання 3. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду для побудови полігону частот та перевірки гіпотези про нормальність розподілу.

Проаналізуйте та порівняйте значення середнього, медіани та моди. Що можна сказати про асиметрію розподілу? Якщо значення сильно відрізняються, про що це може свідчити?

Інтерпретуйте отриманий довірчий інтервал для середнього. Що означає твердження "з ймовірністю 95% середнє генеральної сукупності знаходиться в межах [...]"?

На основі результатів критерію Пірсона зробіть висновок про те, чи підпорядковуються ваші дані нормальному закону розподілу. Якщо ні, запропонуйте, який інший закон міг би їх описувати.

4. Контрольні питання

1. Що таке варіаційний ряд? Чим дискретний варіаційний ряд відрізняється від інтервального?
2. Як побудувати гістограму в Google Sheets та в Python? Яку інформацію про розподіл даних вона надає?
3. Назвіть та охарактеризуйте основні міри центральної тенденції. У яких випадках доцільніше використовувати медіану замість середнього арифметичного?
4. Що таке дисперсія та стандартне відхилення? Яку інформацію про дані вони надають?
5. Поясніть різницю між точковою та інтервальною оцінкою параметра. Що таке довірчий інтервал і як він залежить від обсягу вибірки та рівня надійності?
6. Сформулюйте нульову та альтернативну гіпотези для перевірки відповідності розподілу нормальному закону.
7. Як працює критерій згоди Пірсона (хі-квадрат)? Як інтерпретувати отримане p-value?
8. Які функції Google Sheets використовувалися для обчислення статистичних показників? Для чого призначені функції QUARTILE(), VAR.S(), STDEV.S()?
9. Як згенерувати випадкові дані, що підпорядковуються нормальному розподілу, за допомогою бібліотеки numpy?
10. Як за допомогою бібліотеки scipy.stats перевірити гіпотезу про нормальність розподілу? Які ще критерії згоди існують?

Лабораторна робота №4
Ієрархічний кластерний аналіз даних у Python та Google Sheets

1. Мета роботи

Формування практичних навичок проведення ієрархічного кластерного аналізу з використанням мови програмування Python (бібліотеки pandas, numpy, matplotlib, scipy.cluster.hierarchy) та табличного процесора Google Sheets. Оволодіння методами побудови матриць відстаней, застосування різних методів зв'язку кластерів, візуалізації результатів у вигляді дендрограм та інтерпретації отриманих кластерів.

2. Короткі теоретичні відомості

Кластерний аналіз – це задача групування множини об'єктів на підмножини (кластери) таким чином, щоб об'єкти всередині одного кластера були максимально схожими, а об'єкти з різних кластерів – максимально відмінними. Це метод навчання без учителя, оскільки ми не маємо наперед визначених міток класів.

Ієрархічний кластерний аналіз будує не одне розбиття, а цілу систему вкладених розбиттів – ієрархію кластерів. Результатом є дерево кластерів, яке називається дендрограмою. Існує два основних підходи до побудови ієрархії:

Агломеративні алгоритми (знизу вгору): На початку кожен об'єкт вважається окремим кластером. На кожному кроці два найближчі кластери об'єднуються в один. Процес триває доти, поки всі об'єкти не опиняться в одному кластері.

Дивізимні алгоритми (зверху вниз): На початку всі об'єкти належать до одного кластера, який потім поступово розбивається на менші.

На практиці агломеративні алгоритми є значно поширенішими.

Ключовими елементами ієрархічної кластеризації є:

Міра близькості між об'єктами: Визначає, як обчислювати відстань (або схожість) між двома об'єктами. Найпоширенішою метрикою для числових даних є евклідова відстань.

Метод зв'язку (Linkage Method): Визначає, як обчислювати відстань між двома кластерами після того, як вони містять більше ніж один об'єкт. Найпоширеніші методи:

Метод найближчого сусіда (single linkage): відстань між кластерами – це мінімальна відстань між будь-якими двома об'єктами з різних кластерів. Схильний утворювати довгі "ланцюжкові" кластери.

Метод найдальшого сусіда (complete linkage): відстань між кластерами – це максимальна відстань між об'єктами. Добре працює для компактних, добре відокремлених кластерів.

Метод середнього зв'язку (average linkage): відстань між кластерами – це середнє арифметичне всіх попарних відстаней між об'єктами з різних кластерів.

Метод Уорда (Ward’s method): об’єднує ті кластери, які призводять до найменшого зростання внутрішньокластерної дисперсії. Часто дає дуже гарні, збалансовані результати.

Визначення оптимальної кількості кластерів є важливим етапом аналізу. На дендрограмі це можна зробити, знайшовши горизонтальний рівень, де "стрибок" відстані між об’єднаннями є найбільшим. Інтуїтивно це означає, що ми об’єднуємо вже досить різнорідні групи.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно створити невеликий набір даних, що складається з 8–10 об’єктів, кожен з яких описаний двома числовими ознаками. Тематика обирається самостійно. Наприклад: автомобілі (потужність, ціна), міста (населення, площа), спортсмени (зріст, вага), фільми (бюджет, рейтинг) тощо.

Завдання 1. Робота в Google Sheets. Підготовка даних та візуалізація

Створіть нову таблицю в Google Sheets. Внесіть ваш набір даних: перший стовпець – назви об’єктів (наприклад, "Авто А", "Авто В" тощо), другий та третій стовпці – значення двох числових ознак.

Побудуйте точкову діаграму (діаграму розсіювання) для вашого набору даних. Вісь X – перша ознака, вісь Y – друга ознака. Підпишіть точки назвами об’єктів (це можна зробити вручну за допомогою інструменту "Підпис" або додавши підписи даних).

Візуально проаналізуйте діаграму. Чи можна виділити на ній якісь групи об’єктів? Запишіть свої припущення.

Завдання 2. Робота в Python. Побудова матриці відстаней та ієрархічна кластеризація

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, scipy.cluster.hierarchy as sch, scipy.spatial.distance as dist.

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) або створіть його безпосередньо у вигляді pandas.DataFrame.

Створіть матрицю вхідних даних X (лише числові ознаки). За допомогою функції dist.pdist() обчисліть попарні евклідові відстані між усіма об’єктами. Виведіть отриманий вектор відстаней на екран.

Перетворіть вектор відстаней у квадратну матрицю відстаней за допомогою функції dist.squareform(). Виведіть її на екран у вигляді

pandas.DataFrame, де рядки та стовпці підписані назвами об’єктів. Визначте, які два об’єкти є найближчими (мають мінімальну відстань).

Виконайте ієрархічну агломеративну кластеризацію за допомогою функції sch.linkage(). Проведіть кластеризацію, використовуючи три різні методи зв’язку: метод найближчого сусіда (‘single’), метод найдальшого сусіда (‘complete’) та метод Уорда (‘ward’).

Для кожного з трьох методів побудуйте дендрограму за допомогою функції sch.dendrogram(). Підпишіть нижню вісь назвами об’єктів. Виведіть дендрограми на екран (можна в окремих вікнах або на окремих підграфіках).

Завдання 3. Аналіз результатів та визначення оптимальної кількості кластерів

Для кожної з трьох побудованих дендрограм візуально визначте точку, де відбувається найбільший "стрибок" відстані при об’єднанні кластерів. Запишіть, яка кількість кластерів є оптимальною з точки зору цього критерію.

Для методу Уорда (як найбільш збалансованого) виконайте наступне:

За допомогою функції sch.fcluster() отримайте вектор міток кластерів для вибраної вами оптимальної кількості кластерів.

Додайте цей вектор як новий стовпець до вашого початкового DataFrame.

Виведіть на екран таблицю з об’єктами та їх приналежністю до кластерів.

Побудуйте точкову діаграму, аналогічну до тієї, що ви будували в Google Sheets, але тепер розфарбуйте точки відповідно до отриманих міток кластерів. Додайте легенду.

Завдання 4. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду та вихідну діаграму розсіювання.

Наведіть фрагменти Python-коду для обчислення матриці відстаней, проведення кластеризації та побудови дендрограм.

Порівняйте дендрограми, отримані різними методами зв’язку. Який метод, на вашу думку, найкраще відобразив структуру ваших даних? Чому?

Поясніть, на основі чого ви обрали оптимальну кількість кластерів. Наведіть відповідне значення з дендрограми.

Проаналізуйте склад отриманих кластерів. Чи можна дати їм змістовні назви на основі значень їхніх ознак? Наприклад, "кластер дешевих та малопотужних автомобілів", "кластер дорогих та потужних" тощо. Запишіть вашу інтерпретацію.

4. Контрольні питання

1. У чому полягає сутність ієрархічного кластерного аналізу? Чим він відрізняється від плоских (неієрархічних) методів кластеризації, таких як k-means?
2. Поясніть різницю між агломеративними та дивізимними алгоритмами. Який з них є більш поширеним і чому?

3. Що таке матриця відстаней і як вона будується? Як її отримати в Python за допомогою бібліотеки scipy?
4. Охарактеризуйте основні методи зв'язку кластерів (single, complete, average, Ward). Які форми кластерів вони мають тенденцію утворювати?
5. Для чого перед проведенням кластеризації часто виконують нормалізацію або стандартизацію даних? Що може статися, якщо цього не зробити?
6. Що таке дендрограма? Яку інформацію вона надає? Як за її допомогою визначити оптимальну кількість кластерів?
7. Як за допомогою функції scipy.cluster.hierarchy.fcluster отримати мітки кластерів для заданої кількості кластерів?
8. Як побудувати точкову діаграму з кольоровим кодуванням за кластерами в Python (matplotlib)?
9. Яке призначення функцій pdist() та squareform() з бібліотеки scipy.spatial.distance?
10. Як інтерпретувати результати кластерного аналізу? Що таке профіль кластера і як він може допомогти в інтерпретації?

Лабораторна робота №5
Обчислення мір близькості між об'єктами в Python та Google Sheets

1. Мета роботи

Формування практичних навичок обчислення різних мір близькості (відстаней та мір подібності) для об'єктів, описаних різними типами даних, з використанням мови програмування Python (бібліотеки pandas, numpy, scipy.spatial.distance) та табличного процесора Google Sheets. Оволодіння методами побудови матриць відстаней, матриць подібності та таблиць спряженості для аналізу зв'язків між об'єктами.

2. Короткі теоретичні відомості

В інтелектуальному аналізі даних фундаментальним поняттям є близькість між об'єктами. Саме на її основі будуються алгоритми класифікації, кластеризації, пошуку аномалій та рекомендаційні системи. Залежно від контексту, близькість може вимірюватися як схожість (більші значення для більш схожих об'єктів) або відстань (більші значення для більш відмінних об'єктів). Ці дві концепції є взаємодоповнювальними: якщо міра несхожості нормована від 0 до 1, то міра подібності може бути обчислена як 1 мінус міра несхожості.

Результатом попарного порівняння всіх об'єктів набору даних є матриця близькості. Це квадратна таблиця розмірністю n×n (де n – кількість об'єктів), на головній діагоналі якої для матриці відстаней стоять нулі, а для матриці подібності – одиниці. Матриця є симетричною, оскільки відстань між об'єктами i та j не залежить від напрямку.

Вибір конкретної міри близькості залежить від типу даних:

Для числових даних найпоширенішими є метрики відстані:

Евклідова відстань: геометрична відстань між точками у багатовимірному просторі. Чутлива до масштабу ознак.

Манхеттенська відстань (міських кварталів): сума абсолютних різниць координат. Менш чутлива до викидів, ніж евклідова.

Відстань Чебишева: максимальна різниця за будь-якою координатою. Корисна, коли об’єкти вважаються різними, якщо вони відрізня хоча б за однією важливою ознакою.

Косинусна відстань: (1 - косинус подібності). Косинус подібності вимірює косинус кута між векторами ознак, ігноруючи їх довжину. Це робить його особливо корисним для порівняння текстових документів (розріджених векторів).

Для категоріальних даних використовуються інші підходи:

Для простої номінальної ознаки: несхожість = 0, якщо значення збігаються, і 1 – якщо відрізняються.

Для набору категоріальних ознак: відстань Хеммінга дорівнює кількості ознак, за якими об’єкти відрізняються. Її нормований варіант – відсоток незгоди.

Для аналізу зв’язку між двома категоріальними змінними будують таблиці спряженості, а для оцінки значущості зв’язку використовують критерій хі-квадрат (χ^2).

