КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра обчислювальної математики

Кваліфікаційна робота

на здобуття ступеня бакалавра
за спеціальністю 113 Прикладна математика
на тему:

ПОРІВНЯННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ПО ДЕКІЛЬКОХ МЕТРИКАХ В ЗАДАЧАХ КЛАСИФІКАЦІЇ

Виконав студент 4-го курсу
Сокоренко Андрій Сергійович _____
(підпис)

Науковий керівник:
Доцент кафедри ОМ,
Голубєва Катерина Миколаївна _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань

Студент _____
(підпис)

Роботу розглянуто і допущено до захисту на засіданні кафедри обчислювальної математики
«___» _____2021р.,
Протокол №
Завідувач кафедри
Ляшко С.І _____
(підпис)

Київ – 2021

Зміст

Вступ

Однією з найбільш актуальних задач, яку дозволяє ефективно розв’язувати машинне навчання, є задача класифікації. Суть цієї задачі полягає у віднесенні об’єкту до певного класу на основі характеристик цього об’єкту. Прикладом такої задачі є ідентифікація користувача по його поведінці в мережі інтернет, коли необхідно по послідовності з декількох веб-сайтів, відвіданих поспіль одним і тим самим користувачем, ідентифікувати цю людину.

Задачі класифікації історично були першими, для яких застосовувались алгоритми машинного навчання. Тому на даний момент існує багато підходів та алгоритмів для таких задач. Але разом з тим, постійно ведеться дослідження щодо шляхів вдосконалення цих алгоритмів та пошуку нових, ще більш ефективних методів. В рамках таких досліджень важливою складовою є як теоретичне вивчення властивостей існуючих алгоритмів, так і практичне дослідження їх ефективності.

З метою такого практичного дослідження, в роботі розглядаються кілька актуальних алгоритмів машинного навчання, аналізується їх ефективність та виконується порівняння на чотирьох модельних задачах класифікації.

Метрики для оцінювання ефективності алгоритмів

Для порівняння ефективності важливим є вибір метрик — деяких показників, які дозволяють оцінити основні властивості алгоритму. Адже легко можна потрапити в ситуацію, коли один і той самий алгоритм можна назвати ефективним або ні в залежності від того, які параметри оцінювати. Є багато варіантів метрик, за якими є сенс оцінювати алгоритми класифікації - але з основних можна виділити наступні:

- Метрика якості классифікації
- Час необхідний для здійснення классифікації об’єкта певним алгоритмом
- Розмір моделі классифікатора

Метрика якості класифікації — це метрика по якій безпосередньо вимірюється точність або правильність роботи алгоритма, тобто чим більший показник такої метрики – тим ймовірніше що алгоритм правильно классифікує об’єкт. Очевидно що ця метрика є ключовою, оскільки показує те наскільки алгоритм може справлятися зі своєю роботою – классифікувати об’єкти.

Наступна метрика — час необхідний для здійснення классифікації. Ця метрика показує наскільки швидко алгоритм классифікує об’єкти. Вимірюється у кількості часу необхідного для классифікації N об’єктів. Деколи при постановці задачі ставиться обмеження по цій метриці. Наприклад, необхідно розробити додаток до смартфону який буде по зображенню із камери классифікувати об’єкт перед камерою впродовж 1 секунди. Навіщо необхідне це обмеження? – бо нікому не цікаво чекати годину поки смартфон видасть результат. Зауважимо що в цій роботі ми не будемо ставити обмеження по цій метриці, але показники по ній знімати будемо, адже вони важливі при порівнянні моделей.

Розмір моделі классифікатора – визначає розмір моделі у мегабайтах. На цей показник теж можуть накладатися обмеження. В нашій задачі ми не будемо накладати обмежень на цю метрику.

Необхідно зазначити що оптимізовувати будемо метрику якості, а дві інші метрики будемо обраховувати вже після тренування моделей.

Модельні задачі

Окрім метрики, важливим є вибір модельних (тестових) задач. В ідеалі, набір тестових задач повинен накривати основні класи задач, для яких алгоритм буде застосовуватись на практиці. В даній роботі в якості модельних задач обрано наступні:

- Оцінка задоволення пасажирів авіакомпанією
- Ідентифікація користувача по його поведінці
- Модель задоволення клієнтів Сантандера
- Передбачення заявок на страхове відшкодування водієм.

Далі розглянемо та розв'яжемо ці задачі, а також не забудемо виміряти ключові метрики необхідні для порівняння методів машинного навчання. Варто зазначити що всі задачі – рельні задачі розміщені на платформі Kaggle.com.

Задача 1. Задоволення клієнтів Сантандера (ориг. Santander Customer Satisfaction) посилання

Опис.

Задоволення клієнтів - ключовий показник успіху будь якого бізнесу. Нещасні клієнти довго не тримаються. Більше того, нещасні клієнти рідко висловлюють своє невдоволення, та мовчки переходять до конкурентів.

Банк Сантандер просить допомогти їм виявити незадоволених клієнтів на початку їхніх стосунків. Це дозволить Сантандеру вживати активних заходів для покращення задоволення клієнта, поки не пізно. Адже утримання існуючих клієнтів дешевше за залучення нових.

У цій задачі маючи сотні анонімізованих характеристик, необхідно передбачити передбачити, чи задоволений клієнт своїм банківським досвідом.

