ЛЕКЦІЯ 2
МЕТОДИ КЛАСИФІКАЦІЇ У МАШИННОМУ НАВЧАННІ

ПЛАН

1. Методи класифікації
2. Оцінка якості класифікації
3. Регресія та оцінка її якості
4. Навчання без вчителя

ЛІТЕРАТУРА

Сайт http://www.mmf.lnu.edu.ua/ar/1739

КУПА інших сайтів та книжок з Інтернету.

ВСТУП

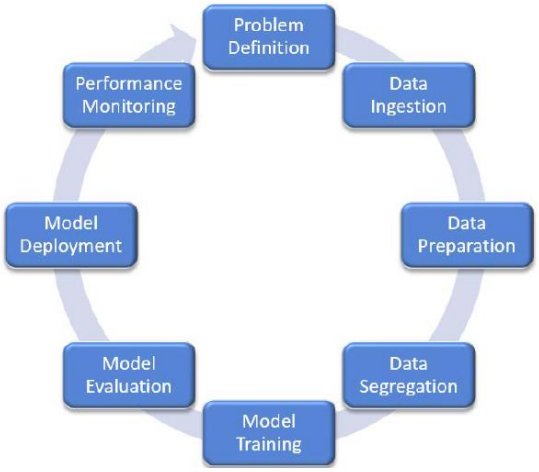
ЗГАДАЄМО, ЩО:

Машинне навчання – це спрощена (адаптована для машини) версія процесу навчання, яке відбувається з людиною.

Мета машинного навчання - передбачити результат за вхідними даними. Чим різноманітніші вхідні дані, тим простіше машині знайти закономірності і тим точніший результат.

Для того щоб навчити машину, нам потрібні три речі: **дані, ознаки, алгоритми.**

У простму вигляді, коли вирішується проста задача, створюється **конвеєр машинного навчання (machine learning pipeline)**



Класифікація методів машинного навчання

Класичне навчання люблять ділити на дві категорії — з вчителем і без. Часто можна зустріти їх англійські назви — Supervised і Unsupervised Learning.

У першому випадку у машини є якийсь учитель, який говорить їй як правильно. Розповідає, що на цій картинці кішка, а на цій собака. Тобто вчитель вже заздалегідь розділив всі дані на кішок і собак, а машина навчається на конкретних прикладах.

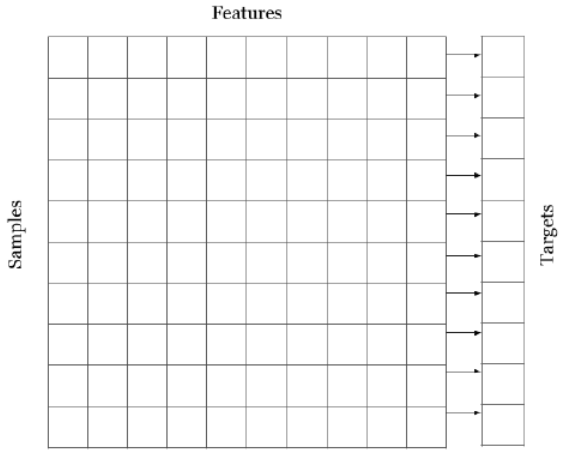


Figure 1: A supervised dataset.

У навчанні без учителя, машині просто вивалюють купу фотографій тварин на стіл і кажуть «розберися, хто тут на кого схожий». Дані не розмічені, у машини немає вчителя, і вона намагається сама знайти будь-які закономірності. Про ці методи поговоримо нижче.

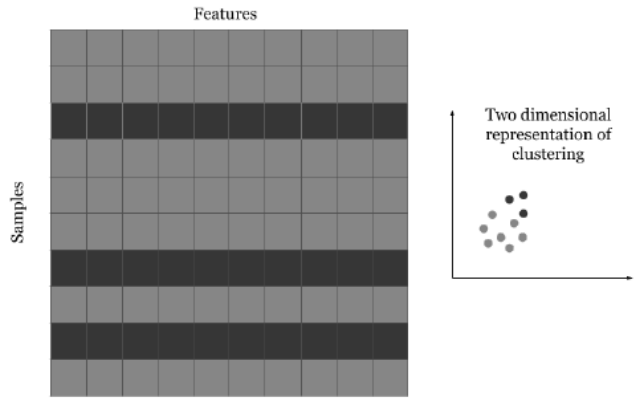
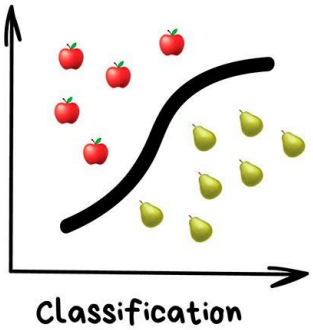


Figure 2: An unsupervised dataset.

Очевидно, що з учителем машина навчиться швидше і точніше, тому в бойових задачах його використовують набагато частіше. **Ці задачі діляться на два типи: класифікація - передбачення категорії об'єкта, і регресія - передбачення місця на числовій прямій.**

1. МЕТОДИ КЛАСИФІКАЦІЇ



«Поділяє об'єкти за заздалегідь відомою ознакою. Шкарпетки за кольорами, документи за мовами, музику за жанрами»

Сьогодні використовують для:

Спам-фільтри

Визначення мови

Пошук схожих документів

Ааналіз тональності

Розпізнавання рукописних букв і цифр

Визначення підозрілих транзакцій

Популярні алгоритми: Наївний Баєс, Дерева Рішень, Логістична Регресія, К-найближчих сусідів, Метод Опорних Векторів.

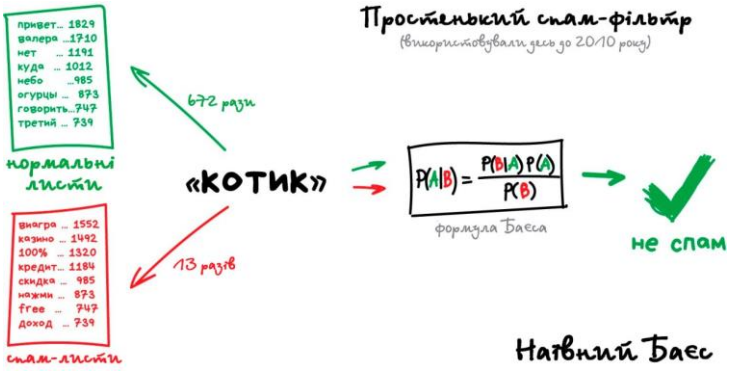
Класифікація речей - найпопулярніша задача у всьому машинному навчанні. Машина в ній як дитина, яка навчається розкладати іграшки: роботів в один ящик, танки в інший. Опа, а якщо це робот-танк? Що ж, час розплакатися і випасти в помилку.

Для класифікації завжди потрібен учитель - розмічені дані з ознаками і категоріями, які машина буде вчитися визначати за цими ознаками. Далі класифікувати можна що завгодно: користувачів за інтересами - так роблять алгоритмічні стрічки, статті за мовами і тематиками - важливо для пошукових систем, музику за жанрами - згадайте плейлисти, навіть листи у вашій поштовій скриньці.

1.1.Наївний Баєс

Наївний метод Баєса – це набір методів класифікації, заснованих на теоремі Баєса. Це не єдиний метод, а сімейство методів, які поділяють загальний принцип, згідно з яким кожна ознака, що класифікується, не залежить від значення будь-якої іншої ознаки. Наприклад, фрукт можна вважати яблуком, якщо він червоний, круглий і не перевищує 15 см у діаметрі. Наївний баєсівський класифікатор вважає, що кожна з цих «ознак» (червоний, круглий, діаметр 15 см) вносить незалежний внесок у ймовірність того, що фрукт – це яблуко, незалежно від будь-яких кореляцій між ознаками. Але насправді функції не завжди незалежні, а їхні параметри не завжди пов'язані один з одним. Саме це є недоліком методу Баєса, і тому його називають «наївним».

Раніше всі спам-фільтри працювали на алгоритмі наївного Баєса. Машина вважала скільки разів слово «кредит» зустрічається в спамі, а скільки разів на нормальних листах. Множила ці дві ймовірності за формулою Баєса, складала результати всіх слів і бац, всім лежати, у нас машинне навчання!



Наївний Баєс

Пізніше спамери навчилися обходити фільтр Байеса, просто вставляючи в кінець листа багато слів з «хорошими» рейтингами. Метод отримав іронічну назву Отруєння Баєса, а фільтрувати спам стали іншими алгоритмами. Але метод назавжди залишився в підручниках як найпростіший, красивий і один з перших практично корисних.

1.2. Дерево рішень

Візьмемо інший приклад корисної класифікації. Ось берете ви кредит в банку. Як банку упевнитися повернете ви його чи ні? Дуже точно ніяк, але банк має тисячі профілів інших людей, які вже брали кредит до вас. Там вказано їхній вік, освіту, посаду, рівень зарплати та головне - хто з них повернув кредит, а з ким виникли проблеми.

Отож, всі здогадалися, де тут дані і який треба передбачити результат. Навчимо машину, знайдемо закономірності, отримаємо відповідь - питання не в цьому. Проблема в тому, що банк не може сліпо довіряти відповіді машини, без пояснень. Раптом збій, злі хакери або бухий адмін вирішив скриптик виправити.

Для цієї задачі придумали Дерево Рішень. Машина автоматично розділяє всі дані за запитаннями, відповідь на які «так» або «ні». Питання можуть бути не зовсім адекватними з точки зору людини, наприклад «зарплата позичальника більше, ніж 25934 гр.?», але машина придумує їх так, щоб на кожному кроці розбиття було найточнішим.

Дерево Рішень

Так отримується дерево питань. Чим вищий рівень, тим загальніше запитання. Потім навіть можна загнати їх аналітикам і вони понавидумують чому саме так.

Дерева знайшли свою нішу в областях з високою відповідальністю: діагностиці, медицині, фінансах.

Два найпопулярніших алгоритми побудови дерев - CART і C4.5 .

У чистому вигляді дерева сьогодні використовують рідко, але ось їх ансамблі (про які буде пізніше) лежать в основі великих систем і часто обскакують навіть нейромережі.

Наприклад, коли ви задаєте запитання Ґуґлу, то саме натовп дурних дерев біжить ранжувати вам результати.

Переваги методу. Їх легко зрозуміти. У кожному вузлі ми можемо точно побачити, яке рішення приймає наша модель. На практиці ми зможемо точно дізнатися, звідки виходять точності і помилки, з якими видами даних модель буде справлятися і як значення ознак впливають на вихід. Опція візуалізація в Scikit-learn є зручним інструментом, що сприяє хорошому розуміння дерев рішень.

Не вимагає об'ємної підготовки даних. Багато моделей машинного навчання вимагають попередньої обробки даних (наприклад, нормалізації) і потребують складних схемах регуляризації. З іншого боку, дерева рішень ефективні після настройки деяких параметрів.

Вартість використання дерева для виведення є логарифмічною від числа точок даних, що використовуються для навчання дерева. Це є великою перевагою, так як велика кількість даних не сильно вплине на швидкість виведення.

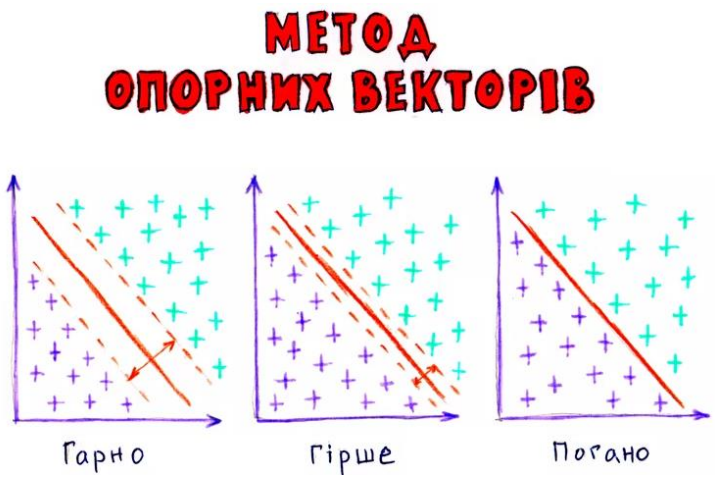
Недоліки методу. Через свій характер навчання дерева рішень схильні до перенавчання. Рекомендується часто застосовувати деякі види зниження розмірності, наприклад, PCA, щоб дерево не створювало розбиття по великій кількості ознак.