Для бінарних даних (окремий випадок категоріальних) важливо розрізняти симетричні та асиметричні ознаки. Для симетричних (де обидва значення рівноправні, наприклад, стать) використовують коефіцієнт простої відповідності (SMC). Для асиметричних (де одне значення важливіше, наприклад, результат тесту) використовують коефіцієнт Жаккара, який ігнорує збіги нулів.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно створити невеликий набір даних, що складається з 6–8 об’єктів, описаних різнотипними ознаками. Тематика обирається самостійно. Наприклад: інформація про кандидатів на роботу (стать, вік, освіта, досвід роботи, знання мов), про фільми (жанр, рік випуску, рейтинг, бюджет), про автомобілі тощо.

Завдання 1. Робота в Google Sheets. Міри близькості для простих типів даних

Створіть нову таблицю в Google Sheets. На окремих аркушах створіть три невеликі набори даних:

Аркуш "Номінальна": 4 об’єкти з однією номінальною ознакою (наприклад, "Стать").

Аркуш "Порядкова": 4 об’єкти з однією порядковою ознакою (наприклад, "Рівень освіти": середня, бакалавр, магістр).

Аркуш "Числова": 4 об’єкти з однією числовою ознакою (наприклад, "Вік").

Для кожного з трьох аркушів вручну (за допомогою формул) побудуйте матрицю несхожості розмірністю 4×4. Для номінальної ознаки використовуйте правило: 0 – значення збігаються, 1 – відрізняються. Для порядкової ознаки попередньо перетворіть значення в ранги, а потім обчисліть нормовану різницю. Для числової ознаки обчисліть абсолютну різницю.

На основі отриманих матриць несхожості створіть матриці подібності за формулою: Подібність = 1 - Несхожість (попередньо переконавшись, що значення несхожості знаходяться в діапазоні [0,1]). Для числової ознаки може знадобитися попередня нормалізація.

Завдання 2. Робота в Python. Міри відстаней для числових даних

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте бібліотеки pandas, numpy, matplotlib.pyplot, scipy.spatial.distance as dist.

Створіть набір даних, що складається з 6 об'єктів, кожен з яких описаний двома числовими ознаками. Наприклад, це можуть бути координати точок на площині або будь-які інші дві характеристики. Збережіть дані у pandas.DataFrame з іменами об'єктів як індекс.

Виберіть два будь-які об'єкти з вашого набору (назвемо їх A та B). За допомогою відповідних функцій з scipy.spatial.distance обчисліть між ними наступні відстані:

1. Евклідова (euclidean)
2. Квадрат евклідової (sqeuclidean) – зверніть увагу, що ця функція також є в бібліотеці.
3. Манхеттенська (cityblock)
4. Чебишева (chebyshev)
5. Косинусна (cosine)

Виведіть результати обчислень на екран у зручному форматі (наприклад, як словник).

Завдання 3. Робота в Python. Матриці відстаней та подібності

Для всього набору з 6 об'єктів за допомогою функції dist.pdist() обчисліть попарні евклідові відстані. Виведіть на екран отриманий вектор стислої форми.

За допомогою функції dist.squareform() перетворіть цей вектор у квадратну матрицю відстаней. Створіть з неї pandas.DataFrame з підписаними рядками та стовпцями (імена об'єктів) та виведіть на екран.

Визначте, які два об'єкти є найближчими, а які – найвіддаленішими.

Використовуючи функцію dist.pdist(), обчисліть косинусну відстань для всіх пар об'єктів. Побудуйте відповідну квадратну матрицю. Порівняйте її з матрицею евклідових відстаней. Чи зберігаються співвідношення між об'єктами?

Завдання 4. Аналіз бінарних даних (Python)

Створіть два бінарні вектори довжиною 8, що описують два об’єкти (наприклад, наявність/відсутність певних характеристик). Вектори можна задати як списки [1,0,1,0,...].

Визначте, чи є ваші бінарні ознаки симетричними чи асиметричними. Обґрунтуйте своє рішення.

"Вручну" (написавши невеликий код на Python) обчисліть значення q, t, p, r для цих двох векторів (де q – кількість збігів одиниць, t – збігів нулів, p – (0,1), r – (1,0)).

На основі цих значень розрахуйте:

Коефіцієнт простої відповідності (SMC)

Коефіцієнт Жаккара

Поясніть, чому отримані значення відрізняються і який коефіцієнт є більш доречним для вашого випадку.

Завдання 5. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду для обчислення різних типів відстаней, побудови матриць та аналізу бінарних даних.

Проаналізуйте отримані матриці відстаней. Чи є в них якісь закономірності? Чи можна на основі цих матриць припустити, які об’єкти можуть утворити кластери?

Порівняйте матриці, отримані за допомогою різних метрик (евклідова, косинусна). Які висновки про структуру даних можна зробити на основі цих відмінностей?

Поясніть, у яких практичних задачах може бути корисним використання коефіцієнта Жаккара замість коефіцієнта простої відповідності.

4. Контрольні питання

1. Поясніть різницю між мірами подібності та мірами несхожості. Як вони співвідносяться між собою?
2. Що таке матриця близькості? Які властивості мають матриці відстаней та матриці подібності?
3. Назвіть основні метрики відстані для числових даних. У яких випадках доцільніше використовувати манхеттенську відстань замість евклідової?
4. Що таке косинусна відстань і для якого типу даних вона особливо добре підходить? Чому?
5. Як обчислити відстань між об’єктами, описаними категоріальними ознаками? Що таке відстань Хеммінга?
6. Як побудувати таблицю спряженості та для чого вона використовується?
7. Поясніть різницю між симетричними та асиметричними бінарними атрибутами. Наведіть приклади.

8. Як обчислюються коефіцієнт простої відповідності (SMC) та коефіцієнт Жаккара? Чому вони можуть давати різні результати для одних і тих самих даних?
9. Які функції бібліотеки scipy.spatial.distance використовувалися в роботі? Для чого призначені функції pdist() та squareform()?
10. Як створити в Google Sheets матрицю відстаней для об’єктів з однією числовою ознакою? Які формули для цього потрібні?

Лабораторна робота №6
Кластеризація даних методами k-means та c-means у Python

1. Мета роботи

Формування практичних навичок проведення кластерного аналізу з використанням алгоритмів k-means (чітка кластеризація) та Fuzzy c-means (нечітка кластеризація) в середовищі Python. Оволодіння методами визначення оптимальної кількості кластерів, візуалізації результатів кластеризації, аналізу матриці нечіткого розбиття та інтерпретації отриманих кластерів.

2. Короткі теоретичні відомості

Кластерний аналіз – це задача групування об’єктів на підмножини (кластери) таким чином, щоб об’єкти всередині одного кластера були максимально схожими, а об’єкти з різних кластерів – максимально відмінними. Це метод навчання без учителя.

Залежно від підходу, алгоритми кластеризації поділяються на:

Ієрархічні (будують дерево кластерів) та неієрархічні (плоскі) (будують одне розбиття).

Чіткі (hard) – кожен об’єкт належить лише одному кластеру.

Нечіткі (fuzzy, soft) – кожен об’єкт належить до кожного кластера з певним ступенем належності (від 0 до 1).

Алгоритм k-means є найпопулярнішим представником плоских чітких алгоритмів. Його робота полягає в ітеративному повторенні двох кроків:

Призначення: Кожен об’єкт відноситься до найближчого центроїда (центру кластера).

Оновлення: Центроїди перераховуються як середнє арифметичне всіх об’єктів, що належать до даного кластера.

Алгоритм завершується, коли центроїди перестають змінюватися (або змінюються незначно). Недоліками k-means є необхідність заздалегідь задавати кількість кластерів k та чутливість до випадкового вибору початкових центроїдів. Для пом’якшення останнього часто використовують ініціалізацію k-means++ (реалізована в scikit-learn за замовчуванням).

Алгоритм Fuzzy c-means є нечітким узагальненням k-means. Замість жорсткого призначення, він обчислює для кожного об’єкта ступінь належності до кожного кластера. Результатом є матриця нечіткого розбиття U розмірністю k×n, де сума значень у кожному стовпці дорівнює 1. На основі цієї матриці об’єкт можна віднести до кластера з максимальним ступенем належності, або використовувати нечітку належність для подальшого аналізу.

Визначення оптимальної кількості кластерів k є важливим етапом. Для k-means часто використовують метод "ліктя" (elbow method), який полягає в аналізі графіка залежності суми квадратів відстаней до центроїдів (inertia) від k. Точка "перегину" графіка вказує на оптимальне k.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно створити набір даних, що складається з 20–30 об’єктів, кожен з яких описаний двома числовими ознаками. Тематика обирається самостійно. Наприклад: координати міст на карті, характеристики спортивних досягнень (зріст, вага), параметри автомобілів (потужність, ціна), показники успішності студентів (кількість годин навчання, середній бал) тощо.

Завдання 1. Робота в Google Sheets. Підготовка даних та візуалізація

Створіть нову таблицю в Google Sheets. Внесіть ваш набір даних: перший стовпець – назви об’єктів (наприклад, "Точка A", "Студент 1" тощо), другий та третій стовпці – значення двох числових ознак.

Побудуйте точкову діаграму (діаграму розсіювання) для вашого набору даних. Вісь X – перша ознака, вісь Y – друга ознака.

Візуально проаналізуйте діаграму. Скільки кластерів ви можете приблизно виділити? Запишіть своє припущення.

Завдання 2. Робота в Python. Кластеризація методом k-means

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, sklearn.cluster.KMeans.

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) у pandas.DataFrame.

Створіть матрицю вхідних даних X (лише числові ознаки).

Використовуючи метод "ліктя", визначте оптимальну кількість кластерів для вашого набору даних:

Створіть цикл для значень k від 1 до 8 (або більше).

Для кожного k створіть модель KMeans, навчіть її на даних X та збережіть значення inertia_ (сума квадратів відстаней до центроїдів).

Побудуйте графік залежності inertia від k. Визначте точку "перегину" (elbow) та оберіть оптимальне k.

Виконайте кластеризацію методом k-means з оптимальним k, використовуючи KMeans. Отримайте мітки кластерів для кожного об’єкта (labels_) та координати центроїдів (cluster_centers_).

Побудуйте точкову діаграму, на якій точки розфарбовані відповідно до отриманих міток кластерів. Додайте на графік центроїди, позначивши їх, наприклад, червоними хрестиками більшого розміру. Додайте легенду та заголовок.

Завдання 3. Робота в Python. Нечітка кластеризація (Fuzzy c-means)

Встановіть бібліотеку для нечіткої кластеризації. Наприклад, scikit-fuzzy (pip install scikit-fuzzy). Імпортуйте її: import skfuzzy as fuzz.

Для роботи з skfuzzy дані потрібно подати у транспонованому вигляді: рядки – ознаки, стовпці – об’єкти. Підготуйте матрицю data з вашого DataFrame, транспонуйте її (.T).

Виконайте нечітку кластеризацію за допомогою функції fuzz.cmeans. Використайте ту саму кількість кластерів k, що й для k-means. Встановіть параметр нечіткості m = 2 (значення за замовчуванням). Функція повертає кілька об’єктів; вам знадобляться:

cntr – координати центрів кластерів.

u – матриця нечіткого розбиття (ступені належності). Розмірність матриці: k×n.

Виведіть на екран матрицю нечіткого розбиття у вигляді pandas.DataFrame, де рядки – кластери, стовпці – об’єкти (підписані їхніми назвами). Значення мають бути округлені до 2-3 знаків після коми.

На основі матриці нечіткого розбиття визначте, до якого кластера належить кожен об’єкт за максимальним ступенем належності (чітка класифікація). Створіть список таких міток.

Побудуйте точкову діаграму, аналогічну до тієї, що ви будували для k-means, але тепер використовуючи мітки, отримані з нечіткої кластеризації. Додайте на графік центроїди (cntr).

Завдання 4. Порівняльний аналіз та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду для визначення оптимального k методом "ліктя", для кластеризації k-means та для нечіткої кластеризації.

Порівняйте результати кластеризації, отримані двома методами. Чи збігаються мітки для більшості об’єктів? Якщо є відмінності, спробуйте їх пояснити (наприклад, об’єкти на межі кластерів).

Проаналізуйте матрицю нечіткого розбиття. Чи є об’єкти, ступінь належності яких до "свого" кластера є не дуже високою (наприклад, менше 0.6)? Що це може означати?

