**Модуль 4.1: Класифікація та оцінка роботи моделі**

**Метрики якості**

**Вступ**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%B2%D1%81%D1%82%D1%83%D0%BF)У цьому розділі ми обговоримо різні метрики оцінки якості, які застосовуються у машинному навчанні. Метрики якості можуть використовуватися для:

* Завдання функціоналу помилки (використовується під час навчання).
* Підбирання гіперпараметрів (використовується при вимірі якості на крос-валідації). У тому числі можна використовувати іншу метрику, яка відрізняється від метрики, за допомогою якої побудовано функціонал помилки.
* Оцінювання підсумкової моделі: чи придатна модель для вирішення задачі.

Далі ми обговоримо метрики, характерні для кожного з розглянутих раніше класів завдань машинного навчання.

**Метрики якості у завданнях регресії**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B8-%D1%8F%D0%BA%D0%BE%D1%81%D1%82%D1%96-%D1%83-%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F%D1%85-%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%96%D1%97)Розглянемо метрики, що використовуються у завданнях регресії. Деякі з них ми вже обговорювали у попередніх розділах.

**Середньоквадратична помилка**[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%8C%D0%BE%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B0-%D0%BF%D0%BE%D0%BC%D0%B8%D0%BB%D0%BA%D0%B0)

**

Такий функціонал легко оптимізувати, використовуючи метод градієнтного спуску. Цей функціонал сильно штрафує за великі помилки, тому що відхилення зводяться у квадрат.

**Середня абсолютна помилка**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%8F-%D0%B0%D0%B1%D1%81%D0%BE%D0%BB%D1%8E%D1%82%D0%BD%D0%B0-%D0%BF%D0%BE%D0%BC%D0%B8%D0%BB%D0%BA%D0%B0)Схожий на попередній функціонал якості - середня абсолютна помилка:

*MAE*(*a*,*X*)=*n*1​∑*i*=1*n*​∣*a*(*xi*​)−*yi*​∣

Цей функціонал складніше мінімізувати, тому що у модуля похідна не існує в нулі.

**Коефіцієнт детермінації**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BA%D0%BE%D0%B5%D1%84%D1%96%D1%86%D1%96%D1%94%D0%BD%D1%82-%D0%B4%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D1%96%D0%BD%D0%B0%D1%86%D1%96%D1%97)Коефіцієнт детермінації  *R*2(*a*,*X*) визначається наступним чином

*R*2(*a*,*X*)=1−∑*i*=1*n*​(*yi*​−*y*ˉ​)2∑*i*=1*n*​(*a*(*xi*​)−*yi*​)2​

​

Дана величина показує, яку частку дисперсії (різноманіття відповідей) у всьому цільовому векторі y*y* модель змогла пояснити. Це дозволяє інтерпретувати значення середньоквадратичної помилки. Для розумних моделей коефіцієнт детермінації лежить у межах 0≤*R*2≤1. Причому випадок *R*2=1 відповідає випадку ідеальної моделі, *R*2=0 - моделі на рівні оптимальної "константної", а *R*2<0 - моделі гірші "константної" (такі алгоритми ніколи не потрібно розглядати). Оптимальним константним алгоритмом називається такий алгоритм, який завжди повертає середнє значення відповідей ​​ для об'єктів навчальної вибірки.

**Несиметричні втрати**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BD%D0%B5%D1%81%D0%B8%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D1%96-%D0%B2%D1%82%D1%80%D0%B0%D1%82%D0%B8)Для того, щоб зрозуміти, що таке несиметричні втрати, розглянемо наступний приклад. Нехай потрібно оцінити попит на ноутбуки. У цьому випадку занижений прогноз призведе до втрати лояльності покупців та потенційного прибутку (буде закуплено недостатню кількість ноутбуків), а завищений - до невеликих додаткових витрат на зберігання непроданого товару. Щоб врахувати цей факт, функція втрат має бути несиметричною та сильніше штрафувати за недопрогноз, ніж за перепрогноз. У завданнях, у яких помилки мають різну ціну, використовують несиметричні функції втрат. Прикладом такої функції може бути **квантильна помилка**, визначається наступним чином:

*ρτ*​(*a*,*X*)=*n*1​∑*i*=1*n*​((1−*τ*)[*yi*​<*a*(*xi*​)]+*τ*[*yi*​≥*a*(*xi*​)])(*yi*​−*a*(*xi*​))

Параметр *τ*∈[0,1] визначає те, за що треба штрафувати сильніше — за недопрогноз чи перепрогноз.