З тих же причин, що і перенавчання, дерева рішень також уразливі до зміщення класів, які є в більшості наборів даних. Хорошим рішенням в даному випадку є періодична балансування класів (ваги класу, вибірка, певна функція втрат).

1.3. Метод Опорних Векторів

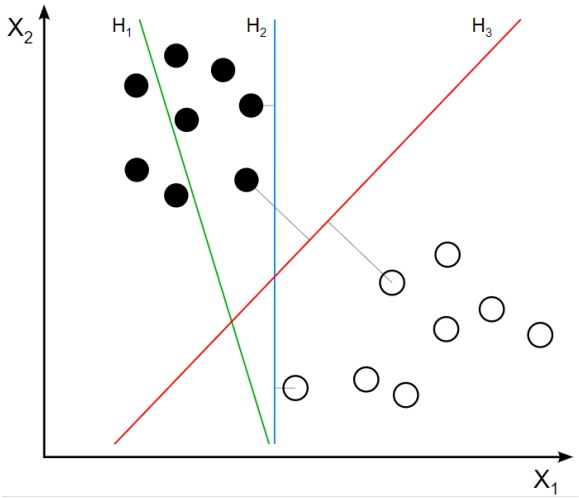
Але найпопулярнішим методом класичної класифікації заслужено є Метод Опорних Векторів (SVM). Ним вже класифікували все, що тільки можна класифікувати: види рослин, обличчя на фотографіях, документи за тематиками... Багато років він був головною відповіддю на питання «який би мені взяти класифікатор».

Ідея SVM проста - він шукає, як так провести дві прямі між категоріями, щоб між ними утворився найбільший зазор. На зображенні видно наочніше:



Метод опорних векторів – найпопулярніший для класичної класифікації, тому що він простий та швидкий. Виходячи з того, що об'єкт, який перебуває в N-вимірному просторі, належить до одного з двох класів, метод опорних векторів будує гіперплощину з розмірністю (N – 1), щоб всі об'єкти виявилися в одній з двох груп. Що ж таке гіперплощина? Як простий приклад для завдання класифікації, що має тільки два класи,

ви можете уявити собі гіперплощину як лінію, яка розділяє і класифікує набір даних. До того ж, що далі від гіперплощини лежать наші точки даних, то більше ми впевнені в тому, що вони були правильно класифіковані. Що менше схожих ознак між даними, то менша ймовірність належності до одного класу. Тому ідеально, якщо точки даних перебувають якомога далі від гіперплощини, і до того ж залишаються на правильній стороні.



H1 не розділяє ці класи. H2 розділяє, але лише з невеликим розділенням.
H3 розділяє їх із максимальним розділенням.

Головним недоліком методу є те, що він підходить тільки до розв'язання завдань з двома класами.

SVM є корисними для категоризації текстів та гіпертекстів, оскільки їхнє застосування може значно знижувати потребу в мічених тренувальних зразках як у стандартній індуктивній, так і в трансдуктивній[en] постановках.

Із застосуванням SVM може виконуватися й класифікація зображень. Експериментальні результати показують, що SVM можуть досягати значно вищої точності пошуку, ніж традиційні схеми уточнення запиту, всього лише після трьох-чотирьох раундів зворотного зв'язку про відповідність.

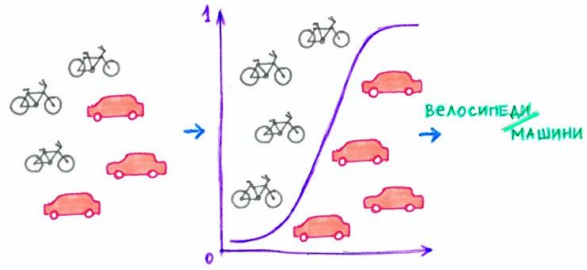
За допомогою ОВМ може здійснюватися розпізнавання рукописних символів.

1.4. Логістична регресія

Логістичну регресію добре використовувати для завдань бінарної класифікації, тобто коли на виході потрібно отримати відповідь, до якого з двох класів належить об'єкт. Логістична регресія схожа на лінійну тим, що в ній теж потрібно знайти значення коефіцієнтів для вхідних змінних, але різниця в тому, що вихідне значення перетвориться за допомогою нелінійної або логістичної функції. Логістична функція перетворює будь-яке значення в число в межах від 0 до 1. Так і передбачається ймовірність належності до одного чи другого класу. Тобто, на відміну від логістичної регресії, тут не проводять передбачення значення числової змінної, виходячи з вибірки вихідних значень, а замість цього значенням функції є ймовірність того, що це початкове значення належить до певного класу.

До недоліків логістичної регресії можна віднести залежність від набору даних та низьку стійкість до помилок.

ЛОГІСТИЧНА РЕГРЕСІЯ



З погляду математики, **логістична регресія** - це статистичний інструмент, спрямований на моделювання біноміального результату з однією або кількома пояснювальними змінними.

За допомогою статистичних методів логістична регресія дозволяє генерувати результат, який насправді представляє ймовірність того, що задане вхідне значення належить даному класу.

У задачах двочленної логістичної регресії ймовірність того, що вихід належить одному класу, буде P, тоді як він належить іншому класу 1-P (де P - число між 0 і 1, оскільки воно виражає ймовірність).

Прикладами проблем, які можна вирішити логістичною регресією, є:

електронний лист є спамом чи ні;

покупка в Інтернеті є шахрайською чи ні, оцінюючи умови покупки;

у пацієнта перелом, оцінюючи його радіуси.

За допомогою логістичної регресії ми можемо робити прогнозний аналіз, вимірюючи взаємозв'язок між тим, що ми хочемо передбачити (залежною змінною), і однією або декількома незалежними змінними, тобто характеристиками. Оцінка ймовірності здійснюється за допомогою логістичної функції.

В подальшому ймовірності перетворюються на двійкові значення, і для того, щоб зробити прогноз реальним, цей результат присвоюється класу, якому він належить, виходячи з того, близький він чи ні до самого класу.

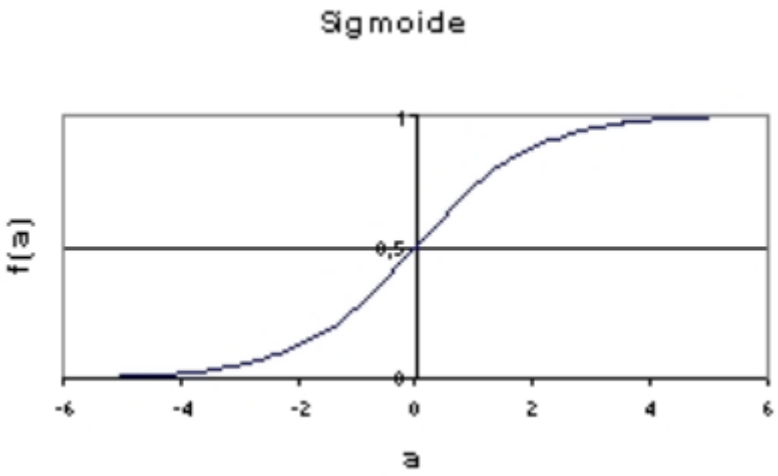
Наприклад, якщо застосування логістичної функції повертає 0,85, то це означає, що вхідні дані генерують позитивний клас, присвоюючи його класу 1. І навпаки, якщо він отримав таке значення, як 0,4 або більш загально <0,5 .

Логістична функція, яка також називається сигмоїдною, - це крива, здатна приймати будь-яке число реальної величини і відображати її до значення між 0 і 1, виключаючи крайності. Функція:

$$f(x) = \frac{1}{1+e^{-(b0 + b1*x)}}$$

де: e: основа природних логарифмів (число Ейлера або функція excel exp ())

b0 + b1 * x: фактичне числове значення, яке ви хочете перетворити.



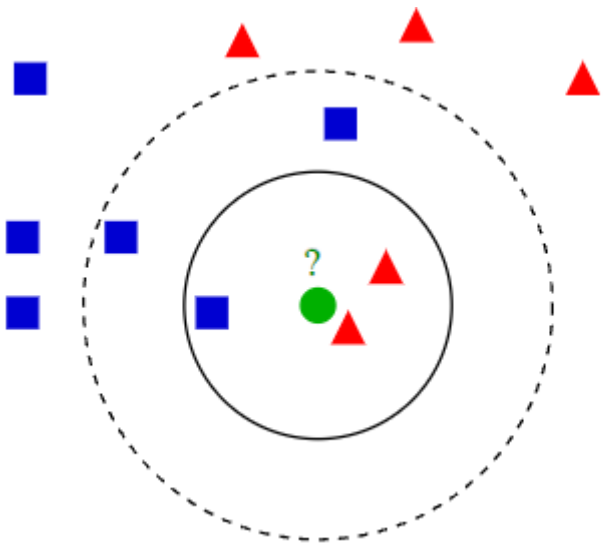
Представлення, що використовується для логістичної регресії
Логістична регресія використовує рівняння як подання, подібно до лінійної регресії

1.5. Метод k-найближчих сусідів

Ме́тод k-найбли́жчих суси́дів (англ. k-nearest neighbor method) — простий непараметричний класифікаційний метод, де для класифікації об'єктів у рамках простору властивостей використовуються відстані (зазвичай евклідові), пораховані до усіх інших об'єктів. Вибираються об'єкти, до яких відстань найменша, і вони виділяються в окремий клас.

Метод k-найближчих сусідів — метричний алгоритм для автоматичної класифікації об'єктів. Основним принципом методу найближчих сусідів є те, що об'єкт приписується класу, найпоширенішому серед його сусідів. Сусіди беруться, виходячи з множини об'єктів, класи яких уже відомі, і, виходячи з ключового для даного методу значення k, вираховується найчисленніший серед них клас. Кожен об'єкт має кінцеву кількість атрибутів (розмірностей). Передбачається, що існує певний набір об'єктів з уже наявною класифікацією.

Прокляття розмірності. Цей класифікатор робить припущення, що подібні точки мають подібні мітки. На жаль, у високорозмірнісних просторах, точки, вибрані з розподілу ймовірностей, майже ніколи не опиняються близько одна до одної.



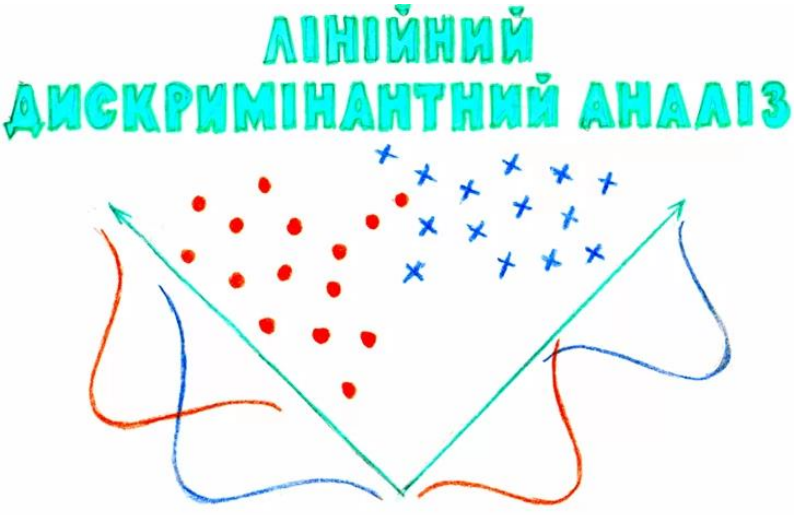
Приклад класифікації k найближчих сусідів. Тестовий зразок (зелене коло) повинен бути класифікований як синій квадрат (клас 1) або як червоний трикутник (клас 2). Якщо k = 3, то класифікується як 2-й клас, тому що всередині меншого кола 2 трикутника і тільки 1 квадрат. Якщо k = 5, то він буде класифікований як перший клас (3 квадрата проти 2-ох трикутників всередині більшого кола).

1.6. Лінійний дискримінантний аналіз

Якщо класів більше, ніж два, то краще використовувати лінійний дискримінантний аналіз. Наприклад, лікарю потрібно визначити діагноз пацієнта. У нього є множина значень аналізів для кожного можливого захворювання. Лікар бере ці аналізи у пацієнта і порівнює їхні значення зі значеннями для захворювань. Так визначають ймовірності кожного з переліку захворювань у пацієнта.