Зробіть висновок про те, який метод (чіткий чи нечіткий) є більш інформативним для вашого набору даних і чому.

4. Контрольні питання

1. У чому полягає сутність алгоритму k-means? Опишіть його основні кроки.

2. Назвіть основні переваги та недоліки алгоритму k-means.
3. Що таке метод "ліктя" (elbow method) і для чого він використовується? Як інтерпретувати отриманий графік?
4. Чим нечітка кластеризація (fuzzy c-means) відрізняється від чіткої (k-means)?
5. Що таке матриця нечіткого розбиття? Які властивості вона має (розмірність, сума елементів у стовпці)?
6. Як на основі матриці нечіткого розбиття отримати чіткі мітки кластерів?
7. Як впливає на результати кластеризації вибір початкових центроїдів? Як цю проблему вирішують на практиці?
8. Для чого перед проведенням кластеризації часто виконують нормалізацію або стандартизацію даних? Чи потрібна вона у вашому випадку?
9. Які функції бібліотеки scikit-learn використовувалися для кластеризації? Для чого призначений атрибут inertia_ моделі KMeans?
10. Як побудувати точкову діаграму з кольоровим кодуванням за кластерами в Python (matplotlib)? Як додати на графік центроїди?

Лабораторна робота №7
Класифікація даних. Лінійний дискримінантний аналіз (LDA) у Python

1. Мета роботи

Формування практичних навичок розв’язання задач класифікації з використанням лінійного дискримінантного аналізу (Linear Discriminant Analysis, LDA) в середовищі Python. Оволодіння методами оцінки якості класифікатора (матриця помилок, точність, повнота, F-оцінка), побудови дискримінантної функції та її застосування для класифікації нових об’єктів.

2. Короткі теоретичні відомості

Класифікація – це задача навчання з учителем, яка полягає у віднесенні нового об’єкта до одного із заздалегідь відомих класів на основі аналізу його ознак (незалежних змінних). Множина класів є фіксованою та відомою наперед.

Процес побудови класифікатора передбачає наявність двох наборів даних:

Навчальна множина (training set): об’єкти з відомими значеннями ознак та відомою міткою класу. Використовується для побудови моделі.

Тестова множина (test set): об’єкти з відомими значеннями ознак та відомою міткою класу, які НЕ використовувалися для навчання. Використовуються для оцінки якості побудованої моделі.

Важливими проблемами при побудові класифікаторів є перенавчання (overfitting) – модель надто добре запам’ятовує навчальні дані, включаючи шум, і погано узагальнюється на нових даних, та недонавчання (underfitting) – модель є надто простою і не здатна виявити закономірності в даних.

Для оцінки якості бінарного класифікатора використовують матрицю помилок (confusion matrix) та обчислені на її основі метрики:

TP (True Positive): правильно класифіковані об’єкти позитивного класу.

TN (True Negative): правильно класифіковані об’єкти негативного класу.

FP (False Positive): об’єкти негативного класу, помилково віднесені до позитивного (помилка I роду).

FN (False Negative): об’єкти позитивного класу, помилково віднесені до негативного (помилка II роду).

На основі цих значень обчислюються:

Точність (Accuracy): (TP + TN) / (TP + TN + FP + FN) – частка правильних відповідей.

Повнота (Recall): TP / (TP + FN) – частка виявлених об’єктів позитивного класу.

Точність позитивних результатів (Precision): TP / (TP + FP) – частка "справжніх" позитивних серед передбачених.

F-оцінка (F1-score): гармонійне середнє точності та повноти.

Лінійний дискримінантний аналіз (LDA) є класичним статистичним методом класифікації. Його ідея полягає в побудові дискримінантної функції – лінійної комбінації незалежних змінних, яка найкращим чином розділяє об’єкти різних класів. LDA шукає проекцію даних на пряму (у просторі ознак), яка максимізує відстань між середніми різних класів та мінімізує розкид (дисперсію) всередині кожного класу. Для класифікації нового об’єкта обчислюється значення дискримінантної функції, і об’єкт відноситься до того класу, для якого це значення є найбільш імовірним.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно обрати або створити набір даних, що містить не менше 20–30 об’єктів, описаних двома числовими ознаками, та бінарну цільову змінну (клас 0 або 1). Можна скористатися відкритими джерелами (наприклад, набір даних Iris, взявши лише два види) або створити власний набір (наприклад, успішність студентів: години навчання, середній бал -> склав/не склав іспит; характеристики автомобілів -> проданий/не проданий тощо).

Завдання 1. Робота в Google Sheets. Підготовка даних та первинний аналіз

Створіть нову таблицю в Google Sheets. Внесіть ваш набір даних: перший стовпець – ідентифікатор об’єкта (необов’язково), другий та третій стовпці – значення двох числових ознак (назвіть їх, наприклад, X1 та X2), четвертий стовпець – мітка класу (0 або 1).

Побудуйте точкову діаграму (діаграму розсіювання) для вашого набору даних, де вісь X – перша ознака, вісь Y – друга ознака. Налаштуйте кольори точок залежно від класу (наприклад, клас 0 – синім, клас 1 – червоним). Додайте легенду.

Візуально проаналізуйте графік. Чи можна розділити класи прямою лінією? Якщо так, спробуйте приблизно провести її "вручну" та записати рівняння у звіті.

Завдання 2. Робота в Python. Побудова та оцінка LDA-класифікатора

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, sklearn.model_selection, sklearn.discriminant_analysis.LinearDiscriminantAnalysis, sklearn.metrics.

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) у pandas.DataFrame.

Розділіть дані на матрицю ознак X (стовпці з X1 та X2) та вектор цільових міток y (стовпець з класами).

За допомогою функції train_test_split з бібліотеки sklearn.model_selection розділіть дані на навчальну (70%) та тестову (30%) вибірки. Встановіть параметр random_state для відтворюваності результатів.

Створіть об’єкт класифікатора LDA: lda = LinearDiscriminantAnalysis().

Навчіть класифікатор на навчальних даних за допомогою методу fit().

Отримайте коефіцієнти дискримінантної функції. Вони зберігаються в атрибутах lda.coef_ та lda.intercept_. Виведіть їх на екран. Запишіть рівняння прямої, що розділяє класи (це буде лінія, де значення дискримінантної функції дорівнює 0).

Завдання 3. Оцінка якості класифікатора

Використайте навчений класифікатор для передбачення міток класів на тестовій вибірці: y_pred = lda.predict(X_test).

Побудуйте матрицю помилок за допомогою функції confusion_matrix з sklearn.metrics. Виведіть її на екран у вигляді pandas.DataFrame з підписаними рядками та стовпцями ("Actual 0", "Actual 1" тощо).

Обчисліть та виведіть на екран наступні метрики якості для тестової вибірки:

Accuracy (точність) – accuracy_score

Precision (точність позитивних результатів) – precision_score

Recall (повнота) – recall_score

F1-score – f1_score

Виведіть звіт з усіма метриками за допомогою функції classification_report.

Завдання 4. Візуалізація результатів

Побудуйте точкову діаграму для тестової вибірки. Точки мають бути розфарбовані відповідно до їхніх фактичних класів.

На тому ж графіку відобразіть розділяючу пряму, отриману за допомогою LDA. Для цього потрібно:

Визначити межі графіка по осі X (мінімум та максимум значень X1).

Обчислити для цих меж відповідні значення X2, використовуючи рівняння прямої, отримане з коефіцієнтів LDA.

Побудувати лінію за допомогою plt.plot().

Позначте на графіку точки, які були класифіковані неправильно (наприклад, обвести їх чорним кружечком). Додайте легенду, заголовок та підписи осей.

Завдання 5. Класифікація нового об’єкта

Задайте координати нового об’єкта (значення X1_new та X2_new), який не входив ні до навчальної, ні до тестової вибірки.

За допомогою навченого класифікатора lda передбачте його клас (lda.predict()).

Виведіть на екран передбачений клас. Додайте цей новий об’єкт на побудований у попередньому завданні графік, позначивши його, наприклад, зеленою зірочкою великого розміру.

Завдання 6. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду для розділення даних, навчання LDA, обчислення метрик та побудови графіка з розділяючою прямою.

Проаналізуйте отриману матрицю помилок та метрики якості. Наскільки добре класифікатор впорався із завданням? Який тип помилок (FP чи FN) трапляється частіше? Про що це може свідчити?

Порівняйте рівняння прямої, отримане за допомогою LDA, з вашим візуальним припущенням із Завдання 1. Наскільки вони близькі?

Зробіть висновок про застосовність лінійного дискримінантного аналізу для вашого набору даних. Чи є дані лінійно роздільними? Якщо ні, то які можна зробити висновки?

4. Контрольні питання

1. Сформулюйте задачу класифікації. Що таке незалежні змінні та залежна змінна (цільова мітка)?
2. Поясніть різницю між навчальною та тестовою вибірками. Чому для оцінки якості моделі використовують саме тестову вибірку?
3. Що таке перенавчання (overfitting) і недонавчання (underfitting)? Як їх можна виявити?
4. Що таке матриця помилок (confusion matrix)? Поясніть зміст TP, TN, FP, FN.

5. Які метрики якості класифікатора можна обчислити на основі матриці помилок? Що характеризує точність (accuracy), повнота (recall) та точність позитивних результатів (precision)?
6. У чому полягає сутність лінійного дискримінантного аналізу (LDA)? Яка геометрична інтерпретація цього методу?
7. Як за допомогою LDA отримати рівняння прямої, що розділяє класи? Яку інформацію надають атрибути coef_ та intercept_ навченої моделі?
8. Як побудувати матрицю помилок та обчислити метрики якості за допомогою бібліотеки sklearn.metrics?
9. Для чого використовується функція train_test_split? Які параметри вона приймає?
10. Як, маючи рівняння прямої, побудувати її на графіку в Python (matplotlib)? Як додати на графік нову точку?

Лабораторна робота №8
Методи класифікації в Python: KNN, Naive Bayes, дерева рішень, SVM

1. Мета роботи

Формування практичних навичок розв'язання задач класифікації з використанням різних методів машинного навчання в середовищі Python. Оволодіння методами побудови та оцінки класифікаторів на основі k-найближчих сусідів (KNN), наївного баєсівського класифікатора (Naive Bayes), дерев рішень (Decision Trees) та методу опорних векторів (SVM). Порівняльний аналіз точності різних методів на прикладі класичного набору даних.

2. Короткі теоретичні відомості

Класифікація – це задача навчання з учителем, яка полягає у віднесенні нового об'єкта до одного із заздалегідь відомих класів на основі аналізу його ознак. Існує багато різних методів класифікації, кожен з яких має свої сильні та слабкі сторони. У цій роботі ми розглянемо чотири популярні алгоритми.

Метод k-найближчих сусідів (k-Nearest Neighbors, KNN) є одним з найпростіших алгоритмів. Він не будує модель у явному вигляді, а запам'ятовує всю навчальну вибірку. Для класифікації нового об'єкта алгоритм знаходить k найближчих до нього об'єктів у навчальній вибірці (за певною метрикою відстані, наприклад, евклідовою) і відносить новий об'єкт до класу, який є найпоширенішим серед цих сусідів (просте голосування). Можливе також зважене голосування, де голос ближчого сусіда має більшу вагу. Критичним параметром є вибір k.

Наївний баєсівський класифікатор (Naive Bayes) базується на теоремі Байєса та "наївному" припущенні про незалежність ознак. Він обчислює ймовірність належності об’єкта до кожного класу на основі апріорних ймовірностей класів та умовних ймовірностей значень ознак для цих класів. Об’єкт відноситься до класу з найбільшою апостеріорною ймовірністю. Незважаючи на "наївність" припущення, цей метод часто працює дуже добре, особливо для текстових даних.

Дерева рішень (Decision Trees) будують модель у вигляді ієрархічної деревоподібної структури. Кожен внутрішній вузол містить перевірку певної ознаки, кожне ребро відповідає результату цієї перевірки, а кожен листок – кінцевому класифікаційному висновку. Процес класифікації полягає в проходженні від кореня до листка. Дерева рішень легко інтерпретувати, але вони схильні до перенавчання. Для покращення узагальнюючої здатності застосовують процедуру відсікання гілок (pruning) або будують ансамблі дерев, такі як випадковий ліс (Random Forest).