**Метрики якості у завданнях класифікації**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B8-%D1%8F%D0%BA%D0%BE%D1%81%D1%82%D1%96-%D1%83-%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%8F%D1%85-%D0%BA%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%97)

Очевидно, що метрики, що використовуються в регресії, не підходять для оцінки моделей, що вирішують завдання класифікації. Для таких моделей розроблено свої способи оцінки якості. Розглянемо деякі з них.

**Частка правильних відповідей**

[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D1%87%D0%B0%D1%81%D1%82%D0%BA%D0%B0-%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%BB%D1%8C%D0%BD%D0%B8%D1%85-%D0%B2%D1%96%D0%B4%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%B5%D0%B9)

Як міру якості у задачах класифікації природно використовувати частку неправильних відповідей:

1n∑i=1n[a(xi)≠yi]

Однак у завданнях класифікації прийнято вибирати метрики таким чином, щоб їх потрібно було максимізувати, тоді як у задачах регресії - так, щоб їх треба було мінімізувати. Враховуючи вищесказане, введемо величину, що характеризує **частку правильних відповідей** класифікатора:

*accuracy*(*a*,*X*)=*n*1​∑*i*=1*n*​[*a*(*xi*​)=*yi*​]

Ця метрика якості проста і широко використовується, проте має кілька суттєвих недоліків.

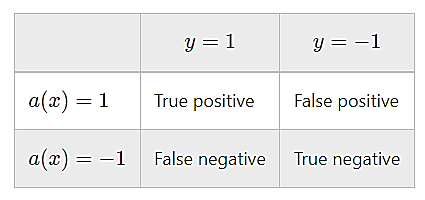
Перша проблема пов'язана із незбалансованими вибірками. Розглянемо приклад. Нехай у вибірці 1000 об'єктів, з яких 950 відносяться до класу −1 та 50  до класу +1. Розглянемо непотрібний константний класифікатор, який на всіх об'єктах дає відповідь −1. Частка правильних відповідей для такого класифікатора дорівнюватиме 0.95. Видно, що така метрика не дає нам повної картини того, на скільки класифікатору можна довіряти.

Тому, якщо виходить високий відсоток правильних відповідей, це може бути пов'язане не з тим, що побудовано хороший класифікатор, а з тим, що якогось класу дуже більше, ніж інших. Щоб "боротися" з цією проблемою, використовується наступний факт. Нехай q0*q*0​ - частка об'єктів найбільшого класу, тоді частка правильних відповідей для розумних алгоритмів повинна перебувати в проміжку [*q*0​,1].

Друга проблема з часткою правильних відповідей полягає в тому, що вона ніяк не враховує різні ціни різних типів помилок. Тоді як ціни справді можуть бути різними. Наприклад, нехай у задачі кредитного скорингу було побудовано дві моделі. При використання першої моделі кредит буде видано 100 клієнтам,  80 з яких його повернуть. Друга ж модель видає кредит лише 50 клієнтам, причому повернуть його в 48 випадках. Те, яка з двох моделей краще, залежить від того, що для нас важливіше: не дати кредит клієнту, який міг би його повернути, або видати кредит клієнту, який його не поверне.Таким чином, потрібні додаткові метрики якості, які враховують ціни тієї чи іншої помилки.

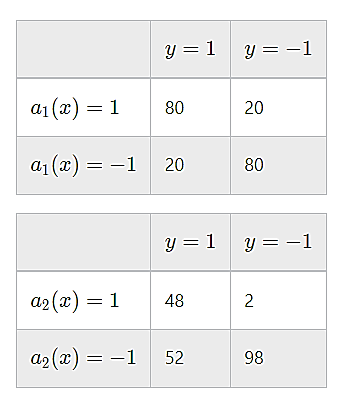
**Точність та повнота**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D1%82%D0%BE%D1%87%D0%BD%D1%96%D1%81%D1%82%D1%8C-%D1%82%D0%B0-%D0%BF%D0%BE%D0%B2%D0%BD%D0%BE%D1%82%D0%B0)Розглянемо метрики якості класифікації, що дозволяють враховувати різні ціни помилок. Співвідношення між результатами роботи класифікатора та справжніми відповідями зручно аналізувати за допомогою матриці помилок.



