

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни «Основи машинного навчання мовою Python»

для магістрів спеціальності 123 Комп'ютерна інженерія
денної форми навчання

Частина III

2024

Методичні вказівки до лабораторних робіт з дисципліни «Основи машинного навчання мовою Python» для магістрів спеціальності 123 Комп'ютерна інженерія, денної форми навчання. Частина III / Укл.: М.Б. Ільяшенко, Н.А. Куликовська – Запоріжжя: НУ «Запорізька політехніка», 2024. – 19 с.

Укладач: М.Б. Ільяшенко, к.т.н., доцент
Н.А. Куликовська, старший викладач

Рецензент: М.Ю. Тягунова, к.т.н., доцент

Відповідальний
за випуск: М.Б. Ільяшенко, к.т.н., доцент

Затверджено:
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 7
від 22 березня 2024р.

Рекомендовано до видання
НМК факультету КНТ
Протокол № 8
від 27 березня 2024 р.

ЗМІСТ

1	ЛАБОРАТОРНА РОБОТА №6 ОБРОБКА ПРИРОДНОЇ МОВИ NLP	4
1.1	Теоретичні відомості	4
1.2	Хід роботи.....	11
1.3	Зміст звіту	12
1.4	Контрольні питання	12
2	ЛАБОРАТОРНА РОБОТА №7 НАЇВНИЙ БАЙЄСОВСЬКИЙ КЛАСИФІКАТОР.....	13
2.1	Теоретичні відомості.....	13
2.2	Хід роботи	14
2.3	Зміст звіту.....	14
2.4	Контрольні питання.....	15
3	ЛАБОРАТОРНА РОБОТА №8 КЛАСТЕРНИЙ АНАЛІЗ	16
3.1	Теоретичні відомості.....	16
3.2	Хід роботи.....	17
3.3	Зміст звіту	17
3.4	Контрольні питання	17
	ЛІТЕРАТУРА.....	19

1 ЛАБОРАТОРНА РОБОТА №6 ОБРОБКА ПРИРОДНОЇ МОВИ NLP

Мета роботи: розвинути практичні вміння у кодуванні програм на Python для аналізу текстових даних за допомогою інструментів NLP.

1.1 Теоретичні відомості

1. Токенізація за пропозиціями

Токенізація (іноді – сегментація) за пропозиціями – це процес поділу письмової мови на пропозиції-компоненти. Ідея виглядає досить простою. В англійській та деяких інших мовах ми можемо виокремлювати пропозицію кожного разу, коли знаходимо певний знак пунктуації – точку.

Приклад:

Обираємо невеликий текст про настільну гру нарди:

Backgammon is one of the oldest known board games. Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East. It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

Щоб зробити токенизацію пропозицій за допомогою NLTK, можна скористатися методом `nltk.sent_tokenize`.

На виході ми отримуємо 3 окремі пропозиції:

Backgammon is one of the oldest known board games.

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.

2. Лексемізація словами

Лексемізація (іноді – сегментація) за словами – це процес поділу речень на слова-компоненти. В англійській та багатьох інших мовах, які використовують ту чи іншу версію латинського алфавіту, пробіл – це непоганий роздільник слів.

Тим не менш, можуть виникнути проблеми, якщо використовувати тільки прогалину - в англійській складові іменники пишуться по-різному і іноді через пропуск.

Приклад:

Візьмемо пропозиції з попереднього прикладу та застосуємо до них метод `nlk.word_tokenize`

Висновок:

['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known', 'board', 'games', '.']

['Its', 'history', 'can', 'be', 'traced', 'back', 'nearly', '5,000', 'years', 'to', 'archeological', 'discoveries', 'in', 'the', 'Middle', 'East', '.']

['It', 'is', 'a', 'two', 'player', 'game', 'where', 'each', 'player', 'has', 'fifteen', 'checkers', 'which', 'move', 'between', 'twenty-four', 'points', 'according', 'to', 'the', 'roll', 'of', 'two', 'dice', '.']

3. Лематизація та стемінг тексту

Зазвичай тексти містять різні граматичні форми однієї й тієї ж слова, і навіть можуть зустрічатися однокореневі слова. Лематизація і стемінг мають на меті привести всі зустрічаються словоформи до однієї, нормальної словникової форми.