Задача.

Ви маєте анонімізований набір даних, що містить велику кількість числових змінних. Стовпець "TARGET" - це змінна для прогнозування. Він дорівнює 1 для незадоволених клієнтів та 0 для задоволених клієнтів.

Завдання полягає в тому, щоб передбачити ймовірність того, що клієнт буде незадоволеним

Метрика якості – площа під ROC кривою.

Розмір датасету – 56 мб.; 370 стовпчиків, 76020 рядків.

Оскільки дані анонімізовані то інтерпретація данних відсутня.

Задача 2. Чи буде водій подавати страхове відшкодування? (ориг. Porto Seguro’s Safe Driver Prediction) посилання

Опис.

Ніщо не руйнує гострих відчуттів від купівлі новенького автомобіля швидше, ніж побачення вашого нового страхового рахунку. Стає дуже прикро, коли ти знаєш, що ти хороший водій, але вам доведеться платити стільки, скільки і тим кто були необережними в дорозі.

Porto Seguro, одна з найбільших бразильських страхових компаній, що займається автомобілями, повністю погоджується що неточності в прогнозах позовів про страхування автомобілів підвищують вартість страхування для хороших водіїв та знижують ціну для поганих.

У цьому змаганні вам буде запропоновано побудувати модель, яка передбачає ймовірність того, що водій подасть заявку на автострахування в наступному році. Хоча Porto Seguro впродовж останніх 20 років використовував машинне навчання, вони прагнуть дослідити нові, більш потужні методи. Точніший прогноз дозволить їм надалі пристосовувати свої ціни та, сподіваємось, зробить покриття автострахування більш доступним для більшої кількості водіїв.

Задача.

У цій задачі необхідно передбачите ймовірність того, що водій буде подавати страхове відшкодування.

Значення -1 вказують на відсутність ознаки під час спостереження (null). Цільові стовпці вказують, чи була подана претензія до цього страхувальника чи ні.

Метрика якості – Нормалізований Коефіцієнт Джині.

Розмір датасету – 110.49 мб.; 59 стовпчиків, 595212 рядків.

Оскільки дані анонімізовані то інтерпретація данних відсутня.

Задача 3. Задоволення пасажирів авіакомпанії (ориг. Airline Passenger Satisfaction) посилання

Опис.

Цей набір даних містить опитування про задоволеність пасажирів авіакомпанії. Серед факторів є такі як: стать, вік, відстань польоту та інші.

Задача.

Спрогнузувати чи буде клієнт задоволений чи ні

Метрика якості – площа під ROC кривою.

Розмір датасету – 11.63 мб.; 25 стовпчиків, 103904 рядків.

Інтерпритація перших 10 характеристик

- Стать: Стать пасажирів (Жінка, Чоловік)
- Тип клієнта: Тип клієнта (Лояльний клієнт, нелояльний клієнт)
- Вік: Фактичний вік пасажирів
- Тип подорожі: мета польоту пасажирів (особисті подорожі, ділові поїздки)
- Клас: Клас подорожей в літаку пасажирів (Бізнес, Еко, Еко Плюс)
- Відстань польоту: відстань польоту цієї подорожі
- Послуга Wi-Fi польоту: рівень задоволеності служби польоту wifi (0: не застосовується; 1-5)
- Час вильоту / прибуття зручний: Рівень задоволеності часом вильоту / прибуття зручний
- Легкість бронювання через Інтернет: рівень задоволення від бронювання через Інтернет
- Розташування воріт: Рівень задоволеності розташування воріт

Як бачимо з опису, слід очікувати як мінімум високу кореляція деяких характеристик з цільовою. Наприклад рівень задоволеності служби польоту wif

та рівень задоволення від бронювання через Інтернет очевидно будуть позитивно впливати на задоволеність вцілому.

Задача 4. Ідентифікація користувача по його поведінці в мережі інтернет (ориг. Catch Me If You Can ("Alice")) посилання

Опис.

будемо вирішувати таку задачу: по послідовності з декількох веб-сайтів, відвіданих поспіль однією і тією ж людиною, ми будемо ідентифікувати цю людину, а точніше знаходити ймовірність того що дана людина - цілова. Маємо задачу бінарної классифікації (необхідно классифікувати цільового користувача від інших). Ідея така: користувачі Інтернету по-різному переходять за посиланнями, і це може допомагати їх ідентифікувати (хтось спочатку в пошту, потім про футбол почитати, потім новини, потім нарешті - працювати, хтось - відразу працювати).

Задача.

Завдання - зробити прогнози для сесій, визначити, чи належать вони Еліс

Метрика якості – площа під ROC кривою.

Розмір датасету – 57.72 MB.; 22 стовпчиків, 82797 рядків.

Дані зібрані з проксі-серверів Університету Блеза Паскаля. "A Tool for Classification of Sequential Data", автори Giacomo Kahn, Yannick Loiseau і Olivier Raynaud.