Метод містить статистичні властивості даних, розраховані для кожного класу. Під час цього передбачається, що дані мають нормальний розподіл, тому перед початком роботи необхідно видалити з даних аномальні значення. Варто зазначити, що, на відміну від цього методу, в інших нормальний розподіл не обов'язковий. Прогнозні класи знаходять шляхом обчислення дискримінантного значення для кожного класу, тобто знаходження лінійної залежності комбінації значень для вибору класу, а потім обирають клас із найбільшим значенням.

Головна перевага методу – це простота реалізації та інтерпретації результатів, недолік – чутливість до розподілу вхідних даних, коли навіть невелике їхнє змінення призводить до значних змін результатів класифікації.



У класифікації є корисна зворотна сторона - пошук аномалій. Коли якась ознака об'єкта аж занадто не вписується в наші класи, ми яскраво підсвічуємо його на екрані. Зараз так роблять в медицині: комп'ютер підсвічує лікарю всі підозрілі області МРТ або виділяє відхилення в аналізах. На біржах таким же чином визначають нестандартних гравців, які швидше за все є інсайдерами. У мережах так виявляють атаки чи неадекватну роботу Навчивши комп'ютер «як правильно», ми автоматично отримуємо і зворотний класифікатор - як неправильно.

Сьогодні для класифікації все частіше використовують нейромережі, адже по-суті їх для цього і винайшли.

Правило таке: складніші дані - складніший алгоритм. Для тексту, цифр, табличок я б починав з класики. Там моделі менші, навчаються швидше і працюють

зрозуміліше. Для картинок, відео та іншої незрозумілої бігдати - одразу б дивився в сторону нейромереж.

Років п'ять тому ще можна було зустріти класифікатор осіб на SVM, але сьогодні під цю задачу сотня готових сіток по інтернету валяються, чом би їх не взяти. А ось спам-фільтри як на SVM писали, так і не бачу сенсу зупинятися.

2. ОЦІНКА ЯКОСТІ КЛАСИФІКАЦІЇ

Провели ми класифікацію і що далі? Як оцінити наскільки якісно ми провели цю класифікацію? Чи правильно вона виконана? Чи достатньо нам цього алгоритму, чи може необхідно застсувати якийсь інший? Відповіді на ці питання дають показники (метрики) якості класифікації.

Матриця помилок (або матриця неточностей, англ. Confusion matrix)

Перед переходом до самих метрик необхідно ввести важливу концепцію для опису цих метрик в термінах помилок класифікації - confusion matrix (матриця помилок). Припустимо, що у нас є два класи $y = \{0,1\}$ і алгоритм, який пророкує (передбачає) приналежність кожного об'єкта одному з цих класів. *Розглянемо приклад*. Нехай система захисту мережі використовує систему класифікації для виявлення атаки: нормальна робота мережі чи аномальна (наявність атаки). При цьому у першому випадку система і далі нормально працює, а у другому – видається сигнал аномальної роботи. Таким чином, виявлення неадекватної (аномальної) роботи мережі ($y = 1$) можна розглядати як "сигнал тривоги", що повідомляє про можливі ризики.

Будьякий реальний класифікатор робить помилки. У нашому випадку таких помилок може бути дві:

Нормальна ситуація у мережі за даними трафіка розпізнається моделлю як аномальна. Даний випадок можна трактувати як "помилкову тривогу".

Аномальна ситуація розпізнається як нормальна і ніяких дій по захисту від атаки не відбувається. Даний випадок можна розглядати як "пропуск цілі".

Неважко помітити, що ці помилки нерівноцінні по зв'язаних з ними наслідками. У разі "помилкової тривоги" втрати складуть тільки марно потрачений час та ресурси на протидію неіснуючій загрозі. У разі "пропуску цілі" можна втратити набагато більше (інформацію, роботу мережі та інше, це залежить від виду атаки). Тому системі захисту важливіше не допустити "пропуск цілі", ніж "помилкову тривогу".

Оскільки з точки зору логіки завдання виявлення аномалій нам важливіше правильно розпізнати аномалію (атаку) з міткою y = 1, ніж помилитися в розпізнаванні нормальної роботи мережі, будемо називати відповідний результат класифікації позитивним (аномалія чи атака виявлені вірно), а протилежний - негативним (аномалії чи атаки немає y = 0). Тоді можливі наступні чотири результати класифікації:

True Positive (TP) – наявність атаки класифікована як наявна атака, тобто позитивний клас розпізнано як позитивний. Спостереження, для яких це має місце називаються істинно-позитивними.

True Negative (TN) – нормальна робота мережі класифікована як нормальна робота без аномалій, тобто негативний клас розпізнано як негативний. Спостереження, яких це має місце, називаються істинно негативними.

False Positive (FP) – нормальна робота мережі класифікована як аномальна, тобто мала місце помилка, в результаті якої негативний клас був розпізнаний як позитивний.

Спостереження, для яких було отримано такий результат класифікації, називаються помилково-позитивними, а помилка класифікації називається помилкою I роду.

False Negative (FN) – атака чи аномальна робота мережі розпізнана як нормальна, тобто мала місце помилка, в результаті якої позитивний клас був розпізнаний як негативний. Спостереження, для яких було отримано такий результат класифікації, називаються помилково-негативними, а помилка класифікації називається помилкою II роду.

Таким чином, помилка I роду, або хибно-позитивний результат класифікації, має місце, коли негативне спостереження розпізнано моделлю як позитивне. Помилкою II роду, або хибно-негативних результатом класифікації, називають випадок, коли позитивне спостереження розпізнано як негативне. Пояснимо це за допомогою матриці помилок класифікації:

	y = 1	y = 0
a (x) = 1	Істинно-позитивний (True Positive - TP)	Помилково-позитивний (False Positive - FP)
a (x) = 0	Помилково-негативний (False Negative - FN)	Істинно-негативний (True Negative – TN)

Тут a(x) - це відповідь алгоритму при конкретній ситуації, а y - справжня мітка класу для цієї ситуації. Таким чином, помилки класифікації бувають двох видів: False Negative (FN) і False Positive (FP). P означає що класифікатор визначає клас об'єкта як позитивний (N - негативний). T означає що клас передбачений правильно (відповідно F - неправильно). Кожен рядок в матриці помилок представляє прогнозований клас, а кожен стовпець - фактичний клас.

Тобто у загальному випадку, матриця неточностей - це матриця розміром N на N, де N – кількість класів, яка представляє собою табличне представлення прогнозованих і фактичних значень для кожного можливого класу.

Матриця помилок, одна з наважливіших речей, на яку потрібно дивитися при оцінці моделі класифікації. Це матриця, яка візуалізує кількість фактичних екземплярів класу в порівнянні з прогнозованими екземплярами класу. Таке подання дозволяє нам швидко побачити кількість правильних і неправильних прогнозів для кожної категорії.

На основі цієї матриці будується ряд інших характеристик. Розглянемо кожну з них більш детально.

Акуратність (англ. Accuracy).

Акуратністю називається пропорція точних прогнозів по відношенню до загальної кількості прогнозів, тобто це ймовірність того, що клас буде передбачений правильно (1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Тобто, accuracy - частка правильних відповідей алгоритму:

Хоча акуратність є швидким та інформативним індикатором продуктивності моделі, ми не можемо покладатися виключно на неї. Це пов'язано з тим, що вона приховує наявність зсуву в моделі, що є звичайним явищем, якщо набір даних незбалансований, тобто негативних моментів значно більше, ніж позитивних, або навпаки. Тобто, ця метрика марна в задачах з нерівними класами, що як варіант можна виправити за допомогою алгоритмів семпліювання. Семпліювання (англ. Data sampling) - метод

коригування навчальної вибірки з метою балансування розподілу класів у вихідному наборі даних. Розглянемо це *на простому прикладі*.

Припустимо, ми хочемо оцінити роботу спам-фільтра пошти. У нас є 100 НЕ-спам листів, 90 з яких наш класифікатор визначив вірно (True Negative = 90, False Positive = 10), і 10 спам-листів, 5 з яких класифікатор також визначив вірно (True Positive = 5, False Negative = 5). Тоді accuracy:

$$Accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86{,}4$$

Але якщо ми просто будемо передбачати, що всі листи Не-спам, то отримаємо більш високу акуратність:

$$Accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90{,}9$$

При цьому, наша модель абсолютно не володіє ніякою прогностичною силою, оскільки спочатку ми хотіли визначати листи зі спамом. Подолати це нам допоможе перехід із загальної для всіх класів метрики до окремимих показників якості класів.

Точність (англ. Precision).

Точністю називається частка правильних відповідей моделі в межах класу - це частка об'єктів, що дійсно належать даному класу, щодо всіх об'єктів які система віднесла до цього класу.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Саме введення precision не дозволяє нам записувати всі об'єкти в один клас, так як в цьому випадку ми отримуємо зростання рівня False Positive.

Повнота (англ. Recall).

Повнота - це частка істинно позитивних класифікацій. Повнота показує, яку частку об'єктів, що реально належать до позитивного класу, ми передбачили вірно. Або ж іншими словами: це частка варіантів, класифікованих як позитивні, які насправді виявилися позитивними.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Повнота (recall) демонструє здатність алгоритму виявляти даний клас взагалі.

Маючи матрицю помилок, дуже просто можна обчислити точність і повноту для кожного класу. Точність (precision) дорівнює відношенню відповідного діагонального елемента матриці і суми всього рядка класу. Повнота (recall) - відношенню діагонального елемента матриці і суми всього стовпчика класу. Оскільки класів може бути багато (не обов’язково два), то формально:

$$Precision_{\text{c}} = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{c,i}}$$

$$Recall_{\mathrm{c}} = \frac{A_{c,c}}{\sum_{i=1}^{n} A_{i,c}}$$

Тобто, результуюча точність класифікатора розраховується як середнє арифметичне його точності по всіх класах. Те ж саме з повнотою.

F-міра (англ. F-score).

Precision і recall не залежить від співвідношення класів (на відміну від accuracy) і тому можуть бути застосовні в умовах незбалансованих вибірок. Часто в реальній практиці стоїть завдання знайти оптимальний (для замовника) баланс між цими двома метриками. Зрозуміло що чим вище точність і повнота, тим краще. Але в реальному житті максимальна точність і повнота недосяжні одночасно і доводиться шукати якийсь баланс. Тому, хотілося б мати якусь метрику яка об'єднувала б у собі інформацію про точність та повноту нашого алгоритму. У цьому випадку нам буде простіше приймати рішення про те, яку реалізацію запускати у виробництво (у кого більше той і крутіше). Саме такою метрикою є F-міра.

F-міра є гармонійним середнім між точністю і повнотою. Вона прагне до нуля, якщо точність або повнота прагне до нуля.

$$F = \frac{2 \times precision \times recall}{precision + recall} \quad (4)$$

Дана формула надає однакову вагу точності і повноти, тому F-міра буде падати однаково при зменшенні і точності і повноти. Можливо розрахувати F-міру надавши різну вагу точності і повноті, якщо ви свідомо віддаєте пріоритет одній з цих метрик при розробці алгоритму:

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{(\beta^2 \times precision) + recall} \quad (5)$$

де β приймає значення в діапазоні 0 <β <1 якщо ви хочете віддати пріоритет точності, а при β> 1 пріоритет віддається повноті. При β = 1 формула зводиться до попередньої і ви отримуєте збалансовану F-міру (також її називають $\boldsymbol{F_1}$).