Приклади:

Приведення різних словоформ до однієї:

dog, dogs, dog's, dogs' => dog

Те ж саме, але вже стосовно цілої пропозиції:

the boy's dogs are different sizes => the boy dog be differ size

Лематизація та стемінг – це окремі випадки нормалізації і вони відрізняються.

Стемінг - це грубий евристичний процес, який відрізає "зайве" від кореня слів, часто це призводить до втрати словотвірних суфіксів.

Лематизація – це тонший процес, який використовує словник і морфологічний аналіз, щоб у результаті привести слово до його канонічної форми – лемі.

Відмінність у тому, що стеммер (конкретна реалізація алгоритму стеммінгу – прим. перекладача) діє без знання контексту і, відповідно,

не розуміє різницю між словами, які мають різний зміст залежно від частини мови.

Приклади:

1. Слово good – це лема для слова better. Стеммер не побачить цей зв'язок, тому що тут потрібно звірятися зі словником.

2. Слово play – базова форма слова playing. Тут упораються і стемінг, і лематизація.

3. Слово meeting може бути як нормальною формою іменника, і формою дієслова to meet, залежно від контексту. На відміну від стеммінгу, лематизація спробує вибрати правильну лему, спираючись на контекст.

Висновок:

Stemmer: seen

Lemmatizer: see

Stemmer: drove

Lemmatizer: drive

4. Стоп-слова

Стоп-слова – це слова, які викидаються з тексту перед обробкою тексту. Коли ми застосовуємо машинне навчання до текстів, такі слова можуть додати багато шуму, тому необхідно позбавлятися нерелевантних слів.

Стоп-слова це зазвичай розуміють артиклі, вигуки, спілки і т.д., які не несуть смислового навантаження. При цьому треба розуміти, що немає універсального списку стоп-слів, все залежить від конкретного випадку.

NLTK має встановлений список стоп-слів. Перед першим використанням знадобиться його завантажити: `nltk.download("stopwords")`. Після завантаження можна імпортувати пакет `stopwords` і подивитися на самі слова:

Висновок:

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',

'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

Висновок:

['Backgammon', 'one', 'oldest', 'known', 'board', 'games', '.']

5. Регулярні вирази.

Регулярний вираз (регулярка, `regex`, `regex`) – це послідовність символів, що визначає шаблон пошуку. Наприклад:

- `.` – будь-який символ, окрім перекладу рядка;
- `\w` – один символ;
- `d` – одна цифра;
- `\s` – одна прогалина;
- `\W` – один НЕсимвол;
- `\D` – одна НЕцифра;
- `\S` – одна НЕЗАДАЧА;
- `[abc]` – знаходить будь-який із зазначених символів `match any of a, b, or c`;
- `[^abc]` – знаходить будь-який символ, крім зазначених;
- `[ag]` – знаходить символ у проміжку від `a` до `g`.

Регулярні вирази використовують зворотний слеш (`\`) для позначення спеціальних форм або щоб дозволити використання спецсимволів. Це суперечить використанню зворотного слеша в Python: наприклад, щоб буквально позначити зворотний слеш, необхідно написати `\\` як шаблон для пошуку, тому що регулярне вираз має виглядати як `\`, де кожен зворотний слеш повинен бути

екранований. Рішення – використовувати нотацію raw string для шаблонів пошуку; зворотні сліші не будуть особливим чином оброблятися, якщо використані з префіксом ‘r’. Таким чином, r”\n”це рядок з двома символами (‘\’ и ‘n’), а “\n”рядок з одним символом (переклад рядка).

Можна використовувати налаштування для додаткового фільтрування нашого тексту. Наприклад, можна усунути всі символи, які не є словами. У багатьох випадках пунктуація не потрібна і її легко забрати за допомогою регулярок.

Модуль **re** в Python представляє операції з регулярними виразами. Можна використовувати функцію **re.sub**, щоб замінити все, що підходить під шаблон пошуку, на вказаний рядок.

Регулярки - це потужний інструмент, за його допомогою можна створювати набагато складніші шаблони.

6. Мішок слів

Алгоритми машинного навчання не можуть працювати безпосередньо з сирим текстом, тому необхідно конвертувати текст у набори цифр (вектори). Це називається вилученням ознак.