Коли алгоритм відносить об'єкт до класу +1, кажуть, що алгоритм спрацьовує. Якщо алгоритм спрацював і об'єкт дійсно відноситься до класу +1, має місце вірне спрацювання (true positive), а якщо об'єкт насправді відноситься до класу −1, має місце хибне спрацювання (false positive). Якщо алгоритм відповідає −1, кажуть, що він пропускає об'єкт. Якщо має місце пропуск об'єкта класу +1, то це хибний пропуск (false negative). Якщо ж алгоритм пропускає об'єкт класу −1, має місце справжній пропуск (true negative). Таким чином, існують два види помилок: хибні спрацьовування та хибні пропуски. Для кожного з них потрібна своя метрика якості, щоб виміряти, скільки помилок якого типу відбувається.

Розглянемо наступний приклад. Нехай розглядаються дві моделі a1(x)*a*1​(*x*) та a2(x)*a*2​(*x*). Вибірка складається з 200200 об'єктів, з яких 100100 відносяться до класу 11 та 100100 - до класу −1−1. Матриці помилок мають вигляд:



Введемо дві метрики. Перша метрика має назву точність (precision):

*precision*(*a*,*X*)=*TP*+*FPTP*​

Ця метрика показує наскільки можна довіряти класифікатору у разі спрацьовування.

Друга метрика, яку нам належить ввести називається повнота (recall) і визначається як:

recall(a,X)=TPTP+NF*recall*(*a*,*X*)=*TP*+*NFTP*​

Ця величина показує, на якій частці істинних об'єктів першого класу спрацьовує алгоритм.

У прикладі вище точність та повнота першого алгоритму виявляються рівними:

*precision*(*a*1​,*X*)=0.8

*recall*(*a*1​,*X*)=0.8

Друга модель є дуже точною, але на збиток повноті.

*precision*(*a*2​,*X*)=0.96

*recall*(*a*2​,*X*)=0.48

Точність показує, наскільки можна довіряти класифікатору у разі спрацювання. Повнота показує, на якій частці істинних об'єктів першого класу алгоритм спрацьовує. Підведемо підсумок.

**Точність** класифікатора – це частка об'єктів що дійсно належать даному класу відносно всіх об'єктів, які класифікатор відніс до цього класу.

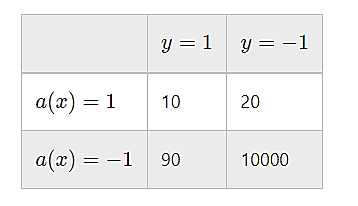
**Повнота** класифікатора – це частка об'єктів що належать класу відносно всіх об'єктів цього класу у вибірці.

**Приклади використання точності та повноти**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%B8-%D0%B2%D0%B8%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%BD%D0%BD%D1%8F-%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%96-%D1%82%D0%B0-%D0%BF%D0%BE%D0%B2%D0%BD%D0%BE%D1%82%D0%B8)Нехай у задачі кредитного скорингу ставиться умова, що невдалих кредитів має бути не більше 5%

5%. В такому випадку завдання є завданням максимізації **повноти** за умови *precision*(*a*,*X*)≥0.95.

Інший приклад. Нехай необхідно побудувати модель, яка визначає, чи є чи ні певне захворювання у пацієнта. При цьому потрібно, щоб були виявлені як мінімум 80% пацієнтів, які справді мають дане захворювання. Тоді ставлять завдання максимізації точності за умови *recall*(*a*,*X*)≥0.8. Слід особливо звернути увагу на те, як точність та повнота працюють у разі незбалансованих вибірок. Нехай розглядається вибірка з наступною матрицею помилок:



Частка вірних відповідей (accuracy), точність (precision) та повнота (recall) для даного випадку:

*accuracy*(*a*,*X*)=0.99

*precision*(*a*,*X*)=0.33

*recall*(*a*,*X*)=0.1

Те, що частка вірних відповідей дорівнює 0.99, ні про що не каже: алгоритм все одно робить 66% хибних спрацьовувань і виявляє тільки 10% позитивних випадків. Завдяки введенню точності та повноти стає зрозуміло, що алгоритм потрібно покращувати.