Метод опорних векторів (Support Vector Machine, SVM) має геометричну інтерпретацію. Він намагається знайти в просторі ознак гіперплощину, яка найкращим чином (з максимальним зазором) розділяє об’єкти різних класів. Найближчі до гіперплощини об’єкти називаються опорними векторами. SVM добре працює для даних високої розмірності. За допомогою "ядерного трюку" (kernel trick) SVM може будувати нелінійні розділяючі поверхні.

3. Завдання до лабораторної роботи

Для виконання роботи ми використаємо класичний набір даних Iris (квіти ірису), який міститься в бібліотеці scikit-learn. Цей набір даних містить 150 зразків ірисів трьох видів (setosa, versicolor, virginica), описаних чотирма ознаками (довжина та ширина чашолистка, довжина та ширина пелюстки).

Завдання 1. Завантаження, підготовка та візуалізація даних

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, seaborn, sklearn.datasets, sklearn.model_selection, sklearn.neighbors, sklearn.naive_bayes, sklearn.tree, sklearn.svm, sklearn.metrics.

Завантажте набір даних Iris за допомогою datasets.load_iris(). Перетворіть його в pandas.DataFrame для зручності. Виведіть перші 5 рядків, загальну інформацію про датафрейм та описову статистику.

Розділіть дані на ознаки (X) та цільову змінну (y). Використовуючи train_test_split, розділіть дані на навчальну (70%) та тестову (30%) вибірки. Встановіть random_state для відтворюваності.

За допомогою бібліотеки seaborn побудуйте парний графік розсіювання (pairplot) для всього набору даних, розфарбувавши точки за видами ірисів (hue='species'). Проаналізуйте, які ознаки найкраще розділяють класи.

Завдання 2. Класифікація методом k-найближчих сусідів (KNN)

Створіть об'єкт класифікатора KNN з параметром n_neighbors=5 (значення за замовчуванням). Навчіть його на навчальних даних.

Зробіть передбачення на тестовій вибірці.

Обчисліть та виведіть точність (accuracy) класифікатора на тестовій вибірці.

Побудуйте матрицю помилок та класифікаційний звіт (precision, recall, f1-score).

Додаткове завдання: Підберіть оптимальне значення k, перебираючи значення від 1 до 20 та будуючи графік залежності точності на тестовій вибірці від k. Чи впливає вибір k на результат?

Завдання 3. Класифікація методом Naive Bayes

Створіть об'єкт класифікатора GaussianNB (для неперервних ознак). Навчіть його на навчальних даних.

Зробіть передбачення на тестовій вибірці.

Обчисліть та виведіть точність, матрицю помилок та класифікаційний звіт.

Порівняйте отримані результати з KNN.

Завдання 4. Класифікація за допомогою дерева рішень

Створіть об'єкт класифікатора DecisionTreeClassifier з параметром random_state=42. Навчіть його.

Зробіть передбачення на тестовій вибірці.

Обчисліть та виведіть точність, матрицю помилок та класифікаційний звіт.

Візуалізуйте побудоване дерево рішень за допомогою функції plot_tree з sklearn.tree. Збережіть отриманий графік. Проаналізуйте, які ознаки є найважливішими для класифікації згідно з деревом.

Завдання 5. Класифікація методом опорних векторів (SVM)

Створіть об'єкт класифікатора SVC (SVM для класифікації) з параметром kernel='linear' (лінійне ядро). Навчіть його.

Зробіть передбачення на тестовій вибірці. Обчисліть та виведіть точність.

Змініть ядро на kernel='rbf' (радіальна базисна функція) та kernel='poly' (поліноміальне). Для кожного варіанту навчіть класифікатор, зробіть передбачення та виведіть точність.

Порівняйте результати для різних ядер. Яке ядро дало найкращу точність?

Завдання 6. Порівняльний аналіз

Створіть зведену таблицю, де для кожного з чотирьох методів (KNN, Naive Bayes, Decision Tree, SVM) буде вказано точність на тестовій вибірці.

Побудуйте стовпчикову діаграму, що візуалізує точність різних методів.

Зробіть висновок про те, який метод найкраще впорався із класифікацією ірисів. Чому, на вашу думку, одні методи працюють краще за інші для цього набору даних?

Завдання 7. Класифікація нового об’єкта (за бажанням)

Виберіть довільний набір значень ознак для нової квітки (наприклад, [5.1, 3.5, 1.4, 0.2] для setosa).

За допомогою кожного з навчених класифікаторів передбачте її вид. Виведіть результати.

4. Контрольні питання

1. Сформулюйте задачу класифікації. Чим вона відрізняється від задачі регресії?
2. Поясніть принцип роботи методу k-найближчих сусідів (KNN). Як вибір параметра k впливає на результат?
3. У чому полягає "наївність" наївного баєсівського класифікатора? На якій теоремі він базується?
4. Як обчислюються апріорні та апостеріорні ймовірності в методі Naive Bayes?
5. Опишіть структуру дерева рішень. Що таке корінь, вузол, листок? Як відбувається процес класифікації?
6. Яка геометрична інтерпретація методу опорних векторів (SVM)? Що таке опорні вектори та розділяюча гіперплощина?
7. Для чого в методі SVM використовуються різні ядра (kernel)? Наведіть приклади ядер.
8. Як оцінити якість класифікатора? Що таке матриця помилок, точність (accuracy), повнота (recall) та точність позитивних результатів (precision)?
9. Чому важливо розділяти дані на навчальну та тестову вибірки? Що таке перенавчання (overfitting)?
10. Як побудувати матрицю помилок та класифікаційний звіт за допомогою бібліотеки sklearn.metrics? Як візуалізувати дерево рішень?

Лабораторна робота №9
Пошук асоціативних правил.
Алгоритм Apriori в Python

1. Мета роботи

Формування практичних навичок пошуку асоціативних правил у транзакційних даних з використанням алгоритму Apriori в середовищі Python (бібліотека mlxtend.frequent_patterns). Оволодіння методами оцінки знайдених

правил за допомогою метрик підтримки (support), достовірності (confidence), ліфту (lift) та левериджу (leverage), а також інтерпретації отриманих результатів.

2. Короткі теоретичні відомості

Асоціативні правила – це імплікації виду X → Y, які дозволяють виявити приховані закономірності між подіями, що відбуваються спільно. Класичним прикладом є аналіз ринкового кошика (market basket analysis), де правило "якщо клієнт купує хліб, то з високою ймовірністю він також купить молоко" може бути використане для оптимізації розташування товарів або створення рекомендацій.

Для формального опису задачі вводяться такі поняття:

Транзакція – набір елементів (товарів, подій), які з'явилися разом (наприклад, один чек).

Предметний набір (itemset) – непорожня множина елементів.

Асоціативне правило – імплікація X → Y, де X та Y – предметні набори, що не перетинаються. X називають умовою (antecedent), а Y – наслідком (consequent).

Для оцінки якості та значущості асоціативних правил використовують кілька метрик:

Підтримка (Support) правила X → Y – частка транзакцій, які містять одночасно і X, і Y. Це міра статистичної значущості правила. Правила з дуже низькою підтримкою базуються на рідкісних подіях.

Достовірність (Confidence) правила X → Y – умовна ймовірність появи Y за умови, що з'явився X. Це міра точності, або сили правила. Висока достовірність означає, що правило виконується часто.

Ліфт (Lift) правила X → Y – показує, у скільки разів ймовірність зустріти Y за умови X вища за ймовірність зустріти Y випадково. Якщо ліфт > 1, зв'язок позитивний і значущий; якщо ліфт = 1, зв'язку немає; якщо ліфт < 1, зв'язок негативний.

Леверидж (Leverage) – різниця між частотою спільної появи X та Y і частотою, яку можна було б очікувати, якби вони були незалежними. Чим більший леверидж, тим правило є більш значущим.

Алгоритм Apriori є класичним алгоритмом для пошуку асоціативних правил. Він базується на ключовій властивості: якщо набір елементів є частим (тобто має підтримку вищу за заданий поріг), то всі його підмножини також є частими. І навпаки, якщо набір є нечастим, то всі його надмножини також будуть нечастими. Це дозволяє значно скоротити простір пошуку.

Алгоритм працює в два етапи:

Генерація частих наборів: ітеративно, збільшуючи розмір наборів, знаходяться всі набори елементів, підтримка яких перевищує заданий мінімальний поріг (min_support).

Генерація правил: зі знайдених частих наборів формуються всі можливі правила X → Y, і з них залишаються ті, чия достовірність перевищує заданий мінімальний поріг (min_confidence).

3. Завдання до лабораторної роботи

Для виконання роботи необхідно створити набір транзакційних даних. Можна взяти невеликий набір з 10–15 транзакцій, кожна з яких містить набір товарів (елементів). Товари можна позначати як рядками (наприклад, ‘хліб’, ‘молоко’, ‘масло’), так і цифровими кодами.

Завдання 1. Робота в Google Sheets. Створення набору транзакцій

Створіть нову таблицю в Google Sheets. Створіть набір даних, що містить 12–15 транзакцій. Кожна транзакція – це набір товарів. Товари можна позначати назвами (наприклад, "хліб", "молоко", "яйця", "сир", "ковбаса", "масло", "йогурт", "печиво", "кава", "чай").

Організуйте дані у форматі, зручному для подальшого експорту. Наприклад, один стовпець "Транзакція" з номерами, а інший стовпець "Товари" з переліком товарів через кому. Або можна створити матрицю "товар-транзакція" (one-hot encoding), де рядки – транзакції, стовпці – товари, а в комірках 1 або 0.

Завдання 2. Робота в Python. Підготовка даних та перший етап Apriori

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, mlxtend.frequent_patterns (функції apriori, association_rules).

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) у pandas.DataFrame. Якщо дані зберігаються у форматі списку товарів у стовпці, їх потрібно буде перетворити у формат one-hot encoding (матрицю з 0 та 1). Для цього можна скористатися функцією, яка розділяє рядки на списки та створює фіктивні змінні, або підготувати дані вже в цьому форматі.

Виведіть на екран отриману матрицю "транзакція-товар". Вона повинна мати вигляд, де рядки – транзакції, стовпці – товари, а значення – 1 (товар є в транзакції) або 0 (товару немає).

Застосуйте алгоритм Apriori для пошуку частих наборів елементів з мінімальним порогом підтримки min_support=0.3 (тобто набір має зустрічатися принаймні в 30% транзакцій). Використовуйте функцію apriori() з параметром use_colnames=True, щоб у результаті відображалися назви товарів, а не їх індекси.

Виведіть на екран отриманий датафрейм з частими наборами. Зверніть увагу на стовпець support.

Завдання 3. Робота в Python. Генерація асоціативних правил

З отриманих частих наборів згенеруйте асоціативні правила за допомогою функції association_rules(). Встановіть мінімальний поріг достовірності min_threshold=0.6 (або інше значення на ваш розсуд). Використовуйте метрику ‘confidence’.

Виведіть на екран отриманий датафрейм з правилами. Він міститиме стовпці antecedents (умова), consequents (наслідок), antecedent support, consequent support, support, confidence, lift, leverage, conviction.

Відфільтруйте правила, залишивши лише ті, що мають lift > 1.2 (або інше значення, що вказує на значущий позитивний зв’язок). Виведіть відфільтровані правила на екран.

Відсортуйте правила за спаданням ліфту (lift) або достовірності (confidence) та виведіть топ-5 найцікавіших правил.

Проаналізуйте отримані правила. Чи є серед них очевидні (наприклад, хліб → масло) та неочевидні? Які правила мають найвищий ліфт?

Завдання 4. Аналіз впливу порогових значень

Створіть цикл, який буде змінювати значення min_support від 0.1 до 0.5 з кроком 0.1. Для кожного значення запустіть пошук частих наборів (з тим самим min_confidence=0.6) та запишіть кількість знайдених правил.

Побудуйте графік залежності кількості знайдених правил від min_support. Зробіть висновок про те, як зміна порогу підтримки впливає на результат.

Аналогічно, зафіксувавши min_support=0.3, змінюйте min_confidence від 0.5 до 0.9 з кроком 0.1 та побудуйте графік залежності кількості правил від min_confidence.

Завдання 5. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду та вихідний набір транзакцій.

Наведіть фрагменти Python-коду для виконання алгоритму Apriori, генерації правил та фільтрації.

Наведіть таблицю з топ-5 найцікавішими правилами (з високим ліфтом) та дайте їм інтерпретацію. Наприклад, "Правило {хліб, масло} -> {молоко} має підтримку 0.25, достовірність 0.8 та ліфт 1.6, що означає, що в 25% транзакцій зустрічаються всі три товари, і якщо покупець купує хліб та масло, то з імовірністю 80% він також купить молоко, причому цей зв’язок у 1.6 раза сильніший за випадковий."