Мішок слів – це популярна та проста техніка вилучення ознак, що використовується під час роботи з текстом. Вона визначає входження кожного слова до тексту. Щоб використати модель, потрібно:

1. Визначити словник відомих слів (токенів).
2. Вибрати рівень присутності відомих слів.

Будь-яка інформація про порядок чи структуру слів ігнорується. Ось чому це називається МІШКОМ слів. Ця модель намагається зрозуміти, чи зустрічається знайоме слово у документі, але не знає, де саме воно зустрічається.

Приклад:

Розглянемо кроки створення цієї моделі. Використовуємо лише 4 пропозиції, щоб зрозуміти, як працює модель.

Завантаження даних.

Приклад даних:

I like this movie, it's funny.

I hate this movie.

This was awesome! I like it.

Nice one. I love it.

Визначення словника.

Збираються всі унікальні слова із 4 завантажених речень, ігноруючи регістр, пунктуацію та односимвольні токени. Це буде словник (відомі слова).

Для створення словника можна використовувати клас `CountVectorizer` із бібліотеки `sklearn`. Переходимо до наступного кроку.

Створення векторів документа

Далі, треба оцінити слова у документі. На цьому кроці мета – перетворити сирий текст на набір цифр. Найпростіший метод скорингу – це відзначити наявність слів, тобто ставити 1, якщо є слово та 0 за його відсутності. Тепер створюється мішок слів, використовуючи клас `CountVectorizer`.

Висновок:

	awesome	funny	hate	it	like	love	movie	nice	one	this	was
0	0	1	0	1	1	0	1	0	0	1	0
1	0	0	1	0	0	0	1	0	0	1	0
2	1	0	0	1	1	0	0	0	0	1	1
3	0	0	0	1	0	1	0	1	1	0	0

Складність цієї моделі полягає в тому, як визначити словник і як підрахувати входження слів. Коли розмір словника збільшується, вектор документа також зростає. У прикладі вище довжина вектора дорівнює кількості відомих слів. У деяких випадках, може бути великий обсяг даних і тоді вектор може складатися з тисяч або мільйонів елементів. Більше того, кожен документ може містити лише малу частину слів зі словника. Як наслідок, у векторному поданні буде багато нулів. Вектори з великою кількістю нулів називаються розрідженими векторами (*sparse vectors*), вони вимагають більше пам'яті та обчислювальних ресурсів. Однак ми можемо зменшити кількість відомих слів, коли використовуємо цю модель, щоб зменшити вимоги до обчислювальних ресурсів. Для цього можна використовувати ту саму техніку, що розглядали до створення мішка слів:

- ігнорування регістру слів;
- ігнорування пунктуації;
- викидання стоп-слів;

- приведення слів до їх базових форм (лематизація та стемінг);
- виправлення неправильно написаних слів.

7. TF-IDF

У частотного скорингу є проблема: слова з найбільшою частотністю мають відповідно найбільшу оцінку. У цих словах може бути не так багато інформаційного виграшу для моделі, як у менш частих словах. Один із способів виправити ситуацію - знижувати оцінку слова, яке часто зустрічається у **всіх подібних документах**. Це називається TF-IDF.

TF-IDF (скорочення від term frequency - inverse document frequency) - це статистичний захід для оцінки важливості слова в документі, який є частиною колекції або корпусу.

Скоринг по TF-IDF зростає пропорційно до частоти появи слова в документі, але це компенсується кількістю документів, що містять це слово.

Формула скорингу для слова X у документі Y :

$$w_{x,y} = tf_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

TF (term frequency — частота слова) — відношення числа входжень слова до загального числа слів документа.

$$TF(\text{term}) = \frac{\text{Number of times } \textbf{term} \text{ appears in a document}}{\text{Total number of items in the document}}$$

IDF (inverse document frequency — зворотна частота документа) — інверсія частоти, з якою деяке слово зустрічається в документах колекції.

$$IDF(\text{term}) = \log \left(\frac{\text{Total number of documents}}{\text{Number of documents with } \textbf{term} \text{ in it}} \right)$$

У результаті обчислити TF-IDF для слова **term** можна:

$$TFIDF(\text{term}) = TF(\text{term}) * IDF(\text{term})$$

Приклад: Можна використовувати клас **TfidfVectorizer** з бібліотеки **sklearn**, щоб обчислити TF-IDF. Використаємо приклад:

I like this movie, it's funny.
 I hate this movie.
 This was awesome! I like it.
 Nice one. I love it.