У навчальній вибірці train_sessions.csv:

- Ознаки site_i - це індекси відвіданих сайтів

- Ознаки time_j - час відвідування сайтів site_j

- Цільовий ознака target - факт того, що сесія належить Еліс (тобто що саме Еліс ходила по всіх цих сайтів)

Перші 5 об’єктів нашої вибірки

session_id	site1	time1	site2	time2	site3	time3	...	time8	site9	time9	site10	time10	target
1	718	2014-02-20 10:02:45	NaN	NaT	NaN	NaT	...	NaT	NaN	NaT	NaN	NaT	0
2	890	2014-02-22 11:19:50	941.0	2014-02-22 11:19:50	3847.0	2014-02-22 11:19:51	...	2014-02-22 11:19:52	1516.0	2014-02-22 11:20:15	1518.0	2014-02-22 11:20:16	0
3	14769	2013-12-16 16:40:17	39.0	2013-12-16 16:40:18	14768.0	2013-12-16 16:40:19	...	2013-12-16 16:40:21	14768.0	2013-12-16 16:40:22	14768.0	2013-12-16 16:40:24	0
4	782	2014-03-28 10:52:12	782.0	2014-03-28 10:52:42	782.0	2014-03-28 10:53:12	...	2014-03-28 10:55:42	782.0	2014-03-28 10:56:12	782.0	2014-03-28 10:56:42	0
5	22	2014-02-28 10:53:05	177.0	2014-02-28 10:55:22	175.0	2014-02-28 10:55:22	...	2014-02-28 10:55:59	177.0	2014-02-28 10:57:06	178.0	2014-02-28 10:57:11	0

Методи машинного навчання

Серед актуальних на даний момент методів машиного навчання для порівняння були виділені наступні — як ті, які є достатньо ефективними і масово використовуються в застосуваннях:

- Логістична регресія
- Випадковий ліс
- Градієнтний Бустинг над деревами (реалізація - Light GBM)
- Градієнтний Бустинг над деревами (реалізація - XGBOOST)

Логістична регресія

Логістична регресія — це класика алгоритмів машинного навчання та найпопулярніший представник лінійних моделей [3] та [5]. Цю модель

застосовують у випадку, коли залежна змінна є категорійною, тобто може набувати тільки двох значень (чи, загальніше, скінченної множини значень).

Далі опишемо суть алгоритму

Нехай є деяка випадкова величина Y, що може набувати лише двох значень, які,як правило, позначаються цифрами 0 і 1. Нехай ця величина залежить від деякої множини пояснювальних змінних $x = (1, x_1, \dots, x_n)^T$. Залежність Y, від $x_1, \dots, x_n$. можна визначити ввівши додаткову змінну y*, де $y^* = \theta^T x = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n + \varepsilon$. Тоді:

$$Y = \begin{cases} 0, & y^* \leqslant 0 \\ 1, & y^* > 0 \end{cases}$$

При визначенні логістичної моделі стохастичний доданок ε вважається випадковою величиною з логістичним розподілом ймовірностей. Відповідно для певних конкретних значень змінних $x^* = x_1^*, \dots, x_n^*$ одержується відповідне значення y^* і ймовірність того, що $Y = 1$, така:

$$p(Y = 1) = p(y^* > 0) = p(\theta^T x^* + \varepsilon > 0) = p(\varepsilon > -\theta^T x^*) = p(\varepsilon \leqslant \theta^T x^*) = \Lambda(\theta^T x^*).$$

Передостання рівність випливає з симетричності логістичного розподілу, Λ позначає логістичну функцію — функцію розподілу логістичного розподілу:

$$\Lambda(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

Таким чином для конкретного значення x^i випадкова величина Y^i, має розподіл Бернуллі:$Y^i \sim B\left(1, \Lambda\left(\theta^T x^i\right)\right)$.

Логіт-модель задовольняє наступній умові:

$$ln\frac{p(1 \vee X)}{1 - p(1 \vee X)} = ln\frac{p(1 \vee X)}{p(0 \vee X)} = b_0 + b_1 x_1 + \ldots + b_J x_J$$

В якості реалізації логістичної регресії було використано пакет sklearn.linear_model.LogisticRegression з білбіотеки sklearn

Дерево ухвалення рішень

Хоча дерево ухвалення рішень [2] не буде безпосередньо використовуватися у цій роботі, воно є базовим алгоримом для більш складних алгоритмів таких як випадковий ліс та градієнтний бустинг. Тож важливо розуміти принцип побудови дерева та його застосування.

Спочатку введемо деякі важливі поняття.

Ентропія Шеннона визначається для системи з можливими N станами наступним чином:

$$S = -\sum_{i=1}^{N} p_i log_2 p_i,$$

де p_i - ймовірність знаходження системи в і-му стані

Оскільки ентропія - по суті степінь хаосу (або невизначеності) в системі, зменшення ентропії називають приростом інформації. Формально приріст інформації (information gain, IG) при розбитті вибірки за ознакою Q визначається як:

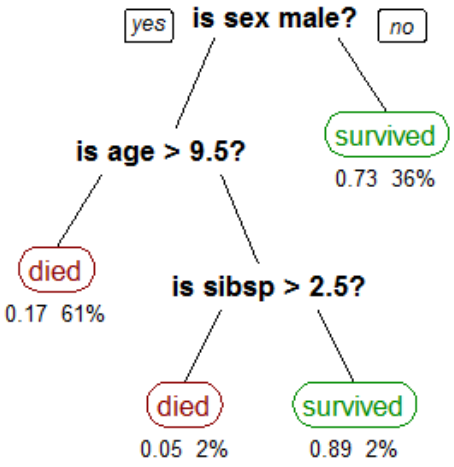
$$IG(Q) = S_O - \sum_{i=1}^{q} \frac{N_i}{N} S_i,$$

де -q число груп після розбиття,N_i - число елементів вибірки, у яких ознака Q має і-те значення. На практиці завжди q = 2.