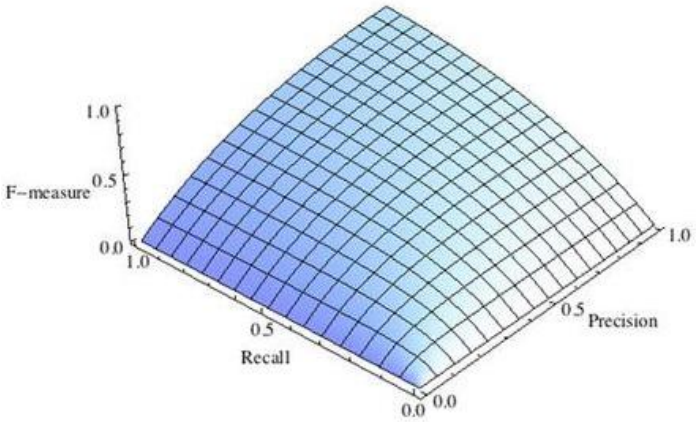


Рис.1 Збалансована F-міра, β=1

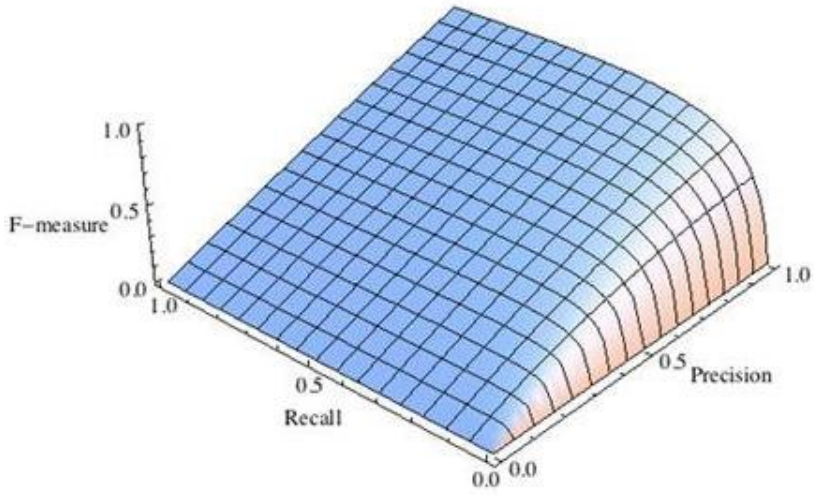


Рис.2 F-міра з приоритетом точності, $\beta^2=1/4$

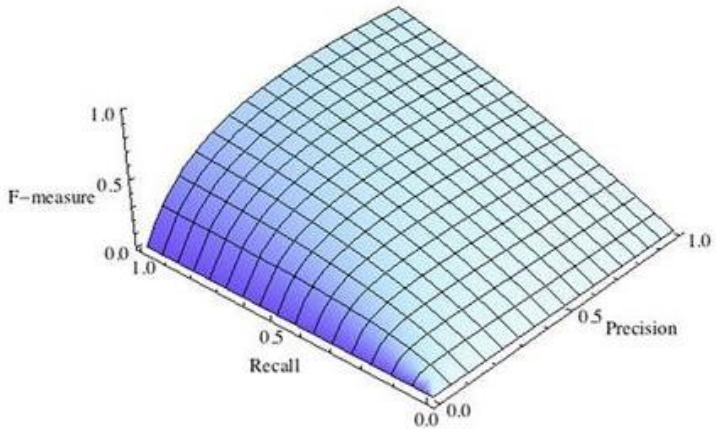


Рис.3 F-міра с пріоритетом повноти, $\beta^2=2$

F-міра досягає максимуму при максимальній повноті і точності, і близька до нуля, якщо один з аргументів близький до нуля.

F-міра є хорошим кандидатом на формальну метрику оцінки якості класифікатора. Вона зводить до одного числа дві інші основоположні метрики: точність і повноту. Маючи "F-міру" набагато простіше відповісти на питання: "змінився алгоритм в кращу сторону чи ні?"

ROC-крива

Крива робочих характеристик (англ. Receiver Operating Characteristics curve). Використовується для аналізу поведінки класифікаторів при різних порогових значеннях. Дозволяє розглянути всі порогові значення для даного класифікатора. Показує частку хибно позитивних прикладів (англ. False positive rate, FPR) в порівнянні з часткою істинно позитивних прикладів (англ. True positive rate, TPR).

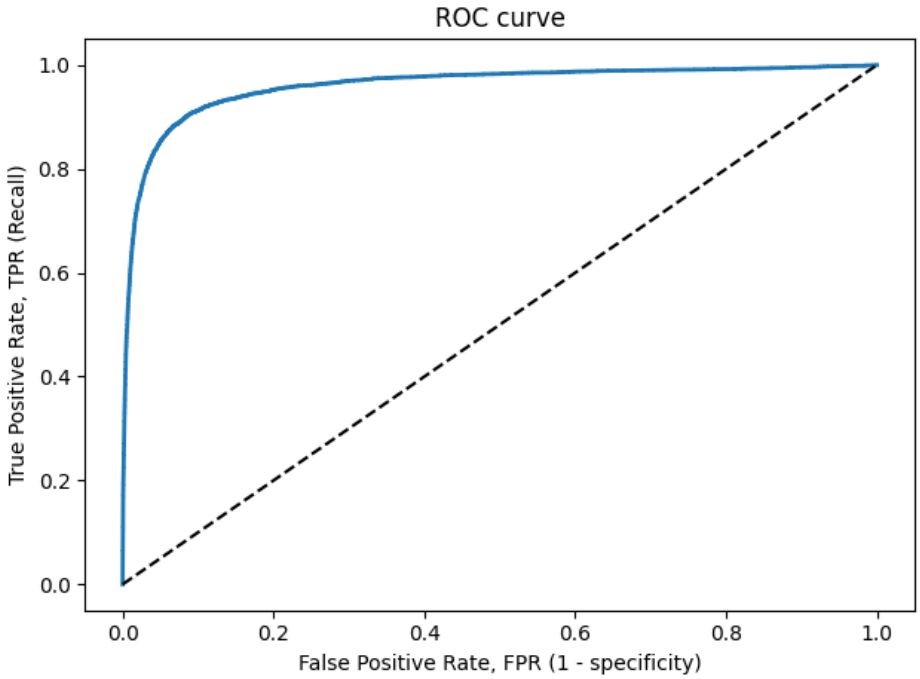


Рисунок 6 –ROC-крива

$$TPR = \frac{TP}{TP + FN} = Recall \quad (6)$$

$$FPR = \frac{FP}{FP + TN} \quad (7)$$

Частка *FPR* - це пропорція негативних зразків, які були некоректно класифіковані як позитивні.

$$FPR = 1 - TNR$$

де TNR - частка істинно негативних класифікацій (англ. Trie Negative Rate), що представляє собою пропорцію негативних зразків, які були коректно класифіковані як негативні.

Частка TNR також називається специфічністю (англ. Specificity). Отже, ROC-крива зображає чутливість (англ. Sensitivity), тобто повноту, в порівнянні з різницею 1 - specificity.

Пряма лінія по діагоналі представляє ROC-криву чисто випадкового класифікатора. Хороший класифікатор тримається від зазначеної лінії настільки далеко, наскільки це можливо (прагнучи до лівого верхнього кута).

Один із способів порівняння класифікаторів передбачає вимір площі під кривою (англ. Area Under the Curve - AUC). Бездоганний класифікатор матиме площу під ROC-кривою (ROC-AUC), що дорівнює 1, тоді як чисто випадковий класифікатор - площу 0.5.

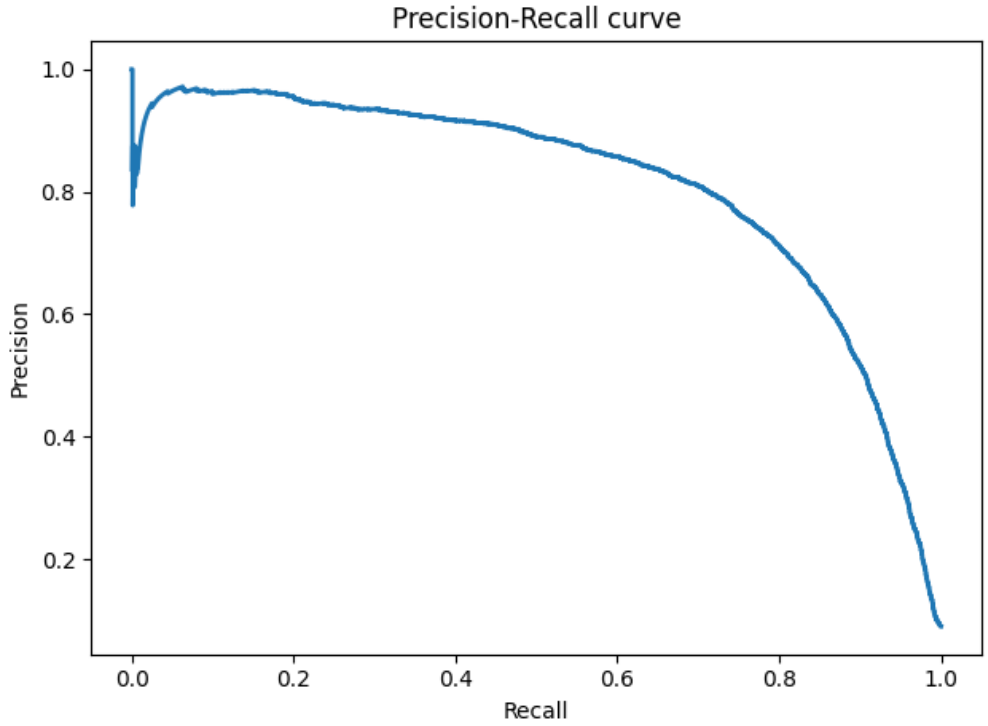
Графік ROC допомагає прийняти рішення про те, де встановити поріг класифікації, щоб максимізувати істинно позитивний рівень або мінімізувати псевдопозитивний показник, що в кінцевому підсумку є бізнес-рішенням.

Precison-recall крива

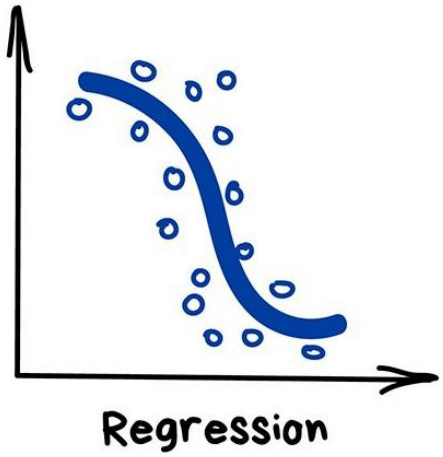
Чутливість до співвідношення класів. Розглянемо задачу виділення математичних статей з безлічі наукових статей. Припустимо, що за все мається 1.000.100 статей, з яких

лише 100 належать до математики. Якщо нам вдасться побудувати алгоритм *a(x)*, що ідеально вирішує завдання, то його TPR буде дорівнює одиниці, а FPR - нулю. Розглянемо тепер поганий алгоритм, що дає позитивну відповідь на 95 математичних і 50.000 нематематичних статтях. Такий алгоритм абсолютно даремний, але при цьому має TPR = 0.95 і FPR = 0.05, що вкрай близько до показників ідеального алгоритму. Таким чином, якщо позитивний клас істотно менше за розміром, то AUC-ROC може давати неадекватну оцінку якості роботи алгоритму, оскільки вимірює частку невірно прийнятих об'єктів щодо загального числа негативних. Так, алгоритм *b(x)*, що поміщає 100 релевантних документів на позиції з 50.001-й по 50.101-ю, матиме AUC-ROC 0.95.

Позбутися від зазначеної проблеми з незбалансованими класами можна, перейшовши від ROC-кривої до Precison-recall (PR) PR-кривої. Вона визначається аналогічно до ROC-кривої, тільки по осях відкладаються НЕ FPR і TPR, а повнота (по осі абсцис) і точність (по осі ординат). Критерієм якості сімейства алгоритмів виступає площа під PR-кривою (англ. Area Under the Curve - AUC-PR).