Висновок:

	awesome	funny	hate	it	like	love	movie	nice	one	this	was
0	0.000000	0.571848	0.000000	0.365003	0.450852	0.000000	0.450852	0.000000	0.000000	0.365003	0.000000
1	0.000000	0.000000	0.702035	0.000000	0.000000	0.000000	0.553492	0.000000	0.000000	0.448100	0.000000
2	0.539445	0.000000	0.000000	0.344321	0.425305	0.000000	0.000000	0.000000	0.000000	0.344321	0.539445
3	0.000000	0.000000	0.000000	0.345783	0.000000	0.541736	0.000000	0.541736	0.541736	0.000000	0.000000

1.2 Хід роботи

Для роботи треба використати англійський варіант книги "Аліса в задзеркаллі".

Завдання.

1. Реалізувати пайплайн обробки вибраного тексту англійською мовою, включаючи всю необхідну попередню обробку тексту, включаючи приведення слів до нижнього регістру, видалення стоп-слів, цифр/неалфавітних символів, символів пунктуації.

2. Розділити текст на розділи та у кожному розділі відібрати Топ-20 слів за допомогою алгоритму TF-IDF.

3. Реалізувати алгоритм LDA і порівняти результати з отриманими раніше за допомогою TF-IDF. Зробити висновки щодо застосування реалізованих підходів.

Дати розширені відповіді:

1. Яке слово має максимальне значення метрики TF-IDF для дванадцятого розділу під назвою "Alice's Evidence"?

2. Як би ви назвали розділи книги Alice in Wonderland, ґрунтуючись на знайдених для кожного розділу важливих словах?

Результатом роботи є Jupyter Notebook із коментарями.

1.3 Міст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань (програмний код, графіки, гістограми);
- відповіді на контрольні питання.

1.4 Контрольні питання

1. Що таке токенізація за пропозиціями та як вона реалізується за допомогою NLTK?
2. Чим відрізняються лематизація та стемінг, і які їхні переваги та недоліки?
3. Що таке стоп-слова та навіщо їх видаляють при обробці текстових даних?
4. Поясніть, як працює алгоритм "мішок слів" при перетворенні тексту в числові вектори.
5. Що таке TF-IDF і як цей метод дозволяє покращити простий частотний аналіз слів?
6. Які кроки потрібно виконати для реалізації повного пайплайну обробки тексту за завданням?
7. Яким чином буде проведено аналіз та порівняння результатів застосування TF-IDF та LDA?
8. Яке слово має максимальне значення метрики TF-IDF для 12-го розділу "Alice's Evidence"?
9. Як можна назвати розділи книги "Аліса в Задзеркаллі" на основі знайдених важливих слів?
10. Які основні висновки будуть зроблені за результатами застосованих підходів до аналізу тексту?

2 ЛАБОРАТОРНА РОБОТА №7

НАЇВНИЙ БАЙЕСОВСЬКИЙ КЛАСИФІКАТОР

Мета роботи: провести EDA та побудувати наївні байесівський класифікатор для бінарної класифікації.

2.1 Теоретичні відомості

Наївний байесів класифікатор — ймовірнісний класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущенні (наївному) незалежності змінних.

Тобто, якщо на основі значень змінних можна однозначно визначити, до якого класу належить спостереження, байесів класифікатор повідомить ймовірність приналежності до цього класу.

У проміжних же випадках, коли спостереження може з різною ймовірністю належати до різних класів, результатом роботи класифікатора буде вектор, компоненти якого є ймовірностями приналежності до того чи іншого класу.

Можна бачити, що ідеальний байесів класифікатор в якомусь сенсі є оптимальним. Його результат не може бути поліпшений, тому що в усіх випадках, коли можлива однозначна відповідь, він її дасть — а в тих випадках, коли відповідь неоднозначна, результат кількісно характеризує міру цієї неоднозначності.