**Об'єднання точності та повноти**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BE%D0%B1%D1%94%D0%B4%D0%BD%D0%B0%D0%BD%D0%BD%D1%8F-%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%96-%D1%82%D0%B0-%D0%BF%D0%BE%D0%B2%D0%BD%D0%BE%D1%82%D0%B8)У деяких завданнях є обмеження на одну з цих метрик, тоді як за другою метрикою буде проводитися оптимізація. Але в деяких випадках хочеться максимізувати і точність, і повноту водночас. Постає питання про поєднання цих двох метрик.

**Арифметичне середнє**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%BD%D0%B5-%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%94)Єдина метрика може бути отримана як середнє арифметичне точності та повноти:

*A*=2*precision*+*recall*​

​

На жаль, ця метрика не позбавлена недоліків. Нехай є алгоритм, точність якого дорівнює 10%, а повнота — 100%: precision = 0.1 recall = 1. Це може бути випадок, коли у вибірці всього 10% об'єктів класу +1, а алгоритм є константним і завжди повертає +1. Очевидно, що цей алгоритм поганий, але введена вище метрика для нього рівна *A*=0.55. У свою чергу інший, набагато кращий алгоритм, з *precision*=0.55 та *recall*=55 теж характеризується *A*=0.55. Ситуація, коли константний і розумний алгоритми можуть лежати на одній лінії, є неприпустимою, тому слід шукати інший спосіб побудови єдиної метрики.

**Мінімум**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BC%D1%96%D0%BD%D1%96%D0%BC%D1%83%D0%BC)Щоб константний та розумний алгоритми не лежали на одній лінії рівня, можна розглядати

M=min(precision,recall)

Цей підхід вирішує вищезгадану проблему. Бо при precision=0.05 та recall=1 отримуємо, що M=0.05.

Але є інша проблема - два алгоритми, для яких точності однакові, але відрізняються значення повноти, лежатимуть на одній лінії рівня *M*. Наприклад, розглянемо два алгоритми. Нехай для першого *precision*=0.4, *recall*=0.5, значить *M*=0.4. А для другого *precision*=0.4, *recall*=0.9, звідки виходить, що *M*=0.4. Таке теж неприпустимо, так як другий алгоритм суттєво кращий за перший.

**F-міра**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#f-%D0%BC%D1%96%D1%80%D0%B0)"Згладити" мінімум можна за допомогою гармонійного середнього, або F-міри:

*F*=*precision*1​+*recall*1​2​=2⋅*precision*+*recallprecision*⋅*recall*​

​

Для двох згаданих вище алгоритмів значення F-міри, на відміну M, відрізнятимуться. Для першого значення F-міри становить 0.44, а для другого - 0.55.

Якщо необхідно віддати перевагу точності або повноті, слід використовувати розширену F-міру, в якій є параметр, використовується параметр *β*:

*F*=(1+*β*2)⋅*β*2*precision*+*recallprecision*⋅*recall*​

​

**Якість оцінок приналежності класу**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D1%8F%D0%BA%D1%96%D1%81%D1%82%D1%8C-%D0%BE%D1%86%D1%96%D0%BD%D0%BE%D0%BA-%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%BB%D0%B5%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%96-%D0%BA%D0%BB%D0%B0%D1%81%D1%83)

Багато алгоритмів бінарної класифікації влаштовані наступним чином: спочатку обчислюється деяке дійсне число b(x)*b*(*x*), яке порівнюється з порогом t*t*:

*a*(*x*)=[*b*(*x*)>*t*]

Тут *b*(*x*) - оцінка приналежності класу +1. Іншими словами, *b*(*x*) виступає в ролі деякої оцінки впевненості, що x належить класу +1. У разі лінійного класифікатора *a*(*x*)=[⟨*ω*,*x*⟩>*t*] оцінка приналежності класу +1 має вигляд *b*(*x*)=⟨*w*,*x*⟩. Часто буває необхідно оцінити якість саме оцінки приналежності, а поріг вибирається пізніше з міркувань на точність чи повноту.