Проаналізуйте отримані графіки залежності кількості правил від порогових значень. Які практичні рекомендації щодо вибору min_support та min_confidence можна зробити на основі цих графіків?

Запропонуйте, як знайдені асоціативні правила можна було б використати в бізнесі (наприклад, для оптимізації розташування товарів у магазині, для створення рекомендаційної системи, для формування наборів товарів зі знижкою).

4. Контрольні питання

1. Сформулюйте задачу пошуку асоціативних правил. Наведіть приклади її застосування.
2. Що таке транзакція, предметний набір, асоціативне правило? З яких частин складається правило?
3. Поясніть зміст метрик підтримки (support) та достовірності (confidence). Як вони обчислюються?
4. Що показує ліфт (lift)? Як інтерпретувати значення ліфта, більші за 1, рівні 1 та менші за 1?
5. Для чого використовуються метрики леверидж (leverage) та поліпшення (improvement)?
6. У чому полягає сутність алгоритму Apriori? Яке ключове припущення дозволяє йому ефективно скорочувати простір пошуку?
7. Опишіть два основні етапи роботи алгоритму Apriori.
8. Як впливає вибір мінімального порогу підтримки (min_support) на результати пошуку асоціативних правил?
9. Як впливає вибір мінімального порогу достовірності (min_confidence) на результати?
10. Які функції бібліотеки mlxtend використовуються для пошуку асоціативних правил? Для чого призначені параметри use_colnames та min_threshold?

Лабораторна робота №10
Аналіз часових рядів та прогнозування в Python

1. Мета роботи

Формування практичних навичок аналізу часових рядів та прогнозування з використанням мови програмування Python (бібліотеки pandas, numpy, matplotlib, statsmodels.tsa) та табличного процесора Google Sheets. Оволодіння методами візуалізації часових рядів, декомпозиції на компоненти (тренд, сезонність, залишки), перевірки стаціонарності, побудови прогностичних моделей (ковзне середнє, експоненційне згладжування, ARIMA) та оцінки їх якості.

2. Короткі теоретичні відомості

Часовий ряд (time series) – це послідовність числових значень деякої ознаки, впорядкована в хронологічному порядку. Кожне окреме значення називається рівнем ряду. Часові ряди є основою для прогнозування в економіці, фінансах, метеорології, соціології та багатьох інших галузях.

Основна мета аналізу часових рядів – виявити структуру ряду та побудувати модель, яка дозволить прогнозувати майбутні значення. У структурі часового ряду виділяють такі компоненти:

Тренд (Trend) – довгострокова тенденція зміни показника (зростання, спадання, стабільність).

Сезонна компонента (Seasonal) – регулярні коливання з фіксованим періодом (день, тиждень, місяць, квартал), зумовлені сезонними факторами.

Циклічна компонента (Cyclical) – коливання з тривалим періодом (більше року), пов'язані з економічними циклами.

Випадкова компонента (Residual, Noise) – залишок після виділення регулярних компонент, який є випадковим шумом.

Процес виділення компонент називається декомпозицією часового ряду.

Важливою характеристикою часового ряду є стаціонарність. Стаціонарний ряд коливається навколо постійного середнього рівня з постійною дисперсією. Більшість моделей прогнозування (наприклад, ARIMA) вимагають, щоб ряд був стаціонарним. Для перевірки стаціонарності використовують, зокрема, тест Дікі-Фуллера (ADF test).

Для моделювання та прогнозування часових рядів існує багато методів:

Ковзне середнє (Moving Average) – простий метод згладжування, який замінює кожне значення ряду середнім за кілька попередніх періодів.

Експоненційне згладжування (Exponential Smoothing) – надає більшої ваги останнім спостереженням.

Модель ARIMA (Autoregressive Integrated Moving Average) – потужна статистична модель, яка поєднує авторегресію (AR), інтегрування (I) для досягнення стаціонарності та ковзне середнє (MA).

Оцінка якості прогнозу здійснюється за допомогою метрик, що показують, наскільки прогнозні значення відхиляються від фактичних:

MAE (Mean Absolute Error) – середня абсолютна похибка.

MSE (Mean Squared Error) – середньоквадратична похибка.

RMSE (Root Mean Squared Error) – корінь з MSE (має ту саму розмірність, що й вихідні дані).

MAPE (Mean Absolute Percentage Error) – середня абсолютна похибка у відсотках.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно обрати або створити набір даних, що є часовим рядом. Можна скористатися відкритими джерелами (наприклад, дані з FRED, Yahoo Finance, Kaggle) або створити власний набір даних (наприклад, щомісячні продажі, щоденна температура, кількість відвідувачів сайту). Ряд має містити не менше 50–100 спостережень.

Завдання 1. Робота в Google Sheets. Первинний аналіз та візуалізація

Створіть нову таблицю в Google Sheets. Внесіть ваш часовий ряд: перший стовпець – дата або номер періоду (t), другий стовпець – значення показника (y).

Побудуйте лінійний графік (точкову діаграму з лініями) часового ряду. Додайте заголовок, підписи осей ("Період", "Значення").

Візуально проаналізуйте графік. Чи можна побачити на ньому тренд (зростання/спадання)? Чи є сезонні коливання? Якщо так, то який приблизно період?

Розрахуйте для ряду основні статистичні показники: середнє, медіану, мінімум, максимум, стандартне відхилення за допомогою вбудованих функцій Google Sheets.

Завдання 2. Робота в Python. Декомпозиція часового ряду

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, statsmodels.tsa.seasonal, statsmodels.tsa.stattools, statsmodels.tsa.arima.model, sklearn.metrics.

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) у pandas.DataFrame. Переконайтеся, що стовпець з датами має тип datetime, і встановіть його як індекс.

Побудуйте графік ряду за допомогою matplotlib.

Виконайте декомпозицію часового ряду на тренд, сезонну та залишкову компоненти за допомогою функції seasonal_decompose з бібліотеки statsmodels. Якщо ваш ряд має квартальну сезонність, встановіть period=4, якщо місячну – period=12. Якщо період невідомий, спробуйте визначити його візуально або за допомогою автокореляції.

Виведіть на екран отримані компоненти (тренд, сезонну, залишки) та побудуйте графік, що показує всі компоненти (оригінальний ряд, тренд, сезонність, залишки) на окремих підграфіках.

Завдання 3. Перевірка стаціонарності

Виконайте тест Дікі-Фуллера (ADF test) для вашого часового ряду за допомогою функції adfuller з statsmodels.tsa.stattools.

Виведіть на екран результати тесту: ADF-статистику, p-значення, критичні значення.

Проаналізуйте p-значення. Якщо p-значення менше за 0.05, ряд є стаціонарним. Якщо ні, ряд потребує диференціювання.

Якщо ряд нестаціонарний, створіть ряд перших різниць (за допомогою методу .diff().dropna()) та повторіть тест Дікі-Фуллера. Зробіть висновок про те, чи став ряд стаціонарним після взяття перших різниць.

Завдання 4. Побудова прогностичних моделей

Розділіть дані на навчальну (80%) та тестову (20%) вибірки. Навчальна вибірка використовуватиметься для побудови моделі, тестова – для оцінки якості прогнозу.

Модель ковзного середнього: Реалізуйте просту модель ковзного середньо для прогнозування. Наприклад, прогноз на наступний період дорівнює середньому за останні 3, 6 або 12 періодів. Виберіть вікно (window) на ваш розсуд. Обчисліть прогнозні значення для тестової вибірки.

Модель експоненційного згладжування: Використовуйте SimpleExpSmoothing з statsmodels для побудови моделі простого експоненційного згладжування. Навчіть модель на навчальній вибірці та зробіть прогноз на довжину тестової вибірки.

Модель ARIMA: Використовуйте ARIMA з statsmodels для побудови моделі. Параметри (p, d, q) можна підібрати експериментально або за допомогою аналізу автокореляційної (ACF) та часткової автокореляційної (PACF) функцій. Навчіть модель на навчальній вибірці та зробіть прогноз.

Завдання 5. Оцінка якості прогнозів

Для кожної з трьох моделей обчисліть метрики якості на тестовій вибірці:

MAE (mean_absolute_error)

MSE (mean_squared_error)

RMSE (корінь з MSE)

MAPE (обчисліть вручну, використовуючи формулу: np.mean(np.abs((y_test - y_pred) / y_test)) * 100)

Зведіть результати в таблицю (pandas DataFrame) та виведіть на екран.

Побудуйте графік, на якому відобразіть:

Фактичні значення ряду (повністю).

Прогнозні значення від кожної моделі на тестовій вибірці (різними кольорами).

Додайте легенду, щоб розрізнити моделі.

Проаналізуйте графік та метрики. Яка модель дала найкращий прогноз для вашого ряду?

Завдання 6. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду для декомпозиції ряду, тесту Дікі-Фуллера та побудови моделей прогнозування.

На основі декомпозиції опишіть структуру вашого часового ряду: чи є тренд, чи є сезонність, який її період.

Поясніть результати тесту Дікі-Фуллера. Чи є ряд стаціонарним? Якщо ні, що було зроблено для досягнення стаціонарності?

Проаналізуйте отримані метрики якості. Для якої моделі вони найкращі? Чи збігається це з вашими очікуваннями?

Зробіть висновок про те, чи можна використовувати побудовані моделі для прогнозування на практиці.

4. Контрольні питання

1. Що таке часовий ряд? З яких компонент він може складатися?
2. Поясніть різницю між трендом, сезонною та циклічною компонентами.
3. Що таке стаціонарний часовий ряд? Чому стаціонарність важлива для багатьох моделей прогнозування?

4. Як перевірити ряд на стаціонарність за допомогою тесту Дікі-Фуллера? Як інтерпретувати p-значення?
5. Що таке декомпозиція часового ряду? Для чого вона використовується?
6. Опишіть принцип роботи моделі ковзного середнього (Moving Average). Які її переваги та недоліки?
7. У чому суть експоненційного згладжування? Чим воно відрізняється від простого ковзного середнього?
8. Що таке модель ARIMA? Що означають параметри p, d, q?
9. Як оцінити якість прогнозу? Назвіть основні метрики (MAE, MSE, RMSE, MAPE) та поясніть їхній зміст.
10. Як побудувати прогноз за допомогою моделі ARIMA в Python (бібліотека statsmodels)?

Лабораторна робота №11
Нейронні мережі в Python: класифікація та прогнозування

1. Мета роботи

Формування практичних навичок створення, навчання та використання штучних нейронних мереж для розв'язання задач класифікації та прогнозування в середовищі Python з використанням бібліотек TensorFlow/Keras та scikit-learn. Оволодіння методами підготовки даних, побудови архітектури мережі, вибору функцій активації, налаштування процесу навчання та оцінки якості отриманих моделей.

2. Короткі теоретичні відомості

Штучні нейронні мережі (ШНМ) – це обчислювальні системи, натхненні будовою та принципами роботи біологічних нейронних мереж. Вони здатні до навчання, узагальнення та виявлення складних нелінійних закономірностей у даних, що робить їх ефективними для задач класифікації, прогнозування, кластеризації та розпізнавання образів.

Базовим елементом нейронної мережі є штучний нейрон, який виконує перетворення вхідних сигналів у вихідний:

Отримує набір вхідних значень (features).

Кожен вхід має свою вагу, яка визначає його важливість.

Обчислюється зважена сума входів (часто з додаванням зміщення, bias).

Результат передається через функцію активації, яка вносить нелінійність і формує вихідний сигнал.

Нейрони об'єднуються в шари. Розрізняють:

Вхідний шар (input layer): отримує вихідні дані.

Приховані шари (hidden layers): виконують основну обробку інформації, виявляючи абстрактні ознаки. Чим більше прихованих шарів, тим мережа "глибша".

Вихідний шар (output layer): формує кінцевий результат (наприклад, імовірності належності до класів).

Найпоширенішим типом архітектури є мережа прямого поширення (feedforward neural network) або багатошаровий перцептрон (MLP), де сигнал поширюється лише в одному напрямку – від входу до виходу.

Навчання нейронної мережі – це процес налаштування ваг та зміщень для мінімізації помилки на навчальних даних. Основним алгоритмом є зворотне поширення помилки (backpropagation), який обчислює градієнт функції втрат за вагами та оновлює їх у напрямку, протилежному градієнту (метод градієнтного спуску). Процес навчання організований у епохи (epochs) – один прохід усіх навчальних даних через мережу.