Алгоритм побудови дерева простий. Кожен раз розбивається вибірка за ознакою так щоб приріст інформації був якнайбільшим. При кожному розбитті з

вибірки утворюється дві підвибірки. Процедура повторюється рекурсивно для кожної з вибірок поки не буде зустрітий критерій зупинки.

Приклад дерева ухвалення рішень



Основні гіперпараметри при побудові дерева:

- максимальна глибина дерева – зменшення глибини допомогає уникнути перенавчання (overfitting)
- максимальне число ознак, за якими шукається краще розбиття в дереві (це потрібно тому, що при великій кількості ознак буде "дорого" шукати краще (за критерієм типу приросту інформації) розбиття серед усіх ознак). Важливий параметр при побудові випадкового лісу, коли необхідно зменшити кореляцію між деревами.
- мінімальне число об'єктів в листі. У цього параметра є зрозуміла інтерпретація: скажімо, якщо він дорівнює 5, то дерево буде породжувати тільки ті класифікують правила, які правильні як мінімум для 5 об'єктів

варто зауважити що хоча використання вирішального дерева як самостійного алгоритму рідко є гарною ідеєю, як базовий алгоритм вирішальне дерево дуже потужне.

Випадковий ліс

Під випадковим лісом розуміють композицію деякої кількості вирішальних дерев [2].

У композиції дерева показуть кращий результат ніж поодинці. Спочатку розглянемо алгоритм побудови випадкового лісу а потім покажемо в чому його перевага

Алгоритм побудови випадкового лісу, що складається з N дерев, виглядає наступним чином:

Для кожного $n = 1, \dots, N$:

- Згенерувати вибірку X_n за допомогою бутстрепа (отримуємо нову вибірку з вхідної шляхом формування підвибірки з поверненням такого ж розміру) ;
- Побудувати вирішальне дерево b_n по вибірці X_n:
    - по заданому критерію ми вибираємо кращу ознаку, робимо розбиття в дереві по ньому до вичерпання вибірки
    - дерево будується, поки в кожному листі не більше $n_{\min}$ об'єктів або поки не досягнемо певної висоти дерева
    - при кожному розбитті спочатку вибирається m випадкових ознак з n вхідних, і оптимальний розподіл вибірки шукається тільки серед них.

Кінцевий классифікотор $a(x) = \frac{1}{N}\sum_{i=1}^{N} b_i(x)$ видає ймовірність приналежності об'єкту x до цільового класу. Рекомендується в задачах класифікації брати $m = \sqrt{n}$.

Щоб зрозуміти головну перевагу випадкового лісу над деревом ухвалення рішення необхідно розглянути **bias–variance tradeoff** [1] (Компроміс зсуву та дисперсії)

Припустимо, що в нас є тренувальний набір, який складається з набору точок $x_1, \dots, x_n$ та відповідних дійсних значень y_i. Ми виходимо з того, що існує функція з шумом$y = f(x) + \epsilon$, в якому шум ϵ має нульове середнє значення та дисперсію σ^2.

Нам треба знайти функцію $\hat{f}(x)$, що якомога краще наближує справжню функцію $f(x)$ засобами якогось алгоритму навчання. ми хочемо, щоби $(y - \hat{f}(x))^2$ було мінімальним, як для $x_1, \dots, x_n$, так і для точок за межами нашої вибірки.

Виявляється, що яку би функцію $\hat{f}$ ми не обрали, ми можемо розкласти математичне сподівання її похибки на небаченому зразкові x наступним чином:

$$\mathrm{E}[(y - \hat{f}(x))^2] \quad = \mathrm{Bias}[\hat{f}(x)]^2 + \mathrm{Var}[\hat{f}(x)] + \sigma^2$$

Доказ:

$$
\begin{aligned}
\mathrm{E}[(y-\hat{f})^2] &= \mathrm{E}[y^2+\hat{f}^2-2y\hat{f}] \\
&= \mathrm{E}[y^2]+\mathrm{E}[\hat{f}^2]-\mathrm{E}[2y\hat{f}] \\
&= \mathrm{Var}[y]+\mathrm{E}[y]^2+\mathrm{Var}[\hat{f}]+\mathrm{E}[\hat{f}]^2-2f\mathrm{E}[\hat{f}] \\
&= \mathrm{Var}[y]+\mathrm{Var}[\hat{f}]+(f-\mathrm{E}[\hat{f}])^2 \\
&= \mathrm{Var}[y]+\mathrm{Var}[\hat{f}]+\mathrm{E}[f-\hat{f}]^2 \\
&= \sigma^2+\mathrm{Var}[\hat{f}]+\mathrm{Bias}[\hat{f}]^2
\end{aligned}
$$

Де:

$$\mathrm{Bias}[\hat{f}(x)] = \mathrm{E}[\hat{f}(x)-f(x)]$$

$$\mathrm{Var}[\hat{f}(x)] = \mathrm{E}[\hat{f}(x)^2]-\mathrm{E}[\hat{f}(x)]^2$$

Основна перевага випадкового лісу на деревом ухвалення рішення полягає у зменшенні доданку $\mathrm{Var}[\hat{f}(x)]$ що відповідає за дисперсію. Справді, теоретичне зміщення в обох методах = 0, на похибку σ^2 ми впливати не можемо, а ость використання композиції дерев в випадковому лісі допомогає краще “генералізувати” алогоритм для роботи з новими даними, зменшуючи тим самим дисперсію алгоритма $\mathrm{Var}[\hat{f}(x)]$.