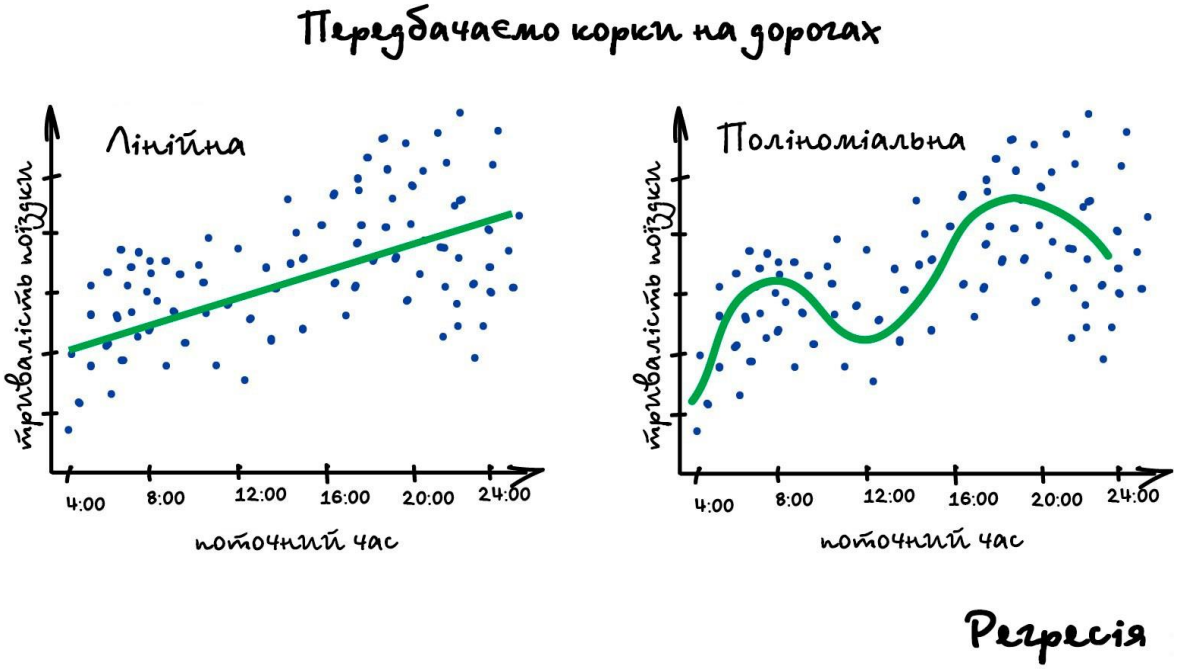
3. РЕГРЕСІЯ ТА ОЦІНКА ЇЇ ЯКОСТІ



«Намалюй лінію уздовж моїх точок. Саме так, це машинне навчання»
Сьогодні використовують для:
Прогноз вартості цінних паперів
Аналіз попиту, обсягу продажів
Медичні діагнози
Будь-які залежності числа від часу
Популярні алгоритми: Лінійна або Поліноміальна Регресія

Регресія - та ж класифікація, тільки замість категорії ми передбачаємо число. Вартість автомобіля з його пробігом, кількість корків щодо часу доби, обсяг попиту на товар від зростання компанії і.т.д. На регресію ідеально лягають будь-які задачі, де є залежність від часу.

Регресію дуже люблять фінансисти і аналітики, вона вбудована навіть в Excel. Всередині все працює, знову ж таки, банально: машина тупо намагається намалювати лінію, яка в середньому відображає залежність. Правда, на відміну від людини з фломастером і вайтбордом, робить вона це з точністю - обчислюючи середню відстань до кожної точки і намагаючись всім догодити.



Коли регресія малює пряму лінію, її називають лінійною :))), коли криву - поліноміальною :)))))). Це два основні типи регресії, далі вже починаються рідкоземельні методи. Але так як в сім'ї не без виродка, є *Логістична Регресія*, яка насправді не регресія, а метод класифікації, від чого у всіх постійно плутанина. Не робіть так.

Схожість регресії і класифікації підтверджується ще й тим, що багато хто з класифікаторів, після невеликого тюнінгу, перетворюються в регресорів. Наприклад, ми можемо не просто дивитися до якого класу належить об'єкт, а запам'ятовувати, наскільки він близький - і ось, у нас регресія.

ПРИКЛАД

Припустимо, ми хочемо купити діамант. У нас є кільце, яке належало моєї бабусі. Його оправа містить діамант масою 1,35 карата, і я хочу знати, скільки він буде коштувати. Я беру з собою блокнот і ручку і йду в ювелірний магазин. Там я записую всі ціни на діаманти, які є у вітрині, а також їх масу в каратах. Починаючи з першого діаманта, який має масу 1,01 карата і стоїть 7366 дол. США.

Я йду далі і роблю те ж саме для інших діамантів в магазині.

Carats	price
1.01	7,366
.49	985
.31	544
1.51	9,140
.37	493
.73	3,011
1.53	11,413
.56	1,814
.41	876
74	

Стовпці з даними діамантів

Зверніть увагу, що наш список містить два стовпці. Кожен стовпець має свій атрибут - масу в каратах і ціну - і кожен рядок є окремим точкою даних, що представляє один діамант.

Фактично ми створили невеликий набір даних у формі таблиці. Зверніть увагу, що він відповідає нашим критеріям якості.

Дані є релевантними: маса безумовно пов'язана з ціною.

Вони точні: ми перевірили ціни, які записали.

Вони пов'язані: всі стовпці і рядки заповнені.

І, як ми побачимо далі, цих даних достатньо, щоб дати відповідь на наше питання.

Постановка точного питання

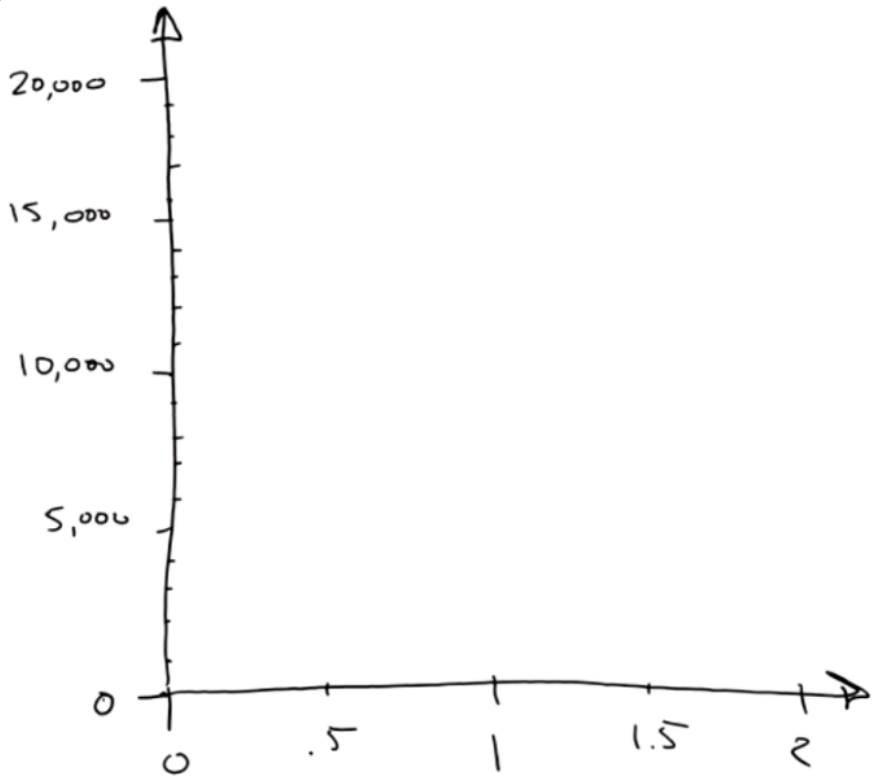
Тепер давайте точно сформулюємо наше запитання: "Скільки буде коштувати діамант масою 1,35 карата?"

У нашому списку немає діаманта масою 1,35 карата, тому необхідно використовувати наявні дані для отримання відповіді на це питання.

Побудова існуючих даних

Перше, що необхідно зробити, це намалювати горизонтальну числову лінію, яку називають віссю. На ній буде відображатися маса. Діапазон мас - від 0 до 2, тому ми намалюємо лінію, що охоплює цей діапазон, і помістимо на неї позначки для кожних 0,5 карата.

Потім ми намалюємо вертикальну вісь, на якій запишемо ціни, і з'єднаємо її з горизонтальною віссю маси. Одиницею виміру для цієї осі буде долар США. Тепер у нас є набір осей координат.

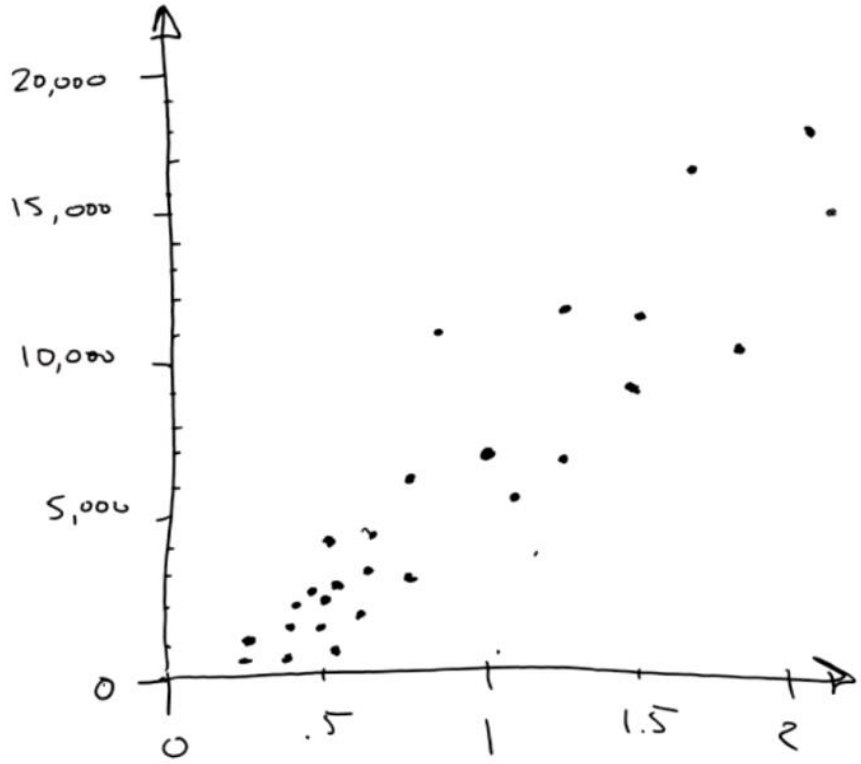


Осі маси і ціни

Тепер ми перетворимо ці дані в точкову діаграму. Це відмінний спосіб візуалізації числових наборів даних.

Беремо першу точку даних і проводимо вертикальну лінію від позначки 1,01 карата. Потім проводимо горизонтальну лінію від позначки 7366 дол. США. Там, де ці лінії перетинаються, малюємо точку. Ця точка і представляє наш перший діамант.

Тепер ми пройдемо по всім діамантів з нашого списку і зробимо те ж саме. В результаті ми отримаємо таку картину: безліч точок, кожна з яких відповідає одному діаманту.

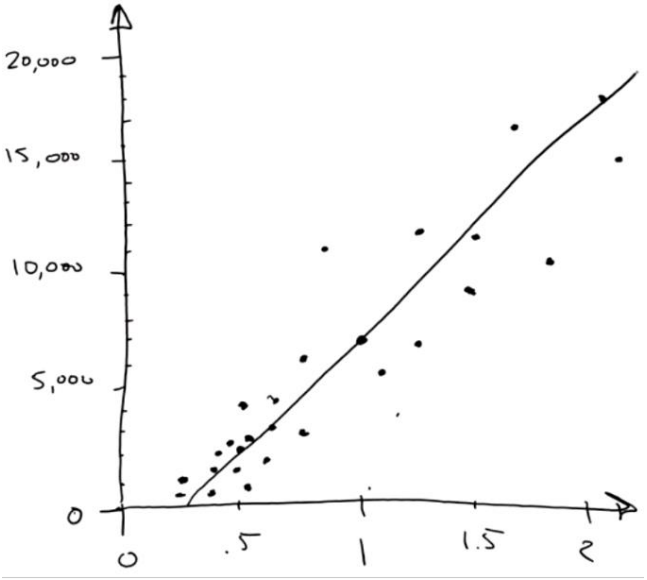


точкова діаграма

Побудова моделі на основі точок даних

Тепер, якщо поглянути на точки під невеликим кутом, то весь цей набір виглядає як товста і нечітка лінія. Можна скористатися маркером і провести через набір точок пряму лінію.

Намалювавши лінію, ми створили модель. Щоб краще зрозуміти, уявіть собі, що реальний світ зображений у спрощеній формі, як комікс. Комікс не відображає всієї дійсності: лінія не проходить через всі крапки даних. Але це корисне спрощення.