**Оцінка приналежності в завданні кредитного скорингу**

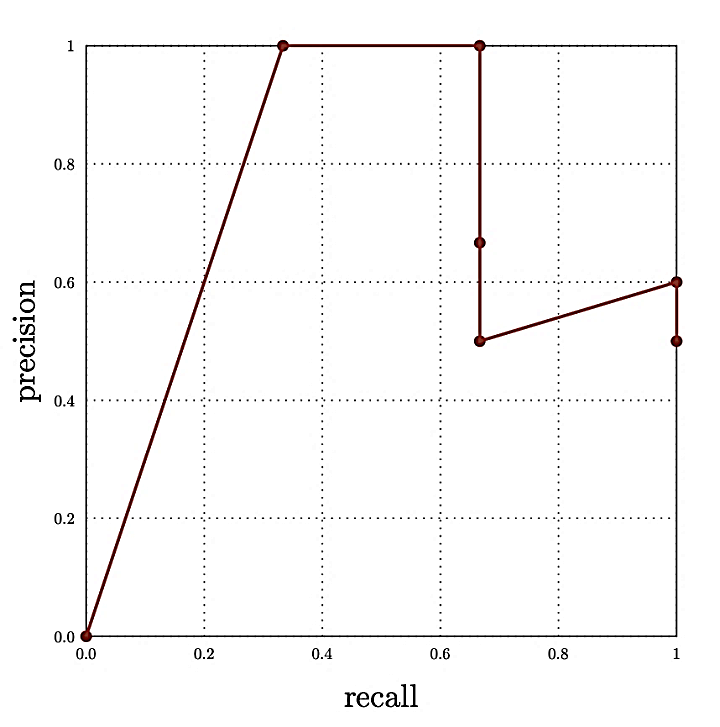
[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BE%D1%86%D1%96%D0%BD%D0%BA%D0%B0-%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%BB%D0%B5%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%96-%D0%B2-%D0%B7%D0%B0%D0%B2%D0%B4%D0%B0%D0%BD%D0%BD%D1%96-%D0%BA%D1%80%D0%B5%D0%B4%D0%B8%D1%82%D0%BD%D0%BE%D0%B3%D0%BE-%D1%81%D0%BA%D0%BE%D1%)Нехай розглядається задачний кредитний скоринг і була побудована деяка функція *b*(*x*), яка оцінює можливість повернення кредиту клієнтом x. Далі класифікатор будується наступним чином:

*a*(*x*)=[*b*(*x*)>0.5]

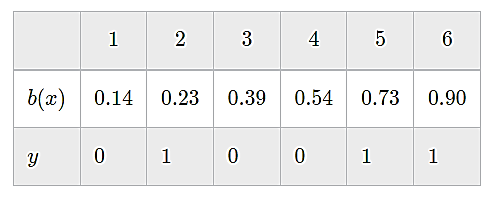
При цьому вийшло, що точність (precision) рівна 10%, а повнота (recall) - 70%. Це дуже поганий алгоритм, так як 90% клієнтів, яким буде видано кредит, не повернуть його. При цьому не зрозуміло, в чому річ: був погано обраний поріг або алгоритм не підходить для вирішення цього завдання. Саме для цього необхідно вимірювати якість самих оцінок *b*(*x*).

**PR-крива**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#pr-%D0%BA%D1%80%D0%B8%D0%B2%D0%B0)Перший спосіб оцінки належності класу заснований на використанні кривої точності-повноти. По осі *X* відкладається повнота, а по осі *Y* — точність. Кожній точці на цій кривій буде відповідати класифікатор з деяким значенням порога.