В алгоритмі будування випадкового лісу, дерева будуються таким чином зменшити кореляцію дерев між собою, робиться це саме для того щоб зменшити дисперсію алгоритму.

Запишемо дисперсію для випадкового лісу як

$$Varf(x) = \rho(x)\sigma^2(x)$$

Тут

- $\rho(x)$ - кореляція вибірки між будь-якими двома деревами, використовуваними при усередненні
- $\sigma^2(x)$ - це вибіркова дисперсія будь-якого довільно обраного дерева

Очевидно, що зменшення $\rho(x)$ призведе до зменшення $Varf(x)$ що і потрібно.

Градієнтний бустинг

Градієнтний бустинг як і вирішальний ліс являє собою композицію деякої кількості вирішальних дерев [2] та [6].

Для того щоб зрозуміти принцип роботи градієнтного бустинга почнемо з простого, а саме з Постановки задачі машинного навчання

Будемо вирішувати задачу відновлення функції в загальному контексті навчання з учителем. У нас буде набір пар ознак x і цільових змінних $y, \{(x_i, y_i)\}_{i=1,\dots,n}$, на якому будемо відновлювати залежність виду $y = f(x)$. Відновлювати будемо наближенням $\hat{f}(x)$, а для розуміння, яке наближення краще, у нас також буде функція втрат $L(y, f)$, яку ми будемо мінімізувати:

$$y \approx \hat{f}(x), \hat{f}(x) = \arg\min_{f(x)} L(y, f(x))$$

основна ідея бустингу полягає у тому щоб шукати наближення $\hat{f}(x)$, у вигляді:

$$\hat{f}(x) = \sum_{i=0}^{M} \hat{f}_i(x)$$

Щоб розв'язати задачу, необхідно обмежити пошук сімейством функцій $\hat{f}(x) = h(x, \theta)$. Відразу врахуємо, на кожному кроці для функцій нам знадобиться

підбирати оптимальний коефіцієнт $\rho \in \mathbb{R}$. Для кроку t завдання виглядає наступним чином:

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

$$(\rho_t, \theta_t) = \arg\min_{\rho,\theta} \mathbb{E}_{x,y}\left[L\left(y, \hat{f}(x) + \rho \cdot h(x,\theta)\right)\right],$$

$$\hat{f}_t(x) = \rho_t \cdot h(x, \theta_t)$$

Задача поставлена, аналітично розв'язати її неможлово. Запишемо ітераційний метод розв'язку цієї задачі. Для кроку t задача виглядає наступним чином:

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x),$$

$$r_{it} = -\left[\frac{\partial L\big(y_i, f(x_i)\big)}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)} \text{,for } i = 1, \dots, n,$$

$$\theta_t = \arg\min_{\theta} \sum_{i=1}^{n} (r_{it} - h(x_i, \theta))^2,$$

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta_t))$$

Алгоритм градієнтного бустингу

На вхід алгоритму потрібно зібрати кілька складових:

- Набір даних
- Число ітерацій M

- Вибрати дифернційовану функцію втрат $L(y, f)$
- Вибір сімейства функцій базових алгоритмів $h(x, \theta)$, з процедурою їх навчання. Зазвичай вибирають дерева ухвалення рішень (їх обрав і я)
- Гіпарпараметри базових алгоритмів $h(x, \theta)$

Власне алгоритм:

- Ініціалізувати GBM константним значенням $\hat{f}(x) = \hat{f}_0, \hat{f}_0 = \gamma, \gamma \in \mathbb{R}$

$$\hat{f}_0 = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

- Для кожної ітерації $t = 1, \dots, M$ повторювати:
  - Порахувати псевдо-залишки r_t

    $r_{it} = -[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}]_{f(x)=\hat{f}(x)}$,for $i = 1, \dots, n$
  - Побудувати новий базовий алгоритм $h_t(x)$ як регресію на псевдо-залишках

    $\{(x_i, r_{it})\}_{i=1,\dots,n}$
  - Знайти оптимальний коефіцієнт ρ_t при $h_t(x)$ відносно початкової функції втрат

    $\rho_t = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta))$
  - Зберегти $\hat{f}_t(x) = \rho_t \cdot h_t(x)$
  - Оновити поточно наближення $\hat{f}(x)$

    $\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}_t(x) = \sum_{i=0}^{t} \hat{f}_i(x)$
- Скомпонувати підсумкову GBM модель $\hat{f}(x)$

$$\hat{f}(x) = \sum_{i=0}^{M} \hat{f}_i(x)$$

Оскільки в цій роботі ми будемо використовувати дві реалізації градієнтного бустинга, та варто записати відмінності реалізацій.

Різниця між XGB and LGBM.

Релалізація градієнтного бустинга LGBM вишла пізнише ніж XGBoost.

Розробники LGBM стверджують що їхня реалізація навчається на данних значно швидше ніж XGBoost, а по іншим показникам приблизно така сама.