3. Завдання до лабораторної роботи

Робота складається з двох незалежних частин: класифікація (на прикладі набору даних Iris) та прогнозування часового ряду.

Завдання 1. Класифікація ірисів за допомогою нейронної мережі

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, seaborn, sklearn.datasets, sklearn.model_selection, sklearn.preprocessing, tensorflow (або keras).

Завантажте набір даних Iris за допомогою datasets.load_iris(). Створіть DataFrame для ознайомлення з даними.

Розділіть дані на ознаки (X) та цільові мітки (y). Виконайте масштабування ознак за допомогою StandardScaler (це важливо для нейронних мереж).

Виконайте one-hot кодування цільових міток (перетворення в категоріальний формат) за допомогою to_categorical з Keras.

Розділіть дані на навчальну (70%) та тестову (30%) вибірки за допомогою train_test_split.

Побудуйте модель нейронної мережі за допомогою Keras Sequential API:

Вхідний шар: кількість нейронів дорівнює кількості ознак (4).

Перший прихований шар: 10 нейронів, функція активації ‘relu’.

Другий прихований шар: 8 нейронів, функція активації ‘relu’.

Вихідний шар: 3 нейрони (за кількістю класів), функція активації ‘softmax’ (для багатокласової класифікації).

Скомпілюйте модель: оптимізатор ‘adam’, функція втрат ‘categorical_crossentropy’, метрика ‘accuracy’.

Навчіть модель на навчальних даних. Встановіть кількість епох (наприклад, 50) та розмір пакету (batch size). Використовуйте валідаційну вибірку (можна відокремити частину навчальної).

Побудуйте графіки історії навчання: функція втрат та точність на навчальній та валідаційній вибірках залежно від епохи.

Оцініть якість моделі на тестовій вибірці: обчисліть точність (evaluate), побудуйте матрицю помилок та класифікаційний звіт.

Додаткове завдання: Виберіть новий об’єкт (наприклад, [5.1, 3.5, 1.4, 0.2]) та за допомогою навченої моделі передбачте його клас.

Завдання 2. Прогнозування часового ряду за допомогою нейронної мережі

У цій частині ми будемо прогнозувати наступне значення часового ряду на основі кількох попередніх. Використайте дані з попередньої лабораторної роботи (Лекція 10) або згенеруйте простий ряд з трендом та сезонністю.

Підготуйте часовий ряд. Якщо використовуєте власні дані, завантажте їх у DataFrame.

Створіть функцію для перетворення часового ряду в задачу навчання з учителем. Функція має приймати ряд (list або array) та параметр window_size (кількість попередніх значень для прогнозу). Вона повертає два масиви:

X: матриця, де кожен рядок – це вектор з window_size попередніх значень.

y: вектор, де кожен елемент – це наступне значення (яке ми хочемо передбачити).

Виберіть window_size (наприклад, 5 або 10). Застосуйте функцію для створення X та y.

Розділіть дані на навчальну (80%) та тестову (20%) вибірки. Для часових рядів важливо зберігати порядок!

Масштабуйте дані (наприклад, за допомогою MinMaxScaler), щоб вони були в діапазоні [0, 1]. Навчіть скейлер на навчальній вибірці та застосуйте до навчальної та тестової.

Побудуйте модель нейронної мережі для регресії:

Вхідний шар: window_size нейронів.

Прихований шар: 10 нейронів, активація ‘relu’.

Прихований шар: 10 нейронів, активація ‘relu’.

Вихідний шар: 1 нейрон, активація ‘linear’ (для регресії).

Скомпілюйте модель: оптимізатор ‘adam’, функція втрат ‘mse’ (mean squared error).

Навчіть модель на навчальних даних. Слідкуйте за значенням loss.

Зробіть прогноз на тестовій вибірці.

Виконайте зворотне масштабування прогнозованих та фактичних значень (використовуючи inverse_transform скейлера).

Обчисліть метрики якості прогнозу: MAE, MSE, RMSE, MAPE.

Побудуйте графік, на якому відобразіть фактичні значення тестової вибірки та прогнозовані.

Завдання 3. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду (якщо використовували її для зберігання даних).

Наведіть фрагменти Python-коду для побудови обох нейронних мереж, їх навчання та оцінки.

Для задачі класифікації: проаналізуйте графіки історії навчання. Чи було перенавчання? Наскільки добре модель працює на тестових даних?

Для задачі прогнозування: проаналізуйте отримані метрики. Наскільки точним є прогноз? Чи добре модель вловлює загальну тенденцію ряду?

Зробіть висновки про застосовність нейронних мереж для розв'язання задач класифікації та прогнозування.

4. Контрольні питання

1. Що таке штучний нейрон? З яких основних компонентів він складається?
2. Які функції активації ви знаєте? Для чого вони використовуються? Поясніть різницю між сигмоїдою, гіперболічним тангенсом та ReLU.
3. Що таке багатошаровий перцептрон (MLP)? Яка роль прихованих шарів?
4. Поясніть процес навчання нейронної мережі. Що таке зворотне поширення помилки (backpropagation)?
5. Що таке епоха (epoch) та розмір пакету (batch size) в контексті навчання нейронних мереж?
6. Як уникнути перенавчання (overfitting) нейронної мережі? Назвіть кілька методів.
7. Чому перед подачею даних у нейронну мережу їх часто масштабують (нормалізують)?
8. Як побудувати нейронну мережу для задачі багатокласової класифікації? Яку функцію активації використовують на вихідному шарі і чому?
9. Як побудувати нейронну мережу для задачі прогнозування (регресії)? Яку функцію втрат зазвичай використовують?
10. Які бібліотеки Python використовуються для створення та навчання нейронних мереж? Яке призначення функцій Dense, Sequential, compile, fit в Keras?

Лабораторна робота №12
Кореляційний та дисперсійний аналіз даних у Python

1. Мета роботи

Формування практичних навичок виявлення та оцінки статистичних зв'язків між змінними з використанням кореляційного аналізу (коефіцієнти Пірсона та Спірмена) та однофакторного дисперсійного аналізу (ANOVA) в середовищі Python. Оволодіння методами візуалізації кореляцій (діаграми розсіювання, теплові карти кореляцій) та інтерпретації отриманих результатів.

2. Короткі теоретичні відомості

Виявлення зв'язків між ознаками є одним із ключових завдань інтелектуального аналізу даних. Розуміння того, як пов'язані між собою різні

характеристики об’єктів, дозволяє будувати прогностичні моделі, зменшувати розмірність даних та глибше розуміти досліджувані процеси.

Кореляційний аналіз використовується для виявлення та оцінки тісноти зв’язку між двома кількісними змінними. Основною мірою є коефіцієнт кореляції, який змінюється в межах від -1 до 1:

Знак коефіцієнта вказує на напрямок зв’язку: додатний – прямий (зі зростанням однієї змінної зростає інша), від’ємний – обернений.

Абсолютна величина вказує на силу (тісноту) зв’язку. Чим ближче значення до 1 або -1, тим сильніший зв’язок.

Найпоширенішим є коефіцієнт кореляції Пірсона, який вимірює силу лінійного зв’язку між змінними. Він є параметричним методом, тобто передбачає нормальний розподіл даних. Якщо зв’язок є нелінійним, або дані не підпорядковуються нормальному розподілу, використовують непараметричні аналоги, зокрема коефіцієнт рангової кореляції Спірмена. Він базується на порівнянні не самих значень, а їхніх рангів.

Для візуальної оцінки кореляційного зв’язку використовують діаграму розсіювання (scatter plot). Форма "хмари" точок дозволяє візуально визначити наявність, напрямок та форму зв’язку (лінійний, нелінійний). Для одночасного відображення кореляцій між багатьма змінними зручно використовувати теплову карту кореляцій (correlation heatmap).

Дисперсійний аналіз (ANOVA) використовується для перевірки гіпотези про вплив однієї або кількох категоріальних змінних (факторів) на одну кількісну змінну (відгук). Найпростішим є однофакторний дисперсійний аналіз, який порівнює середні значення відгуку в кількох групах, сформованих різними рівнями фактора. Ідея методу полягає в порівнянні міжгрупової дисперсії (спричиненої впливом фактора) та внутрішньогрупової дисперсії (спричиненої випадковими факторами). Для порівняння використовується F-критерій. Якщо розраховане значення F-статистики перевищує критичне, або відповідне p-значення менше за обраний рівень значущості (зазвичай 0.05), нульова гіпотеза (про відсутність впливу) відхиляється.

3. Завдання до лабораторної роботи

Робота складається з двох частин. Для виконання завдань необхідно створити набір даних, що містить як кількісні, так і категоріальні змінні. Можна скористатися вбудованими наборами даних (наприклад, iris з seaborn або tips), або створити власний невеликий набір даних (наприклад, інформація про студентів: години навчання, середній бал, стать, форма навчання).

Завдання 1. Робота в Google Sheets. Первинний аналіз даних

Створіть нову таблицю в Google Sheets. Внесіть ваш набір даних. Він має містити:

Не менше 20 рядків (спостережень).

Не менше 3 числових змінних.

Не менше 1 категоріальної змінної з 2-3 рівнями (наприклад, "Стать", "Тип клієнта", "Група").

Для кожної пари числових змінних побудуйте діаграму розсіювання (точкову діаграму). Для цього можна створити окремі діаграми або скористатися інструментом "Діаграма" з вибором "Точкова".

Візуально проаналізуйте отримані діаграми. Чи можна побачити на них лінійний зв'язок? Якщо так, то він прямий чи обернений?

За допомогою вбудованої функції =CORREL() обчисліть коефіцієнт кореляції Пірсона для кожної пари числових змінних. Результати зведіть у невелику таблицю (матрицю кореляцій).

Порівняйте отримані числові значення з візуальними висновками з п.3.

Завдання 2. Робота в Python. Кореляційний аналіз

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, seaborn, scipy.stats.

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) у pandas.DataFrame.

Виберіть дві числові змінні, які, на вашу думку, можуть бути пов'язані. Побудуйте для них діаграму розсіювання за допомогою matplotlib або seaborn.

Обчисліть для цієї пари змінних:

Коефіцієнт кореляції Пірсона за допомогою scipy.stats.pearsonr(). Виведіть значення коефіцієнта та p-значення.

Коефіцієнт рангової кореляції Спірмена за допомогою scipy.stats.spearmanr(). Виведіть значення коефіцієнта та p-значення.

Порівняйте отримані значення. Чи збігаються вони? Якщо ні, спробуйте пояснити, чому (можливо, зв'язок не є суто лінійним).

Для всіх числових змінних вашого набору даних обчисліть матрицю кореляцій Пірсона за допомогою методу df.corr(). Виведіть її на екран.

Візуалізуйте отриману матрицю кореляцій за допомогою теплової карти (seaborn.heatmap). Налаштуйте відображення значень коефіцієнтів у комірках.

Завдання 3. Робота в Python. Однофакторний дисперсійний аналіз (ANOVA)

Для виконання ANOVA вам знадобиться категоріальна змінна (фактор) та числова змінна (відгук). Виберіть відповідну пару з вашого набору даних. Наприклад, "Стать" (фактор) та "Зріст" або "Вага" (відгук); або "Тип клієнта" та "Сума покупки".

Сформулюйте нульову та альтернативну гіпотези:

H_0: середні значення відгуку в усіх групах рівні (фактор не впливає).

H_1: середні значення відгуку в групах відрізняються (фактор впливає).

Використовуючи бібліотеку scipy.stats, виконайте однофакторний дисперсійний аналіз за допомогою функції f_oneway(). Для цього вам потрібно передати функції окремі масиви значень для кожної групи.

Виведіть на екран значення F-статистики та p-значення.

Проаналізуйте p-значення. Якщо воно менше за 0.05, відхиліть нульову гіпотезу та зробіть висновок про статистично значущий вплив фактора на відгук. В іншому випадку – прийміть нульову гіпотезу.

Для візуалізації результатів побудуйте ящик з вусами (boxplot) за допомогою seaborn.boxplot, де вісь X – фактор, вісь Y – відгук. Це дозволить побачити розподіл значень у кожній групі.

Завдання 4. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду для обчислення кореляцій, побудови теплової карти та виконання ANOVA.