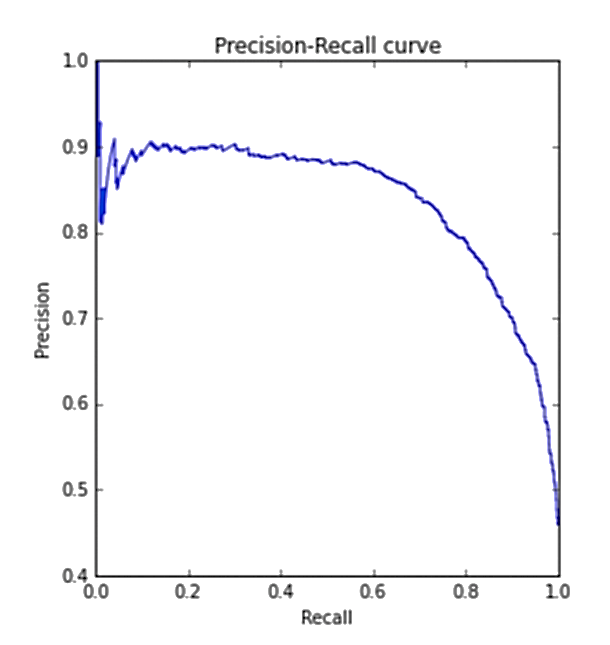
Для прикладу буде наведено побудову PR-кривої для вибірки з 6 об'єктів, три з яких відносяться до класу 1 та 3 — до класу 0. Відповідний їй графік зображено вище.



**Пояснення таблиці**

1. При досить великому порозі жоден об'єкт не буде віднесений до класу 1. У цьому випадку і точність, і повнота дорівнюють 0.
2. При такому порозі, що рівно один об'єкт віднесено до класу 1, точність буде 100% (оскільки цей об'єкт дійсно з 1 класу), а повнота — 1331​​ (оскільки всього 3 об'єкта 1 класу).
3. При подальшому зменшенні порога вже два об'єкти віднесено до класу 1, точність також залишається 100%, а повнота стає рівною 2332​.
4. При такому порозі, що вже три об'єкти будуть віднесені до класу 1, точність стає рівною 2332​​, а повнота залишається такою ж.
5. При такому порозі, що чотири об'єкти віднесено до класу 1, точність зменшиться до 0.5, а повнота знову не зміниться.
6. При подальшому зменшенні порога вже 5 об'єктів будуть віднесені до 1 класу, повнота стане рівною 100% а точність - 3553​

У реальних завданнях з кількістю об'єктів близько кількох тисяч або десятків тисяч, крива точності - повноти виглядає приблизно наступним чином.



Слід зазначити, що починається PR-крива завжди з точки (0,0), а закінчується точкою (1,r), де *r* — частка об'єктів класу 1. У разі ідеального класифікатора, тобто якщо існує такий поріг, що і точність, і повнота рівні 100%, крива проходитиме через точку (1,1). Таким чином, чим ближче крива пройде до цієї точки, тим краще оцінки. Площа під цією кривою може бути гарним заходом якості оцінок приналежності до класу 1. Така метрика називається AUC–PRC, або площа під PR-кривою.

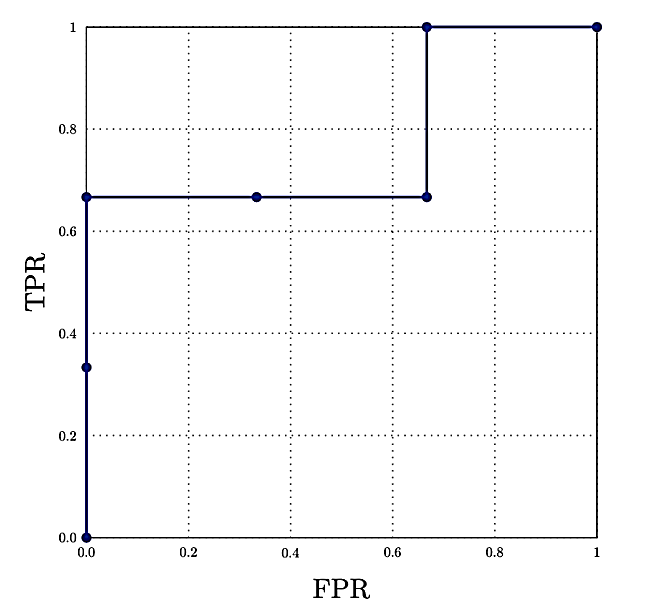
**OC-крива**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#roc-%D0%BA%D1%80%D0%B8%D0%B2%D0%B0)Другий спосіб виміряти якість оцінок приналежності до класу 1 — ROC-крива, яка будується в осях False Positive Rate (ось X*X*) та True Positive Rate (ось *Y*):