Разом з тим, в оптимальності криється і основний недолік ідеального байесового класифікатора: для його побудови потрібна вибірка, що містить всі можливі комбінації змінних — а розмір такої вибірки експоненціально зростає із зростанням числа змінних (див. «прокляття розмірності»). Для подолання описаної вище проблеми на практиці використовують наївний байесів класифікатор — класифікатор, побудований на основі припущення про незалежність змінних, тобто припущення про те, що використання цього припущення дозволяє не вивчати взаємодію всіх можливих поєднань змінних, обмежившись лише впливом кожної змінної окремо на приналежність образу до одного з класів.

Перевагою цього підходу є те, що вимоги до розміру вибірки скорочуються від експоненційних до лінійних. Недоліком — те, що

модель є точною лише у випадку, коли виконується припущення про незалежність. В іншому випадку, щиро кажучи, обчислені ймовірності вже не є точними (і навіть більше того, їх сума може не дорівнювати одиниці, через що потрібно нормувати результат). Однак на практиці незначні відхилення від незалежності призводять лише до незначного зниження точності, і навіть у разі істотної залежності між змінними результат роботи класифікатора продовжує корелювати з істинною приналежністю образу до класів. При цьому переваги класифікатора (висока швидкість роботи, простота і масштабованість, помірні вимоги до пам'яті) часто переважають недоліки.

2.2 Хід роботи

Завдання. Необхідно провести EDA і побудувати наївні байєсівський класифікатор для бінарної класифікації даних вище. Вивести метрики класифікатора, що вийшов. Запропонувати варіанти оптимізації моделі.

Для роботи треба використати датасет, що описує предиктори діабету:

Pregnancies - вагітність

Glucose – рівень глюкози

BloodPressure - тиск крові

SkinThickness - товщина шкірного покриву

Insulin – рівень інсуліну

BMI – індекс маси тіла

DiabetesPedigreeFunction - функція, що описує ризик діабету виходячи з історії сім'ї (генетичні фактори)

Age - вік

Outcome - клас, який необхідно передбачити (буде захворювання на діабет чи ні)

Результатом виконання роботи є Jupyter Notebook із коментарями.

2.3 Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань (програмний код, графіки, гістограми);
- відповіді на контрольні питання.

2.4 Контрольні питання

1. Що таке наївний баєсів класифікатор?
2. Яке основне припущення робить наївний баєсів класифікатор про ознаки даних?
3. Як наївний баєсів класифікатор використовує теорему Байеса?
4. Які існують типи наївних баєсових класифікаторів?
5. Як наївний баєсів класифікатор обробляє числові ознаки?
6. Які переваги наївного баєсового класифікатора?
7. Які недоліки наївного баєсового класифікатора?
8. У яких ситуаціях наївний баєсів класифікатор може показати хороші результати, незважаючи на свою "наївність"?
9. Як впливає залежність між ознаками на результати роботи наївного баєсового класифікатора?
10. В яких практичних застосуваннях може бути використаний наївний баєсів класифікатор?

3 ЛАБОРАТОРНА РОБОТА №8 КЛАСТЕРНИЙ АНАЛІЗ

Мета роботи: провести EDA і реалізувати кластеризацію даних з метою виділення груп покупців з різними перевагами.

3.1 Теоретичні відомості

Кластеризація, або кластерний аналіз — це статистична процедура, задача якої полягає в розбитті вибірки об'єктів на підмножини, що не перетинаються і називаються кластерами. Кожен кластер має складатися зі схожих об'єктів, а об'єкти різних кластерів мають істотно відрізнятися один від одного. Задача кластеризації відноситься до статистичної обробки, а також до широкого класу задач навчання без вчителя. Ще її можна описати через задачу класифікації.

Задача кластеризації — це по факту задача класифікації, бо в обох випадках ми ділимо об'єкти на основі їх подібності між собою, але у випадку кластеризації приналежність навчальних об'єктів будь-яким класам не задається. Така задача — загальна, тому для її розв'язання використовуються різні підходи. Алгоритми побудови кластерів можуть дуже відрізнятися у підходах до того, що відносити в один кластер і як їх взагалі ефективніше шукати. Кластери можна утворювати ґрунтуючись на відстані між ними, на щільності ділянок у просторі даних, інтервалах або на конкретних статистичних розподілах. Це все залежить від конкретного набору даних та мети використання результатів. Кластерний аналіз не є автоматизованим, це скоріше ітераційний процес, тому що часто доводиться змінювати метод опрацювання даних та параметри моделі, поки не буде отримано з результат з заданими властивостями.