Лінія лінійної регресії

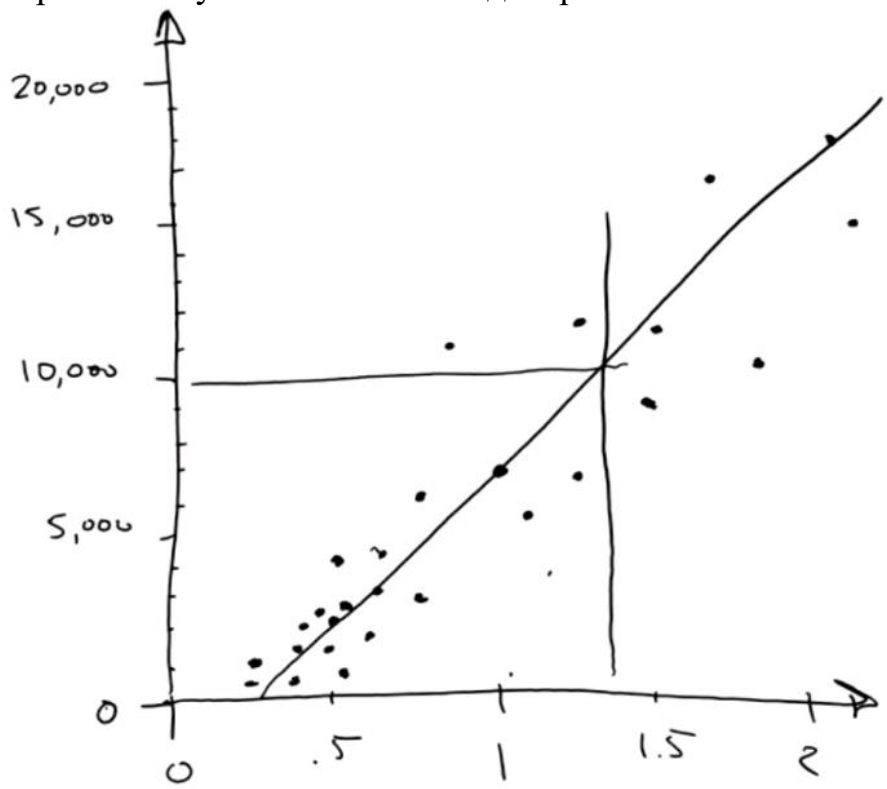
Той факт, що лінія не проходить через всі крапки, є нормальним. Фахівці з обробки даних пояснюють це так: існує модель (в нашому випадку - лінія), і кожна точка в моделі піддається впливу деякого шуму або відхилення, пов'язаного з нею. Є базова функціональна зв'язок, а є мінливий реальний світ, який додає шум і невизначеність.

Так як ми намагаємося відповісти на питання Скільки? , Це називається регресією. І оскільки ми використовуємо пряму лінію, це - лінійна регресія.

Використання моделі для пошуку відповіді

Тепер у нас є модель і ми можемо поставити їй наше запитання: "Скільки буде коштувати діамант масою 1,35 карата?"

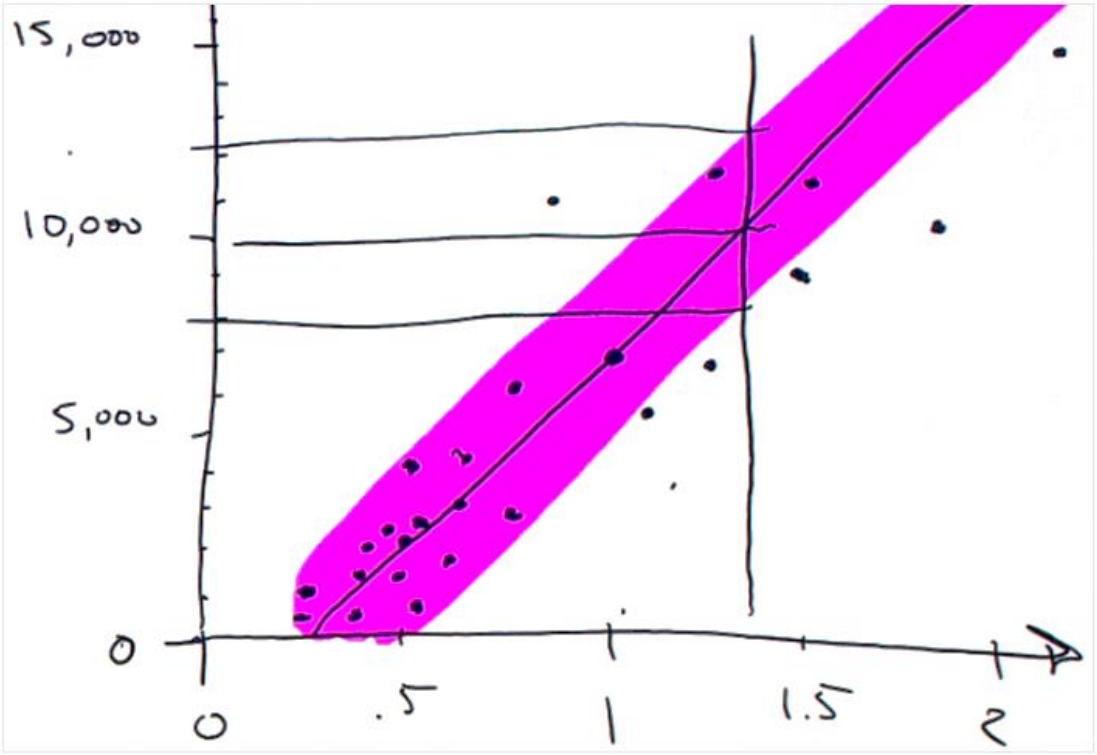
Для відповіді на питання ми проводимо вертикальну лінію від позначки 1,35 карата. Від точки її перетину з лінією моделі проводимо горизонтальну лінію до осі цін. Вона приводить нас до позначки 10 000. Ось так! Це і є відповідь: діамант масою 1,35 карата коштує близько 10 000 доларів США.



Пошук відповіді за допомогою моделі

Створення довірчого інтервалу

Було б логічним перевірити, наскільки точний цей прогноз. Корисно знати, чи буде діамант масою 1,35 карата коштувати рівно 10 000 доларів США, а може бути набагато більше або менше. Щоб це визначити, давайте намалюємо навколо лінії регресії так званий конверт, який буде включати в себе більшість точок. Цей конверт - наш довірчий інтервал. Ми можемо бути впевнені, що ціни потраплять в зазначений діапазон, адже це справедливо для більшості цін в минулому. Можна провести ще дві горизонтальні лінії від точок перетину лінії, проведеної від позначки 1,35 карата, з верхньою і нижньою межами даного конверта.



довірчі інтервали

Тепер ми можемо щось сказати про наш довірчий інтервал. Можна з упевненістю говорити, що ціна діаманта 1,35 карата складе близько 10 000 дол. США, при цьому вона може бути не нижче 8 000 і не вище 12 000 долл. США.

Тобто:

Ми поставили запитання, на яке можна відповісти за допомогою даних.

Ми створили модель, використовуючи лінійну регресію.

Ми зробили прогноз, доповнений довірчим інтервалом.

При цьому ми не користувалися математичними формулами або комп'ютерами.

Якби у нас були додаткові відомості, такі як:

огранювання діаманта;

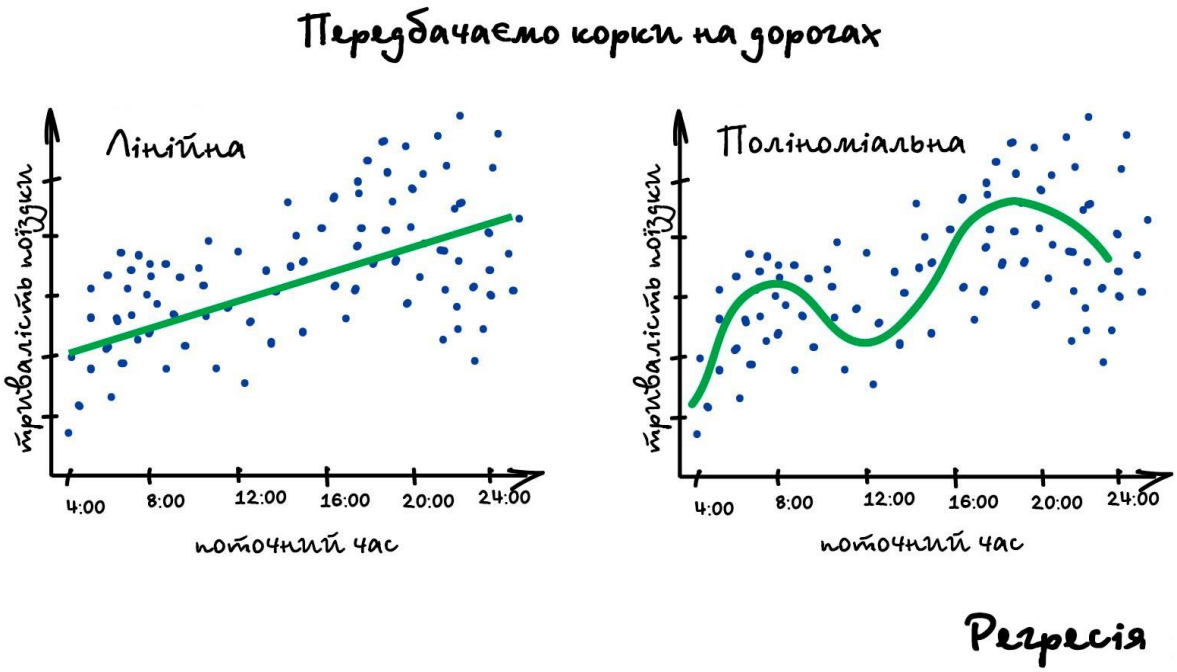
відтінки кольорів (наскільки близький колір діаманта до білого);

наявність в камені сторонніх включень;

то ми б мали додаткові стовпці. В цьому випадку математика вже буде корисною. Коли є більше двох стовпців, важко намалювати точки на папері. Математика дозволить відмінно вписати отриману лінію або площину в ваші дані.

Крім того, якби замість невеликого переліку діамантів у нас було б дві тисячі або два мільйони каменів, то набагато швидше цю роботу можна було б зробити за допомогою комп'ютера.

Сьогодні ми поговорили про те, як створити лінійну регресію, а також ми зробили прогноз, використовуючи дані.



Оцінки якості регресії

Найбільш типовими метриками якості в задачах регресії є

Середня квадратична помилка (англ. Mean Squared Error, MSE)

MSE застосовується в ситуаціях, коли нам треба підкреслити великі помилки і вибрати модель, яка дає менше помилок прогнозу. Грубі помилки стають помітнішими за рахунок того, що помилку прогнозу ми зводимо в квадрат. І модель, яка дає нам менше значення середньоквадратичної помилки, можна сказати, що що у цій моделі менше грубих помилок.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(a(x_i) - y_i)^2$$

Середня абсолютна помилка (англ. Mean Absolute Error, MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|a(x_i) - y_i|$$

Середньоквадратичний функціонал сильніше штрафує за великі відхилення в порівнянні з средньоабсолютним, і тому більш чутливий до викидів. При використанні будь-якого з цих двох функціоналів може бути корисно проаналізувати, які об'єкти вносять найбільший внесок у загальну помилку - не виключено, що на цих об'єктах була допущена помилка при обчисленні ознак або цільової величини.

Ефективне значення помилки підходить для порівняння двох моделей або для контролю якості під час навчання, але не дозволяє зробити висновків про те, на скільки добре дана модель вирішує завдання. Наприклад, MSE = 10 є дуже поганим показником, якщо цільова змінна приймає значення від 0 до 1, і дуже хорошим, якщо цільова змінна лежить в інтервалі (10000, 100000). У таких ситуаціях замість середньоквадратичної помилки корисно використовувати коефіцієнт детермінації - R^2

Коефіцієнт детермінації

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(a(x_i) - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Коефіцієнт детермінації вимірює частку дисперсії, внесеною моделлю, в загальній дисперсії цільової змінної. Фактично, дана міра якості - це нормована середньоквадратична помилка. Якщо вона близька до одиниці, то модель добре пояснює дані, якщо ж вона близька до нуля, то прогнози можна порівняти за якістю з сталою пророкуванням.

Середня абсолютна процентна помилка (англ. Mean Absolute Percentage Error, MAPE)

$$MAPE = 100\% \times \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - a(x_i)|}{|y_i|}$$

Це коефіцієнт, який не має розмірності, з дуже простою інтерпретацією. Його можна вимірювати в частках або відсотках. Якщо у вас вийшло, наприклад, що MAPE = 11.4%, то це говорить про те, що помилка склала 11,4% від фактичних значень. Основна проблема цієї помилки - нестабільність.