Проаналізуйте отриману теплову карту кореляцій. Які змінні мають сильний позитивний зв'язок? Які – сильний обернений? Чи є змінні, що практично не корелюють?

На основі результатів ANOVA зробіть висновок про вплив обраного фактора. Чи підтверджується ваша гіпотеза? Як візуалізація (boxplot) ілюструє цей висновок?

Підсумуйте, які методи аналізу зв'язків були використані та які нові знання про дані вони дозволили отримати.

4. Контрольні питання

1. Які основні питання про зв'язок між змінними допомагає з'ясувати кореляційний аналіз?
2. Поясніть різницю між коефіцієнтом кореляції Пірсона та коефіцієнтом Спірмена. У яких випадках доцільніше застосовувати кожен з них?
3. Що означають знак та абсолютна величина коефіцієнта кореляції?
4. Як за допомогою діаграми розсіювання можна візуально оцінити наявність, силу та напрямок кореляційного зв'язку?
5. Для чого використовують теплову карту кореляцій? Яку інформацію вона надає?
6. Сформулюйте мету дисперсійного аналізу (ANOVA). Яке основне питання він дозволяє вирішити?
7. Поясніть логіку однофакторного дисперсійного аналізу. Що порівнюється за допомогою F-критерію?
8. Як інтерпретувати результати ANOVA? Що таке p-значення і як воно використовується для прийняття рішення про вплив фактора?
9. Як побудувати ящик з вусами (boxplot) в Python (seaborn) і яку інформацію про розподіл даних він надає?
10. Які функції бібліотек pandas, scipy.stats та seaborn використовувалися в роботі для кореляційного та дисперсійного аналізу?

Лабораторна робота №13
Регресійний аналіз даних. Лінійна регресія в Python

1. Мета роботи

Формування практичних навичок побудови, аналізу та оцінки якості лінійних регресійних моделей з використанням мови програмування Python (бібліотеки pandas, numpy, matplotlib, seaborn, scikit-learn, statsmodels) та табличного процесора Google Sheets. Оволодіння методами визначення параметрів регресійної моделі, перевірки її адекватності, оцінки значущості коефіцієнтів та прогнозування на основі побудованої моделі.

2. Короткі теоретичні відомості

Регресійний аналіз – це сукупність статистичних методів, які дозволяють вивчати залежність між однією або кількома незалежними змінними (факторами, предикторами) та однією залежною змінною (відгуком). Основна мета регресійного аналізу – побудувати математичну модель, яка описує цю залежність, та використовувати її для прогнозування значень відгуку для нових значень факторів.

Найпростішим і найпоширенішим видом є лінійна регресія, яка описує зв'язок між змінними за допомогою рівняння прямої лінії. Для випадку однієї незалежної змінної (проста лінійна регресія) рівняння має вигляд:

$y = a_0 + a_1 \cdot x + \varepsilon$

де:

y – залежна змінна;

x – незалежна змінна;

a_0 – вільний член (intercept), значення y при x=0;

a_1 – коефіцієнт регресії (slope), який показує, на скільки одиниць у середньому зміниться y при зміні x на одну одиницю;

ε – випадкова похибка (залишки).

Для знаходження коефіцієнтів a_0 та a_1 найчастіше використовують метод найменших квадратів. Він полягає в мінімізації суми квадратів відхилень фактичних значень y від значень, передбачених моделлю (залишків).

Після побудови моделі необхідно оцінити її якість та адекватність. Для цього використовують:

Коефіцієнт детермінації R^2 – показує, яку частку варіації залежної змінної пояснює побудована модель. Значення R^2 близьке до 1 свідчить про високу точність моделі.

F-критерій – перевіряє статистичну значущість регресійної моделі в цілому (гіпотезу про те, що всі коефіцієнти регресії одночасно дорівнюють нулю).

t-критерій – перевіряє статистичну значущість окремих коефіцієнтів регресії (гіпотезу про те, що конкретний коефіцієнт дорівнює нулю).

Аналіз залишків – дослідження залишків (різниць між фактичними та передбаченими значеннями) на предмет відповідності припущенням методу найменших квадратів (нормальність, гомоскедастичність, незалежність).

3. Завдання до лабораторної роботи

Для виконання роботи необхідно створити набір даних, що містить залежну (y) та одну незалежну (x) числові змінні. Можна скористатися відкритими джерелами або створити власний набір даних (наприклад, залежність витрат на рекламу від продажів, залежність кількості годин навчання від отриманого балу, залежність площі квартири від її ціни тощо). Набір має містити не менше 15-20 спостережень.

Завдання 1. Робота в Google Sheets. Первинний аналіз та візуалізація

Створіть нову таблицю в Google Sheets. Внесіть ваш набір даних: перший стовпець – значення незалежної змінної X, другий стовпець – значення залежної змінної Y.

Побудуйте точкову діаграму (діаграму розсіювання) для вашого набору даних. Додайте заголовок, підписи осей.

Візуально проаналізуйте графік. Чи можна припустити наявність лінійного зв’язку між X та Y? Якщо так, то він прямий чи обернений?

За допомогою інструменту "Додати лінію тренду" побудуйте лінійну регресійну модель. Налаштуйте відображення рівняння лінії тренду та коефіцієнта детермінації R^2 на діаграмі.

Запишіть отримане рівняння регресії та значення R^2. Як ви інтерпретуєте отримане рівняння?

Завдання 2. Робота в Python. Побудова моделі лінійної регресії

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, seaborn, sklearn.linear_model, sklearn.metrics, statsmodels.api.

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) у pandas.DataFrame.

Побудуйте діаграму розсіювання за допомогою matplotlib або seaborn, щоб ще раз візуально оцінити зв’язок.

Створіть об’єкт лінійної регресії з sklearn.linear_model.LinearRegression. Навчіть модель на ваших даних (X та y). Оскільки sklearn очікує, що X буде двовимірним масивом, використайте X.values.reshape(-1, 1) або X.values.reshape(-1, 1).

Отримайте коефіцієнти моделі: вільний член (intercept_) та коефіцієнт при X (coef_). Виведіть їх на екран. Запишіть рівняння регресії.

Порівняйте отримані коефіцієнти з тими, що ви отримали в Google Sheets.

Завдання 3. Оцінка якості моделі

Використовуючи навчену модель, зробіть передбачення для всіх значень X: y_pred = model.predict(X).

Обчисліть коефіцієнт детермінації R^2 за допомогою функції r2_score з sklearn.metrics. Виведіть його значення. Порівняйте зі значенням з Google Sheets.

Обчисліть середньоквадратичну похибку (MSE) та корінь із середньоквадратичної похибки (RMSE). Ці метрики показують абсолютну величину помилки моделі.

Використовуючи бібліотеку statsmodels, побудуйте більш детальну регресійну модель. Для цього спочатку додайте константу до незалежної змінної за допомогою sm.add_constant(X), а потім створіть та навчіть модель sm.OLS(y, X_with_const). Викличте метод .fit() та виведіть звіт .summary().

Проаналізуйте звіт statsmodels. Зверніть увагу на:

R^2 та скоригований R^2.

F-статистику та її p-значення (Prob (F-statistic)). Якщо p-значення менше 0.05, модель в цілому є значущою.

t-статистики та p-значення для коефіцієнтів (P>|t|). Якщо p-значення для коефіцієнта при X менше 0.05, він є статистично значущим.

Завдання 4. Візуалізація результатів та прогнозування

Побудуйте графік, на якому відобразіть:

Вихідні дані (точки діаграми розсіювання).

Лінію регресії, побудовану за моделлю (можна побудувати, обчисливши передбачені значення для всього діапазону X або використовуючи функцію abline з seaborn, або просто з'єднавши точки y_pred).

Додайте легенду, заголовок та підписи осей.

Виберіть нове значення незалежної змінної X_new, яке не входило у вихідний набір даних, але знаходиться в межах діапазону X.

За допомогою навченої моделі (з sklearn) спрогнозуйте для нього значення Y_new. Виведіть результат на екран.

Додайте цю точку на побудований графік, позначивши її, наприклад, червоним маркером іншого розміру.

Завдання 5. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду для побудови моделі, оцінки її якості та прогнозування.

Запишіть рівняння регресії, отримане в Python, та поясніть зміст коефіцієнтів (на що вказує знак і величина коефіцієнта при X).

Проаналізуйте отримані показники якості моделі (R^2, F-статистику, p-значення). Наскільки добре модель описує дані? Чи є вона статистично значущою?

Зробіть прогноз для нового значення X_new та прокоментуйте його. Чи можна вважати цей прогноз надійним? Чому?

4. Контрольні питання

1. Сформулюйте основну мету регресійного аналізу. Чим він відрізняється від кореляційного аналізу?
2. Поясніть суть методу найменших квадратів для знаходження параметрів лінійної регресії.
3. Як інтерпретуються коефіцієнти рівняння простої лінійної регресії $y = a_0 + a_1 \cdot x$?
4. Що показує коефіцієнт детермінації R^2? Які значення вважаються задовільними?
5. Для чого використовуються F-критерій та t-критерій в регресійному аналізі? Як інтерпретувати їхні p-значення?
6. Що таке залишки (residuals) регресійної моделі? Чому їх аналіз є важливим?
7. Як обчислити MSE та RMSE і що вони характеризують?
8. Як побудувати лінійну регресійну модель за допомогою бібліотеки scikit-learn? Які основні методи (fit, predict, score) використовуються?
9. Яку додаткову інформацію про модель надає звіт statsmodels.api.OLS у порівнянні з scikit-learn?
10. Як, маючи рівняння регресії, зробити прогноз для нового значення незалежної змінної? Які обмеження має таке прогнозування?

Лабораторна робота №14
Факторний аналіз даних. Метод головних компонент (PCA) у Python

1. Мета роботи

Формування практичних навичок проведення факторного аналізу даних з використанням методу головних компонент (Principal Component Analysis, PCA) в середовищі Python. Оволодіння методами визначення оптимальної кількості факторів, інтерпретації факторних навантажень, обертання факторів та візуалізації результатів факторного аналізу.

2. Короткі теоретичні відомості

Факторний аналіз – це сукупність методів, які дозволяють описати зв’язки між великою кількістю спостережуваних змінних за допомогою меншої кількості прихованих (латентних) змінних, які називаються факторами. Головна мета факторного аналізу – "стиснення" інформації, зменшення розмірності простору ознак шляхом об’єднання тісно корелюючих змінних в один узагальнений фактор. Це робить подальший аналіз простішим, а структуру даних – більш зрозумілою.

Розрізняють два основні типи факторного аналізу:

Дослідницький факторний аналіз (Exploratory Factor Analysis, EFA) – застосовується, коли ми не маємо попередніх гіпотез про структуру даних і хочемо її виявити.

Підтверджуючий факторний аналіз (Confirmatory Factor Analysis, CFA) – використовується для перевірки вже існуючої гіпотези про те, що дані мають певну факторну структуру.

Найпоширенішим методом факторного аналізу є метод головних компонент (Principal Component Analysis, PCA). Його основна ідея полягає в лінійному перетворенні вихідного простору ознак у новий простір меншої розмірності. Нові осі (головні компоненти) обираються таким чином, щоб вони були взаємно перпендикулярними (незалежними) і щоб перша компонента пояснювала максимально можливу частку загальної дисперсії даних, друга – максимально можливу частку з тієї, що залишилася, і так далі.

Ключовими поняттями факторного аналізу є:

Факторні навантаження (factor loadings) – коефіцієнти кореляції між вихідними змінними та виявленими факторами. Вони показують, наскільки тісно пов’язана кожна змінна з кожним фактором.

Власні значення (eigenvalues) – характеризують внесок кожної головної компоненти в загальну дисперсію. Чим більше власне значення, тим важливіша компонента.

Обертання факторів (factor rotation) – процедура, яка застосовується для отримання більш інтерпретованої матриці факторних навантажень. Найпоширенішим методом є варімакс-обертання (varimax), яке максимізує дисперсію квадратів навантажень, роблячи великі навантаження ще більшими, а малі – ще меншими.

3. Завдання до лабораторної роботи

Для виконання роботи необхідно обрати набір даних, що містить не менше 5-7 числових змінних. Можна скористатися вбудованими наборами даних (наприклад, iris з seaborn, wine з sklearn.datasets) або будь-яким іншим набором даних з відкритих джерел.

Завдання 1. Робота в Google Sheets. Первинний аналіз даних

Створіть нову таблицю в Google Sheets. Імпортуйте ваш набір даних або створіть його вручну. Переконайтеся, що дані представлені у вигляді таблиці, де рядки – об’єкти, а стовпці – числові змінні.