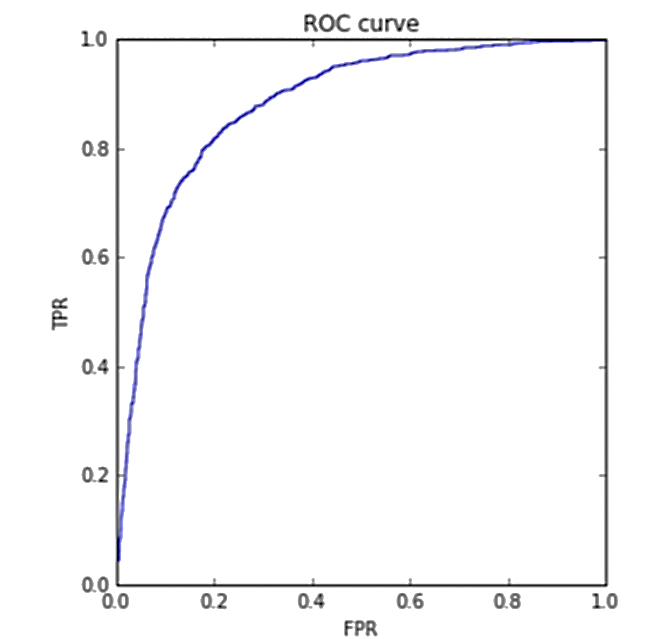
*FRP*=*FP*+*TNFP*​

​*TPR*=*TP*+*FNTP*​

​ROC-крива будується аналогічно PR-кривий: поступово розглядаються випадки різних значень порогів та відзначаються точки на графіку. Для згаданої вище вибірки ROC-крива має наступний вигляд:



У випадку великої вибірки ROC-крива виглядає наступним чином:



Крива стартує з точки (0,0) і приходить в точку (1,1). При цьому якщо існує ідеальний класифікатор, крива має пройти через точку (0,1). Чим ближче крива до цієї точки, тим краще будуть оцінки, а площа під кривою буде характеризувати якість оцінок належності до першого класу. Така метрика називається AUC–ROC, або площа під ROC-кривої.

**Особливості AUC-ROC та AUC-PRC**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%BE%D1%81%D0%BE%D0%B1%D0%BB%D0%B8%D0%B2%D0%BE%D1%81%D1%82%D1%96-auc-roc-%D1%82%D0%B0-auc-prc)Як було написано вище, ROC-крива будується в осях FPR та TPR, які нормуються на розміри класів:

*FPR*=*FP*+*TNFP*​

*TPR*=*TP*+*FNTP*​​

Отже, при зміні балансу класів величина AUC-ROC та незмінні властивості об'єктів вибірки площа під ROC-кривій не зміниться. У разі ідеального алгоритму *AUC*−*ROC*=1, а у разі гіршого *AUC*−*ROC*=1/2​. Значення AUC−ROC має сенс ймовірності того, що якщо були обрані випадковий позитивний та випадковий негативний об'єкти вибірки, позитивний об'єкт отримає оцінку приналежності вище, ніж негативний об'єкт.

З іншого боку PR-крива будується в осях precision та recall:

precision(a,X)=TPTP+FP*precision*(*a*,*X*)=*TP*+*FPTP*​

​

recall(a,X)=TPTP+NF*recall*(*a*,*X*)=*TP*+*NFTP*​

а отже змінюється за зміни балансу класів.

**Замість висновку**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-04/main-02#%D0%B7%D0%B0%D0%BC%D1%96%D1%81%D1%82%D1%8C-%D0%B2%D0%B8%D1%81%D0%BD%D0%BE%D0%B2%D0%BA%D1%83)

Ми з вами обговорили основні метрики оцінки якості моделей у задачі регресії та класифікації. Розглянутий список далеко не є вичерпним. Проте навіть у такому обсязі він дозволяє нам зрозуміти, чому недостатньо просто скористатися однією з тривіальних метрик, які ми обговорювали раніше. Імплементація деяких із них може бути дуже нетривіальним завданням. Тому на практиці рекомендується користуватися чимось готовим. Наприклад, велику кількість метрик якості можна знайти у модулі [metrics](https://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.metrics" \t "_blank) (https://scikit-learn.org/stable/api/index.html) бібліотеки sklearn.