Корінь із середньої квадратичної помилки (англ. Root Mean Squared Error, RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y_i})^2}$$

Приблизно така ж проблема, як і в MAPE: так як кожне відхилення зводиться в квадрат, будь невелике відхилення може значно вплинути на показник помилки. Варто зазначити, що існує також помилка MSE, з якої RMSE якраз і виходить шляхом вилучення кореня.

Симетричная MAPE (англ. Symmetric MAPE, SMAPE)

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{2 \times |y_i - a(x_i)|}{|y_i| + |a(x_i)|}$$

Середня абсолютна масштабована помилка (англ. Mean absolute scaled error, MASE)

$$MASE = \frac{\sum_{i=1}^{n} |Y_i - e_i|}{\frac{n}{n-1} \sum_{i=2}^{n} |Y_i - Y_{i-1}|}$$

MASE є дуже хорошим варіантом для розрахунку точності, так як сама помилка не залежить від масштабів даних і є симетричною: тобто позитивні і негативні відхилення від факту розглядаються в рівній мірі. *Зверніть увагу, що в MASE ми маємо справу з двома сумами: та, що в чисельнику, відповідає тестової вибірці, та, що в знаменнику - навчальної.* Друга фактично являє собою середню абсолютну помилку прогнозу. Вона ж відповідає середньому абсолютному відхиленню ряду в перших різницях. Ця величина, по суті, показує, наскільки навчальна вибірка передбачувана. Вона може бути дорівнює нулю тільки в тому випадку, коли всі значення в навчальній вибірці рівні один одному, що відповідає відсутності будь-яких змін в ряді даних, ситуації на практиці майже неможливою. Крім того, якщо ряд має тенденцію до зростання або зниження, його перші різниці будуть коливатися близько деякого фіксованого рівня. В результаті цього з різних рядах з різною структурою, знаменники будуть більш-менш порівнянними. Все це, звичайно ж, є очевидними плюсами MASE, так як дозволяє складати різні значення за різними рядах і отримувати незміщені оцінки.

Недолік MASE в тому, що її важко інтерпретувати. Наприклад, MASE = 1.21 ні про що, по суті, не говорить. Це просто означає, що помилка прогнозу виявилася в 1.21 рази вище середнього абсолютного відхилення ряду в перших різницях, і нічого більше.

Крос-валідація

Хороший спосіб оцінки моделі передбачає застосування крос-валідації (ковзного контролю або перехресної перевірки).

В цьому випадку фіксується деяка множина розбиття вихідної вибірки на дві підвибірки: навчальну і контрольну. Для кожного розбиття виконується настройка алгоритму за навчальною підвибіркою, потім оцінюється його середня помилка на об'єктах контрольної підвибірки. Оцінкою ковзного контролю називається середня по всім розбиттям величина помилки на контрольних підвибірках.

4. НАВЧАННЯ БЕЗ ВЧИТЕЛЯ

Навчання без вчителя (Unsupervised Learning) було винайдено пізніше, аж у 90-ті, і на практиці використовується рідше. Але бувають задачі, де у нас просто немає вибору.

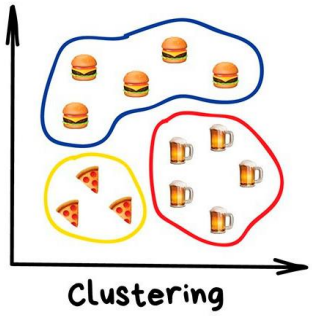
Розмічені дані - дорога рідкість. Але що робити якщо я хочу, наприклад, написати класифікатор автобусів - йти на вулицю, руками фотографувати мільйон Неопланів і підписувати де який? Так і все життя пройде, а у мене ще ігри в танчики не пройдено.

Коли немає міток, є надія на капіталізм, соціальне розшарування і мільйон китайців з сервісів типу Яндекс. Толока, які готові робити для вас що завгодно за п'ять центів. Так зазвичай і роблять на практиці. А ви думали де Яндекс бере більшість своїх датасетів?

Або можна спробувати навчання без учителя. Хоча, відверто кажучи, зі своєї практики я не пам'ятаю, щоб десь воно спрацювало добре.

Навчання без вчителя, все ж, частіше використовують як метод аналізу даних, а не як основний алгоритм. Спеціальний юзер з дипломом котрогось НУ вкидає туди купу сміття і спостерігає. Кластери є? Залежності з'явилися? Ні? Ну що ж, продовжуй, праця облагороджує. Ти ж хотів працювати в датасаєнсі.

4.1. Кластеризація



«Розділяє об'єкти за невідомою ознакою. Машина сама вирішує як краще»

Сьогодні використовують для:

Сегментація ринку (типів покупців, лояльності)

Об'єднання близьких точок на карті

Стиснення зображень

Аналіз і розмітки нових даних

Детектори аномальної поведінки

Популярні алгоритми: Метод К-середніх , Mean-Shift , DBSCAN

Кластеризація - це класифікація, але без заздалегідь відомих класів. Вона сама шукає схожі об'єкти та об'єднує їх в кластери. Кількість кластерів можна задати заздалегідь або довірити це машині. Схожість об'єктів машина визначає за тими ознаками, які ми їй розмітили - у кого багато схожих характеристик, тих давай в один клас.

Відмінний приклад кластеризації - маркери на картах в Інтернеті. Коли ви шукаєте всі крафтові бари у Львові, машині доводиться групувати їх в кружечки з циферками, інакше браузер зависне в потугах намалювати мільйон маркерів.

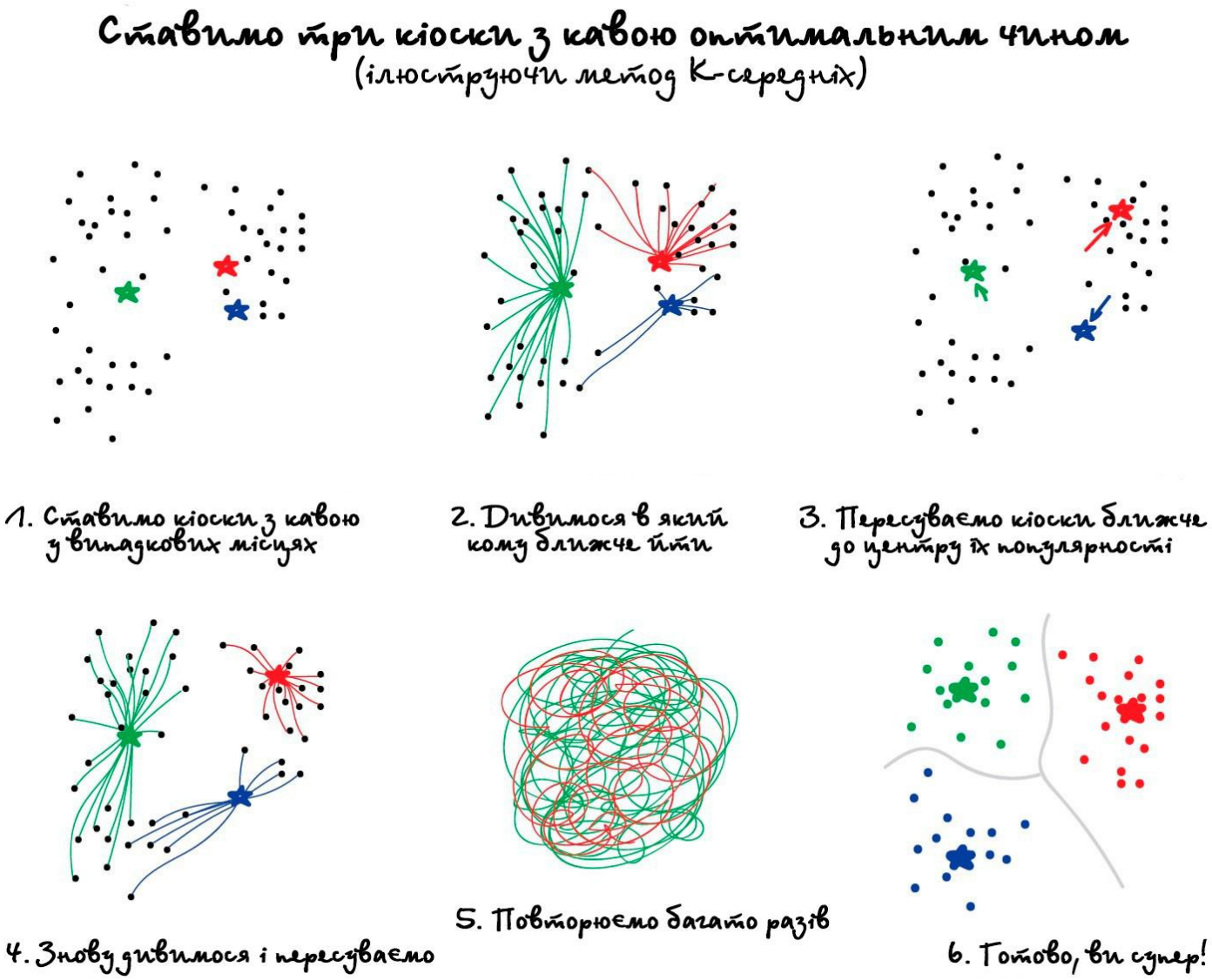
Більш складні приклади кластеризації можна згадати в додатках iPhoto або Google Photos, які знаходять особи людей на фотографіях і групують їх у альбоми. Додаток не знає як звуть ваших друзів, але може відрізнити їх за характерними рисами обличчя. Типова кластеризація.

Правда для початку їм доводиться знайти ці самі «характерні риси», а це вже тільки з учителем.

Стиснення зображень - ще одна популярна проблема. Зберігаючи картинку в PNG, ви можете встановити палітру, скажімо, в 32 кольори. Тоді кластеризація знайде всі «приблизно червоні» пікселі зображення, вирахує з них «середній червоний по лікарні» і замінить всі червоні на нього. Менше кольорів - менший файл.

Проблема тільки, як бути з кольорами типу Cyan - ось він ближче до зеленого чи синього? Тут нам допоможе популярний алгоритм кластеризації - Метод К-середніх (K-Means) . Ми випадковим чином кидаємо на палітру кольорів наші 32 точки, називаючи їх центроїдами. Всі інші точки відносимо до найближчого центроїду від них - виходять так би мовити сузір'я з найближчих кольорів. Потім рухаємо центроїд в центр свого сузір'я і

повторюємо поки центроїди не перестануть рухатися. Кластери виявлені, стабільні і їх рівно 32 як і треба було.

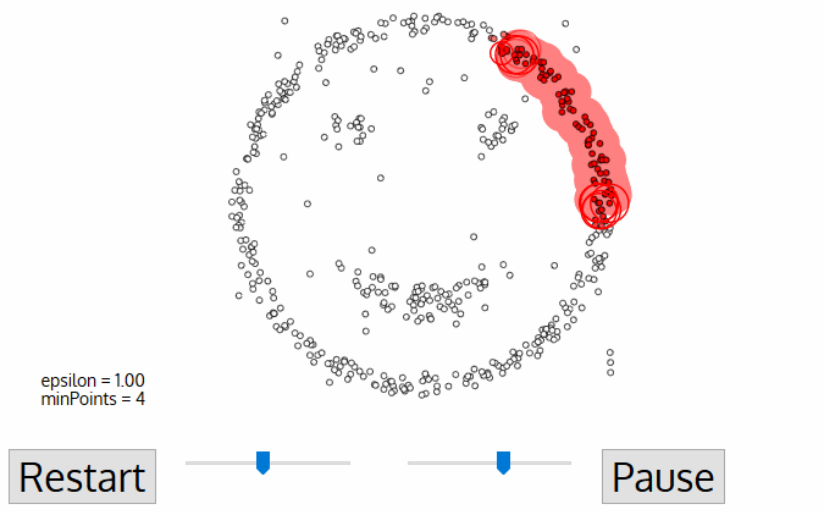


Шукати центроїди зручно і просто, але в реальних задачах кластери можуть бути зовсім не круглої форми. Ось ви геолог, якому потрібно знайти на карті схожі за структурою гірські породи - ваші кластери не тільки будуть вкладені один в одний, але ви ще й не знаєте скільки їх взагалі вийде.

Є більш досконалі способи кластеризації, але не всі знають, який коли слід застосовувати, і далеко не всі розуміють, як вони працюють.