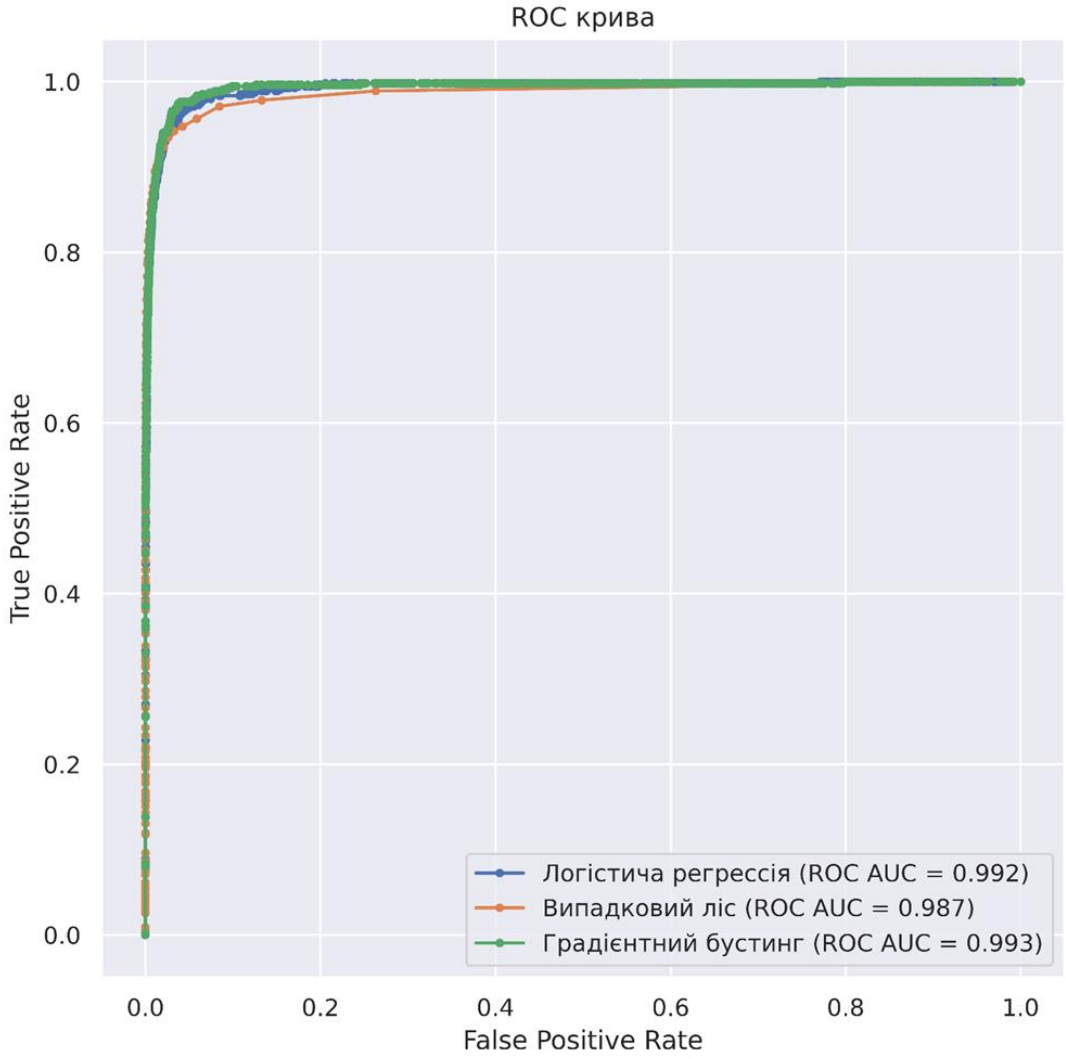
Власне у цій роботі ми визначими на практичних задачах різницю цих двох реалізацій.

Метрики якості

ROC AUC

Для бінарної класифікації дуже часто використовується ROC-крива [5] (Receiver Operator Characteristic), яка демонструє залежність кількості правильно класифікованих зразків від кількості невірно класифікованих зразків. Перша група висновків називаються хибнопозитивними, а друга — хибнонегативними. Вважається, що існує параметр, за допомогою якого можно отримати розбиття на два класи. Цей параметр називається точкою відсікання (cut-off value). Від цієї точки залежить величини помилок 1-го і 2-го роду.

Приклад з задачі про ідентифікацію користувача



За моделлю	**Насправді**	
	Так	**Ні**
Так	TP	FR
Ні	FN	TN

Характеристикою якості класифікації вважаються такі показники:

- Доля істинно-позитивних зразків (True Positives Rate):

  $TRP = \frac{TP}{TP+FN}$. 100%

- Доля істинно-позитивних зразків (False Positives Rate):

$$FRP = \frac{FP}{TN+FP}.100\%$$

AUC (area under curve). – означає що якість классифікації вимірюємо як площадь під кривою.