Побудуйте матрицю кореляцій для всіх змінних за допомогою функції =CORREL(). Для цього створіть окрему таблицю, де рядки та стовпці – назви змінних, а в комірках – відповідні коефіцієнти кореляції.

Візуально проаналізуйте матрицю кореляцій. Чи є змінні, які сильно корелюють між собою ($|r| > 0.7$)? Якщо так, це свідчить про наявність прихованих факторів.

Завдання 2. Робота в Python. Підготовка даних та стандартизація

Створіть новий Python-скрипт або Jupyter Notebook. Імпортуйте необхідні бібліотеки: pandas, numpy, matplotlib.pyplot, seaborn, sklearn.decomposition.PCA, sklearn.preprocessing.StandardScaler.

Зчитайте ваш набір даних з CSV-файлу (експортованого з Google Sheets) у pandas.DataFrame.

Оскільки PCA чутливий до масштабу змінних, виконайте стандартизацію даних за допомогою StandardScaler. Перетворіть дані так, щоб кожна змінна мала середнє 0 та стандартне відхилення 1. Збережіть стандартизовані дані в окремій змінній.

Завдання 3. Визначення оптимальної кількості головних компонент

Створіть об’єкт PCA без вказання кількості компонент (тобто PCA()). Навчіть його на стандартизованих даних за допомогою методу fit().

Отримайте власні значення (eigenvalues) з атрибуту explained_variance_. Виведіть їх на екран.

Побудуйте графік "кам’янистого осипу" (scree plot) – графік залежності власних значень від номера компоненти. Додайте на графік горизонтальну лінію на рівні 1 (критерій Кайзера).

Проаналізуйте графік. Скільки компонент мають власне значення більше за 1? Де знаходиться точка "перелому" графіка? На основі цього визначте оптимальну кількість факторів k.

Отримайте та виведіть на екран частку поясненої дисперсії для кожної компоненти (explained_variance_ratio_) та накопичену частку поясненої дисперсії (np.cumsum(explained_variance_ratio_)). Яка частка дисперсії пояснюється першими k компонентами?

Завдання 4. Побудова факторної моделі з обертанням

Створіть новий об’єкт PCA з кількістю компонент, рівною оптимальному k. Навчіть його на стандартизованих даних.

Отримайте матрицю факторних навантажень. Вона знаходиться в атрибуті components_, але це будуть навантаження для стандартизованих даних. Щоб отримати кореляції між змінними та компонентами, потрібно помножити components_ на квадратний корінь з відповідних власних значень: loadings = pca.components_.T * np.sqrt(pca.explained_variance_). Створіть з цієї матриці pandas.DataFrame, де рядки – вихідні змінні, а стовпці – головні компоненти.

Виведіть матрицю факторних навантажень на екран. Проаналізуйте її. Які змінні мають найбільші навантаження на першу компоненту? На другу?

Виконайте варімакс-обертання факторів. У scikit-learn для цього можна використати клас FactorAnalysis (який підтримує обертання) або скористатися функцією varimax з sklearn.utils.extmath (потрібно буде реалізувати або знайти). Альтернативно, для спрощення можна проінтерпретувати фактори без обертання, але зазначити, що обертання покращило б інтерпретацію.

Якщо ви змогли виконати обертання, порівняйте матриці навантажень до та після обертання. Чи стала вона більш контрастною?

Завдання 5. Інтерпретація факторів та візуалізація

На основі матриці факторних навантажень (бажано після обертання) дайте назви виявленим факторам. Наприклад, якщо на перший фактор мають високі навантаження змінні, пов’язані з розмірами квітки (довжина пелюстки, ширина пелюстки), його можна назвати "Розмір квітки".

Побудуйте теплову карту факторних навантажень за допомогою seaborn.heatmap. Це допоможе візуально оцінити, які змінні з якими факторами пов’язані.

Якщо кількість факторів дорівнює 2, побудуйте графік навантажень у просторі факторів. Для цього використайте plt.scatter або seaborn.scatterplot, де вісь X – навантаження на перший фактор, вісь Y – навантаження на другий фактор. Підпишіть точки назвами змінних.

Завдання 6. Звіт та інтерпретація

У звіті наведіть посилання на вашу Google-таблицю з доступом для перегляду.

Наведіть фрагменти Python-коду для визначення оптимальної кількості факторів, побудови матриці навантажень та візуалізації.

Наведіть матрицю кореляцій, отриману в Google Sheets, та коротко її проаналізуйте.

Покажіть графік "кам’янистого осипу" та поясніть, чому ви обрали саме k факторів. Вкажіть, яка частка дисперсії ними пояснюється.

Наведіть матрицю факторних навантажень (бажано після обертання). Для кожного фактора перелічіть змінні з найбільшими навантаженнями та дайте фактору змістовну назву.

Зробіть висновок про те, чи вдалося зменшити розмірність даних та чи мають виявлені фактори чітку інтерпретацію.

4. Контрольні питання

1. Сформулюйте основну мету факторного аналізу. Що таке латентна змінна (фактор)?
2. Поясніть різницю між дослідницьким та підтверджуючим факторним аналізом.
3. У чому полягає сутність методу головних компонент (PCA)? Як він дозволяє зменшити розмірність даних?

4. Що таке факторні навантаження? Яку інформацію вони надають для інтерпретації факторів?
5. Що таке власні значення (eigenvalues) і як вони використовуються для визначення оптимальної кількості факторів?
6. Поясніть, як за допомогою графіка "кам’янистого осипу" (scree plot) визначити оптимальну кількість головних компонент.
7. Для чого застосовується обертання факторів? Поясніть суть методу варімакс-обертання (varimax).
8. Як інтерпретувати матрицю факторних навантажень після обертання? Як дати назву фактору?
9. Чому перед застосуванням PCA необхідно стандартизувати дані?
10. Які функції бібліотеки sklearn.decomposition використовуються для PCA? Що означають атрибути explained_variance_, explained_variance_ratio_, components_?

ПІСЛЯМОВА

Ось ми й дісталися фінальної сторінки нашого практикуму. Чотирнадцять лабораторних робіт – чотирнадцять кроків, які провели нас довгим і, сподіваємося, захопливим шляхом – від першого знайомства з інтерфейсом Google Sheets та основами Python до створення власних нейронних мереж та факторних моделей. Цей шлях не був простим, адже інтелектуальний аналіз даних є однією з найскладніших, але водночас і найцікавіших дисциплін сучасної комп'ютерної науки.

Коли ми розпочинали цю роботу, нашою головною метою було створити не просто черговий збірник завдань, а практично орієнтований посібник, який стане для вас надійним провідником у реальному світі аналізу даних. Ми прагнули, щоб кожна лабораторна робота давала вам не лише знання, але й розуміння того, як застосувати ці знання на практиці – з використанням сучасних, затребуваних інструментів. Судити про те, наскільки нам це вдалося, звісно, вам.

Протягом нашої подорожі ви опанували фундаментальні процедури підготовки даних, навчилися «очищати» сирі дані від шуму та викидів, кодувати категоріальні ознаки та приводити числові показники до єдиного масштабу. Ви занурилися у світ кластеризації, навчившись знаходити приховані групи там, де їх не видно неозброєним оком, і розрізняти чіткі та нечіткі алгоритми групування. Ви опанували мистецтво класифікації, озброївшись цілим арсеналом методів – від простого правила найближчих сусідів до складних ансамблів дерев рішень та нейронних мереж. Ви навчилися прогнозувати майбутнє, аналізуючи часові ряди, та виявляти приховані зв'язки між подіями за допомогою асоціативних правил. І, нарешті, ви доторкнулися до найсучасніших технологій зменшення розмірності та факторного аналізу, які відкривають безмежні можливості для дослідження складних систем.

Однак важливо пам'ятати, що будь-який посібник, навіть найкращий, – це лише відправна точка. Справжнє опанування професії аналітика даних відбувається не в аудиторії, а в процесі розв'язання реальних задач, коли ви стикаєтеся з неструктурованими даними, нечіткими вимогами, обмеженнями в часі та необхідністю самостійно приймати складні рішення. Саме тоді вам знадобляться не лише знання алгоритмів, але й розвинене критичне мислення, інтуїція та, найголовніше, наполегливість. Не бійтеся помилятися – кожна помилка в аналізі даних, кожен неправильно побудований прогноз – це безцінний досвід, який робить вас кращими фахівцями.

Світ даних невпинно змінюється: з'являються нові алгоритми, зростають обчислювальні потужності, виникають нові виклики. Те, що сьогодні здається передовим краєм науки, завтра стане рутинним інструментом. Тому найважливіше, що ви маєте винести з цього курсу, – це не запам'ятовування конкретних функцій чи параметрів, а розуміння фундаментальних принципів, логіки мислення аналітика та, найголовніше, здатність навчатися самостійно.

Сподіваємося, що цей практикум став для вас не просто збірником завдань, а вірним супутником, який допоміг зробити перші, найважчі кроки в професії. Ми щиро бажаємо вам цікавих задач, несподіваних відкриттів, чистих даних та високої точності прогнозів. Пам’ятайте: за кожним числом, за кожною змінною стоять реальні люди, процеси та явища, а ваша робота робить світ трохи зрозумілішим, передбачуванішим і кращим.

Бажаємо успіхів, натхнення та нових звершень!

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Болюбаш Н. М. Інтелектуальний аналіз даних : навч. посіб. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2023. 320 с.
2. Гороховатський В. О., Творошенко І. С. Методи інтелектуального аналізу та оброблення даних : навч. посіб. / М-во освіти і науки України ; Харків. нац. ун-т радіоелектроніки. Харків : ХНУРЕ, 2021. 92 с. URL: https://openarchive.nure.ua/handle/document/15868
3. Іванчук Я. В., Месюра В. І., Яровий А. А., Манжілевський О. Д. Інтелектуальний аналіз даних та машинне навчання. Частина 1. Базові методи та засоби аналізу даних : навчальний посібник. Вінниця : ВНТУ, 2021. 69 с.
4. Ланде Д. В., Субач І. Ю., Бояринова Ю. Є. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки : навчальний посібник. Київ : ІСЗЗІ КПІ ім. Ігоря Сікорського», 2018. 300 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/5eb4cb50-3ef0-44d1-b35f-a80eade1554b/content
5. Литвин В. В., Пасічник В. В., Нікольський Ю. В. Аналіз даних та знань. Львів : Магнолія 2006, 2015. 276 с.
6. Лупан І. В. Інтелектуальний аналіз даних Data Mining : навчально-методичний посібник. Кропивницький : ФОП Піскова М. А., 2022. 112 с. URL: https://dspace.cusu.edu.ua/server/api/core/bitstreams/f8bbe456-1b5e-47b6-a80f-0a20b5d17264/content
7. Мельников О. С. Інтелектуальний аналіз даних : навч.-метод. посібник / Нац. техн. ун-т "Харків. політехн. ін-т". Харків : Impress, 2023. 196 с. URL: https://repository.kpi.kharkov.ua/handle/KhPI-Press/72877
8. Мельников О. С. Інтелектуальний аналіз даних : навч.-метод. посібник / Нац. техн. ун-т "Харків. політехн. ін-т". Харків : Impress, 2023. 196 с. URL: https://repository.kpi.kharkov.ua/handle/KhPI-Press/72877
9. Сергеєв-Горчинський О. О., Іщенко Г. В. Інтелектуальний аналіз даних: Комп’ютерний практикум : навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки та інформаційні технології», спеціалізацій «Інформаційні системи та технології проектування», «Системне проектування сервісів» / КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2018. 73 с.
10. Талах М. В., Дворжак В. В. Інтелектуальний аналіз даних. Частина 1. Чернівці : Технодрук, 2022. 367 с.

Навчальне видання

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

Методичні рекомендації

Укладачі: **Пархоменко** Олександр Юрійович

Тищенко Світлана Іванівна

Ємельянов Святослав Ігорович

Жебко Олександр Олегович

Богатєнкова Олександра Євгенівна

Формат 60х84 1/16. Ум. друк. арк. 4,0.
Тираж 20 прим. Зам. № ____

Надруковано у видавничому відділі
Миколаївського національного аграрного університету
54008, м. Миколаїв, вул. Георгія Гонгадзе, 9

Свідоцтво суб'єкта видавничої справи ДК № 4490 від 20.02.2013