Алгоритм DBSCAN

Хитрим завданням - хитрі методи. DBSCAN , наприклад. Він сам знаходить скупчення точок і будує навколо кластери. Його легко зрозуміти, якщо уявити, що точки - це люди на площі. Знаходимо трьох людей, які знаходяться близько одна до одної, і пропонуємо їм взятися за руки. Потім вони починають брати за руку тих, кого можуть досягти. Так по ланцюжку, поки ніхто більше не зможе взяти когось за руку - це і буде перший кластер. Повторюємо, поки не поділимо всіх. Ті, кому взагалі нікого брати за руку - це аномалії, викидаємо. В динаміці виглядає досить красиво:

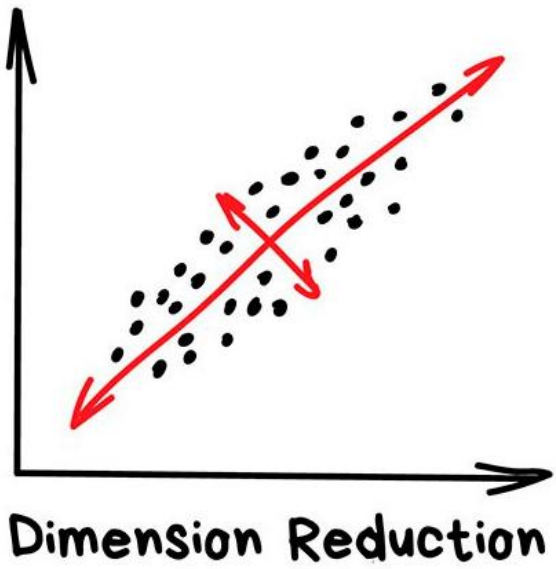


Тим, хто цікавиться кластеризацією, рекомендую статтю [The 5 Clustering Algorithms Data Scientists Need to Know]

Як і класифікація, кластеризація також може використовуватися **як детектор аномалій.** Поведінка користувача після реєстрації різко відрізняється від нормального? Заблокувати його і створити тікет саппорту, щоб перевірили бот це чи ні. При цьому нам навіть не треба знати, що є «нормальна поведінка» - ми просто вивантажуємо всі дії користувачів в модель, і нехай машина сама розбирається хто тут нормальний.

Працює такий підхід, в порівнянні з класифікацією, не дуже. Але за спробу не б'ють, раптом вийде.

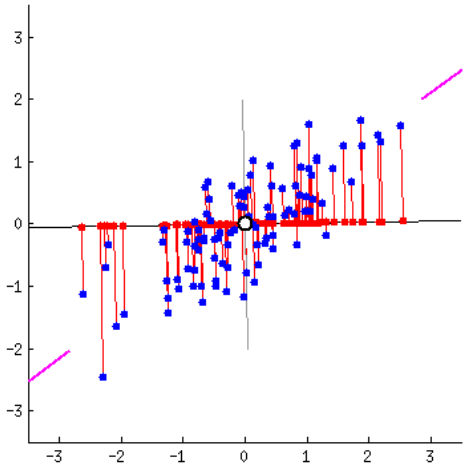
4.2. Зменшення Розмірності (Узагальнення)



«Збирає конкретні ознаки в абстракції вищого рівня»
Сьогодні використовують для:
Рекомендаційні Системи (★)
Красиві візуалізації
Визначення тематики та пошуку схожих документів
Аналіз фейковий зображень
Ризик-менеджмент

Популярні алгоритми: Метод головних компонент (PCA), Сингулярне розкладання (SVD), Латентне розміщення Діріхле (LDA), Латентно-семантичний аналіз (LSA, pLSA, GLSA), t-SNE (для візуалізації)

Спочатку це були методи хардкорних Data Scientist'ів, яким вивантажували дві фури цифр і говорили знайти там що-небудь цікаве. Коли просто будувати графіки в екселі вже не допомагало, вони придумали напружити машини шукати закономірності замість них. Так у них з'явилися методи, які назвали Dimension Reduction або Feature Learning.

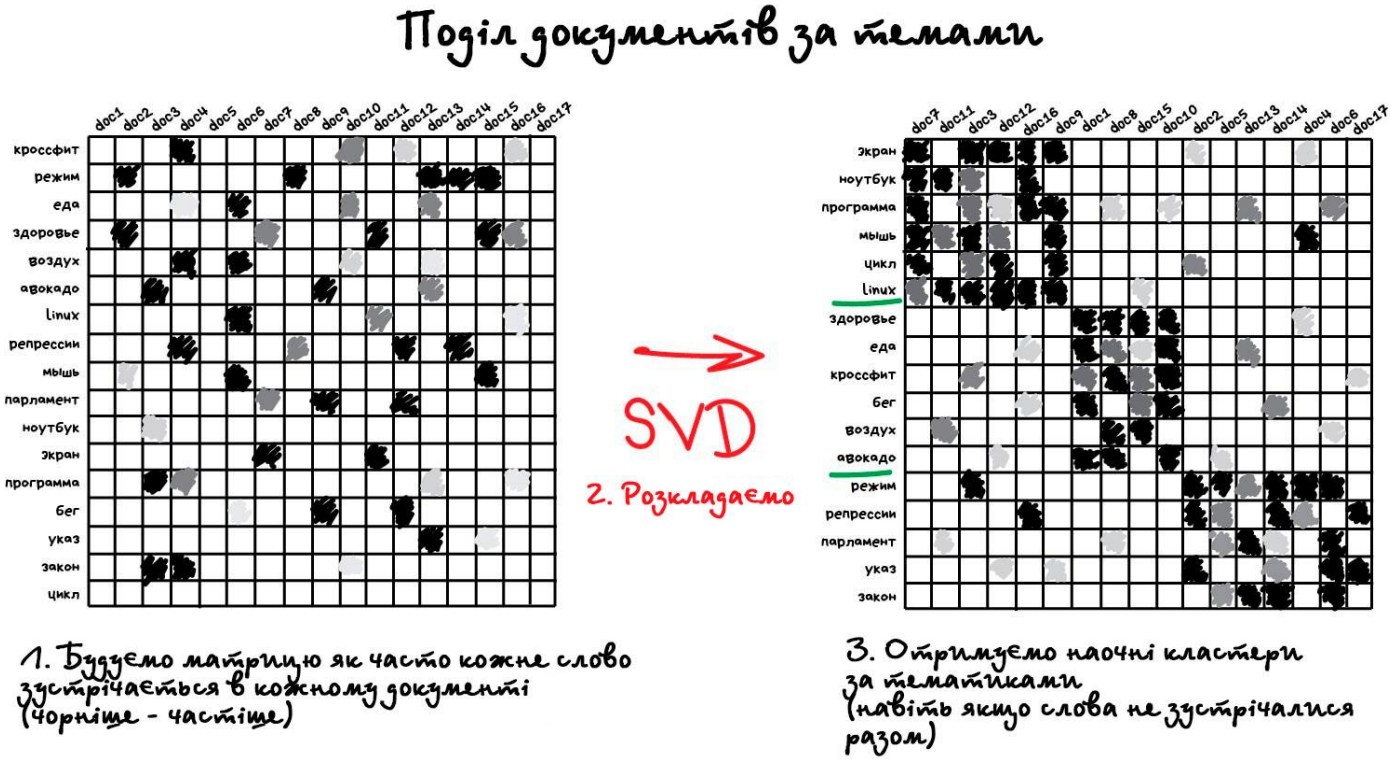


Для нас практична користь їх методів в тому, що ми можемо об'єднати декілька ознак в одну і отримати абстракцію. Наприклад, собаки з трикутними вухами, довгими носами і великими хвостами об'єднуються в корисну абстракцію «вівчарки». Очевидно, що ми втрачаємо інформацію про конкретних вівчарок, але нова абстракція по-любому корисніша цих зайвих деталей. Плюс, навчання на меншій кількості розмірностей йде сильно швидше.

Інструмент на диво добре підійшов для визначення тематик текстів (Topic Modelling). Ми змогли абстрагуватися від конкретних слів до рівня смислів навіть без залучення вчителя зі списком категорій. Алгоритм назвали Латентно-семантичний аналіз (LSA), і його ідея була в тому, що частота появи слова в тексті залежить від його тематики: в наукових статтях більше технічних термінів, в новинах про політику - імен політиків. Так, ми могли б просто взяти всі слова з статей і кластеризувати, як ми робили з кіосками вище, але тоді ми б втратили всі корисні зв'язки між словами, наприклад, що батарейка і акумулятор означають одне і те ж в різних документах.

Точність такої системи - повне дно, навіть не намагайтеся.

Потрібно якось об'єднати слова і документи в одну ознаку, щоб не втрачати ці приховані (латентні) зв'язку. Звідси і з'явилася назва методу. Виявилося, що Сингулярне розкладання (SVD) легко справляється з цим завданням, виявляючи для нас корисні тематичні кластери зі слів, які зустрічаються разом.



Латентно-семантичний Аналіз (LSA)

Для розуміння рекомендую статтю Як зменшити кількість вимірювань і отримати з цього користь, а практичне застосування добре описано в статті Алгоритм LSA для пошуку схожих документів .

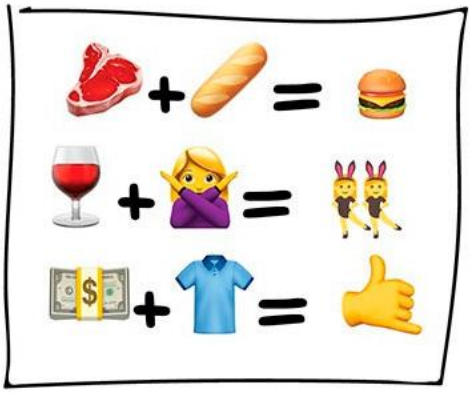
Інше мега-популярне застосування методу зменшення розмірності знайшли в рекомендаційних системах і колаборативній фільтрації. Виявилося, що якщо абстрагувати ними оцінки користувачів фільмів, виходить непогана система рекомендацій кіно, музики, ігор і чого завгодно взагалі.

Отримана абстракція буде важко зрозуміла мозком, але коли дослідники почали пильно розглядати нові ознаки, вони виявили, що якісь з них явно корелюють з віком користувача (діти частіше грали в майнкрафт і дивилися мультфільми), інші з певними жанрами кіно, а треті взагалі з синдромом пошуку глибокого сенсу життя.

Машина, яка не знала нічого, крім оцінок користувачів, змогла дістатися до таких високих матерій, навіть не розуміючи їх. Достойно. Далі можна проводити соцопитування і писати дипломні роботи про те, чому бородаті дядьки люблять дегенеративні мультики.

На цю тему є непогана лекція - Як працюють рекомендаційні системи

4.3. Пошук правил (асоціація)



«Шукає закономірності в потоці замовлень»

Сьогодні використовують для:

Прогноз акцій і розпродажів

Аналіз товарів, які купують разом

Розташування товарів на полицях

Аналіз патернів поведінки на веб-сайтах

Популярні алгоритми: Apriori, Euclat, FP-growth

Сюди входять всі методи аналізу продуктових кошиків, стратегій маркетингу і інших послідовностей.

Припустимо, покупець бере в дальньому кутку магазину пиво і йде на касу. Чи варто ставити на його шляху горішки? Чи часто люди беруть їх разом? Горішки з пивом, напевно так, але які ще товари купують разом? Коли ви власник мережі гіпермаркетів, відповідь для вас не завжди очевидна, але одне тактичне поліпшення в розташуванні товарів може принести гарний прибуток.

Це ж стосується інтернет-магазинів, де завдання ще цікавіше - за яким товаром покупець повернеться наступного разу?

З незрозумілих причин, пошук правил - найбільш погано продумана категорія серед всіх методів навчання. Класичні способи полягають в тупому переборі пар всіх куплених товарів за допомогою дерев або множин. Самі алгоритми працюють наполовину - можуть шукати закономірності, але не вміють узагальнювати або відтворювати їх на нових прикладах.

У реальності кожен великий ритейлер пиляє свій велосипед і жодних особливих проривів в цій області не зустрічається. Максимальний рівень технологій тут - створити систему рекомендацій, як в пункті вище. Хоча може я просто далекий від цієї області?


Юзер купив диван
ВІН ОБОЖНЮЄ ДИВАНИ!!!
Рекомендувати йому ще 148 диванів
amazon
ebay
AliExpress