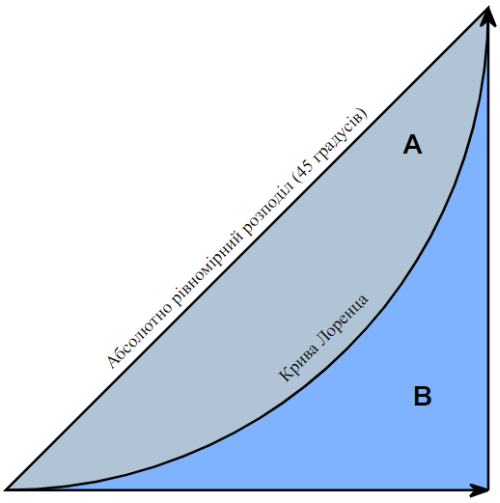
Normalized GINI Coefficient

Коефіцієнт Джині - показник нерівності розподілу деякої величини чисел [2], що приймає значення між 0 і 0.5, де 0 означає абсолютну рівність (величина приймає лише одне значення), а 0.5 позначає повну нерівність.

Обчислити можна за формулою

$$G = \frac{2\Sigma_{i=1}^{n} i y_i}{n\Sigma_{i=1}^{n} y_i} - \frac{n+1}{n}$$

Графічна інтерпритація – площа сірої фігури (A)



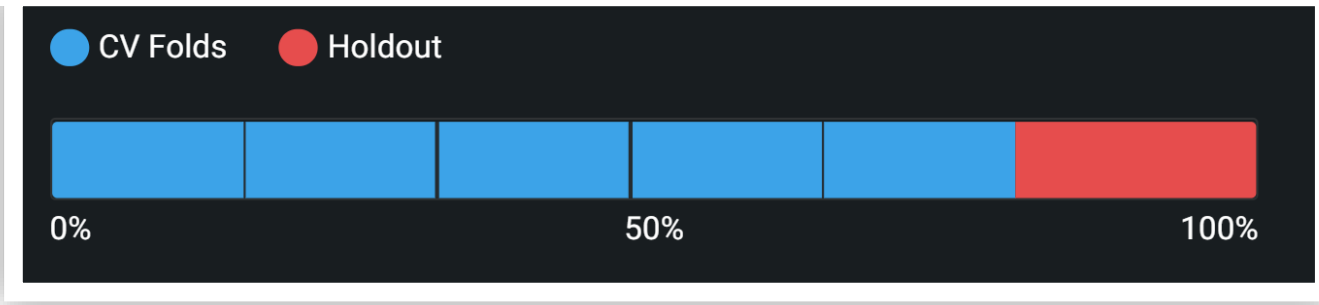
Результати обчислень

Задача 1. Задоволення клієнтів Сантандера (ориг. Santander Customer Satisfaction) посилання

Схема навчання моделей.

алгоритм:

- Ділимо дані у співвідношенні 4 до 1. (0.8 данних використаємо для знаходження оптимальної моделі, решту відкладемо для остаточної валідації (по цих 0.2 даних і виміряємо “якість” алгоритму за відповідною метрикою.
- 0.8 данних розіб’ємо на 5 кошиків і проведемо стратифіковону (зберігається пропорція цільового классу в кошиках) кросс валідацію. На єтапі кросс-валідації знаходимо оптимальні значення гіперпараметрів моделі порівнюючи якість классифікації за відповідною метрикою.
- Далі модель з найкращими значеннями гіперпараметрів навчаємо на всій навчальній вибірці та знаходимо якість классифікації на відкладених 0.2 данних за вибраною метрикою. Таким чином отримали значення метрики якості классифікації.
- Маючи готову модель, обраховуємо дві інші метрики – швидкість классифікації та розмір моделі.



(поділ даних на частини. Holdout – 0.2 відкладених даних. CV Folds – “кошики” для кросс-валідації)

Після виконання алгоритму отримали такі результати для кожної з моделей

	Лог. Рег	Вип. Ліс	XGBoost	Light GBM
Якість к-ції	0.8360	0.8456	0.8429	0.8406
Швидкість к-ції*	0.1296s	0.2373s	0.0909 сек	0.1473s
Розмір моделі	1.181 MB	18.144 MB	3.718 MB	1.728 MB

*необхідний час для классифікації 10_000 об'єктів.

Як бачимо якість классифікації у всіх моделей приблизно однокова та на доволі високому рівні.

Тепер розглянемо два інших показника. Як бачимо всі моделі, крім випадкового лісу перебувають приблизно на одному рівні. Випадковий ліс програє як і за швидкістю классифікації так і за розміром самої моделі (причому доволі суттєво) іншим моделям.

Задача 2. Чи буде водій подавати страхове відшкодування? (ориг. Porto Seguro’s Safe Driver Prediction) посилання

Схема навчання моделей ідентична до попередньої задачі.

Після виконання алгоритму отримали такі результати для кожної з моделей

	Лог. Рег	Вип. Ліс	XGBoost	Light GBM
Якість к-ції	0.2503	0.2533	0.2707	0.2676
Швидкість к-ції*	0.0270s	0.0423s	0.0439s	0.0350s
Розмір моделі	8.752 MB	16.652 MB	17.639 MB	10.296 MB

*необхідний час для классифікації 10_000 об'єктів.

Як бачимо бустингові алгоритми показали кращі помітно кращі результати классифікації.

Також помітно що логістична регрессія найкраща за швидкодією та за розміром моделі.

Також важливо зазначити деяку тенденцію. Зазвичай модель з кращою якістю классифікації має гірші показники по двом іншим метрикам.

Задача 3. Задоволення пасажирів авіакомпанії (ориг. Airline Passenger Satisfaction) посилання

Схема навчання моделей ідентична до першої задачі.