Розв'язок неоднозначний, і на це є кілька причин. По-перше, не існує найкращого критерію якості кластеризації. Відомий цілий ряд досить ефективних критеріїв, а також ряд алгоритмів, які не мають чітко вираженого критерію, але все одно здійснюють досить якісну кластеризацію по побудові. Всі вони можуть давати різні результати.

По-друге, число кластерів, як правило, не відомо заздалегідь і встановлюється відповідно до деякого суб'єктивного критерія. По-третє, результат кластеризації істотно залежить від метрики ρ , вибір

якої, як правило, також суб'єктивний і визначається спеціалістом.

3.2 Хід роботи

Завдання. Необхідно провести EDA і реалізувати кластеризацію даних з метою виділення груп покупців зі значно різними перевагами. Необхідно реалізувати як мінімум метод кластеризації K-Means та будь-який другий, який можна вибрати самостійно. Необхідно обґрунтувати вибір числа кластерів та вивести метрики характерні для методів кластеризації, зробити текстові висновки за результатами аналізу.

Для виконання завдання необхідно використати датасет, що зберігає інформація про покупців магазину:

CustomerID – ідентифікатор покупця

Gender - підлога

Age - вік

Annual Income - річний дохід

Spending Score - рівень витрат

Результатом виконання роботи є Jupyter Notebook із коментарями.

3.3 Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань (програмний код, графіки, гістограми);
- відповіді на контрольні питання.

3.4 Контрольні питання

1. Що таке кластерний аналіз та для чого він використовується?
2. Які існують основні типи алгоритмів кластеризації?

3. Що таке критерій однорідності кластера і як він використовується при кластеризації?
4. Як визначити оптимальну кількість кластерів у даних?
5. Що таке метод k-середніх і як він працює?
6. Які є обмеження та недоліки методу k-середніх?
7. Що таке ієрархічна кластеризація і як вона відрізняється від методу k-середніх?
8. Які метрики відстані часто використовуються в кластерному аналізі?
9. Що таке метод густинної кластеризації і в чому його особливості?
10. Як оцінити якість кластеризації?

JITEPATYPA

1. Machine Learning and Knowledge Extraction [Electronic resource] / ed. by A. Holzinger [et al.]. – Cham : Springer Nature Switzerland, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-40837-3>
2. Intelligent Systems and Machine Learning [Electronic resource] / ed. by S. Nandan Mohanty, V. Garcia Diaz, G. A. E. Satish Kumar. – Cham : Springer Nature Switzerland, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-35078-8>
3. Joshi A. V. Machine Learning and Artificial Intelligence [Electronic resource] / Ameet V. Joshi. – Cham : Springer International Publishing, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-12282-8>
4. Zollanvari A. Machine Learning with Python [Electronic resource] / Amin Zollanvari. – Cham : Springer International Publishing, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-33342-2>
5. Hu F. AI, Machine Learning and Deep Learning [Electronic resource] / Fei Hu, Xiali Hei. – Boca Raton : CRC Press, 2023. – Mode of access: <https://doi.org/10.1201/9781003187158>
6. Karpatne A. Knowledge-Guided Machine Learning [Electronic resource] / Anuj Karpatne, Ramakrishnan Kannan, Vipin Kumar. – Boca Raton : Chapman and Hall/CRC, 2022. – Mode of access: <https://doi.org/10.1201/9781003143376>
7. Aggarwal C. C. Machine Learning for Text [Electronic resource] / Charu C. Aggarwal. – Cham : Springer International Publishing, 2022. – Mode of access: <https://doi.org/10.1007/978-3-030-96623-2>
8. Valkenborg D. Mathematics for Machine Learning [Electronic resource] / Dirk Valkenborg, Jeroen Wynen. – Washington, DC, USA : American Chemical Society, 2022. – Mode of access: <https://doi.org/10.1021/acsinfocus.7e5034>
9. Indovina J. Machine Learning Systems: Learn How to Develop Machine-Learning Systems / Johnson Indovina. – [S. l.] : Independently Published, 2022.
10. Applied Machine Learning / Setareh Rafatirad [et al.]. – [S. l.] : Springer International Publishing AG, 2022.