Після виконання алгоритму отримали такі результати для кожної з моделей

	Лог. Рег	Вип. Ліс	XGBoost	Light GBM
Якість к-ції	0.9645	0.9941	0.9952	0.9950
Швидкість к-ції*	0.0707s	0.1295s	0.0420s	0.1688s
Розмір моделі	1.546 MB	60.397 MB	50.758 MB	20.06 MB

*необхідний час для классифікації 1_000 об'єктів.

Як бачимо логістична регресія суттєво програє за якістю классифікації іншим моделям. Скоріше за все в данних є складна нелінійна залежність між цільовою змінною та іншими характеристиками, з якою краще працюють моделі де в якості базового алгоритму взяте вирішальне дерево.

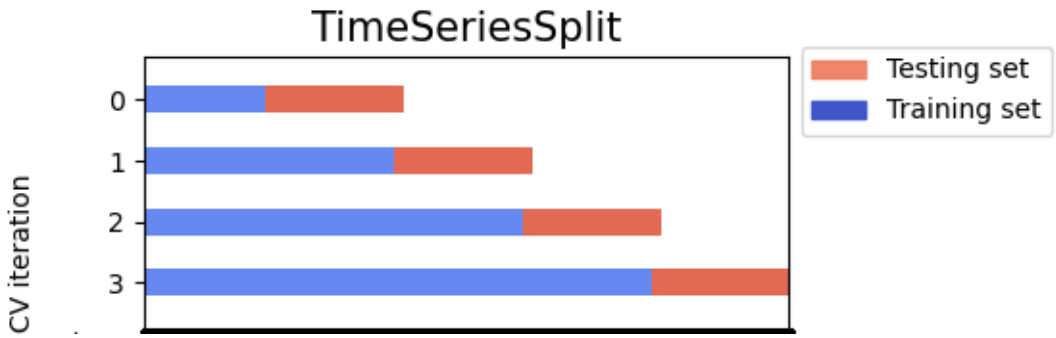
Також бачимо що логістична регрессія має тренд займати лідуючі позиції за швидкістю к-ції та розміром моделі.

Задача 4. Ідентифікація користувача по його поведінці в мережі інтернет (ориг. Catch Me If You Can ("Alice")) посилання

У тому вигляді в якому нам дані дані неможливо навчати алгоритми машинного навчання. Необхідно їх підготувати для алгоритмів.

Після перетворення отримали нову кількість стовпчиків – 48371 (у розрідженому форматі). Що значно більше ніж в попередніх задачах.

Схема кросс-валідації дещо інша і виглядає так



На першому прогоні – навчаємо на [0, 0.2] даних, а тестуємо на [0.2, 0.4]

На останньому – навчаємо на [0, 0.8] даних, а тестуємо на [0.8, 1].

Після виконання алгоритму отримали такі результати для кожної з моделей

	Лог. Рег	Вип. Ліс	XGBoost	Light GBM
Якість к-ції	0.9920	0.9887	0.9965	0.9967
Швидкість к-ції* **	0.0145 s	2.02 s	2.86 s	2.52 s
Розмір моделі	3.885 MB	123.934 MB	14.915MB	12.344

*необхідний час для классифікації 21_000 об'єктів.

** час необхідний для трансформації даних не врахований

Бачимо що логістична регресія на порядки швидше классифікує ніж інші моделі. Бо сучасні реалізації логістичної регрессії дуже добре працюють за розрідженими даними. По розмірам моделі логістична регрессія лідує також.

По якості классифікації всі моделі приблизно на одному рівні.

Висновки

Побудуємо результуючу таблицю по нашим методам . Будувати будемо таким чином. Спочатку для кожної задачі для кожної метрики нормуємо показники (ділимо значення метрики для кожного з методів на максимальний показник, отримуємо нові значення у діапазоні [0, 1]. Далі знаходимо середнє арифметичне по всім задачам. Результат має вигляд:

	Лог. Рег	Вип. Ліс	XGBoost	Light GBM
Якість к-ції	0.970221	0.981646	0.999125	0.995604
Швидкість к-ції	0.3962713	0.85925575	0.657967	0.82477945
Розмір моделі	0.15425	0.986	0.54075	0.27725

Далі сформуємо висновки по кожній з моделей.

Логістична регресія

В цілому про логістичну регресію можна сказати що вона має найнижчий показник за якістю класифікації але має найвищі показники за двома іншими метриками.

Час, необхідний для навчання моделі логістичної регресії, найменший із всіх моделей. В цілому по комп’ютерним ресурсам логістична регресія є найбільш вигідною серед всіх моделей.

Також слід зазначити, що в обраній реалізації логістична регресія має найменшу кількість гіперпараметрів для тюнингу моделі. Що спрощує процес тюнингу.

переваги:

- добре вивчені (тобто результа роботи моделі можна легко пояснити, а не інтерпретувати як “чорний ящик”)
- Дуже швидкі, можуть працювати на дуже великих вибірках
- Практично поза конкуренцією, коли ознак дуже багато (від сотень тисяч і більше), і вони розріджені (в задачі 4 це явно показано, логістична регресія значно обходить інші методи по швидкості в такій ситуації)
- Коефіцієнти перед ознаками можуть інтерпретуватися (за умови що ознаки масштабовані)
- Логістична регресія видає ймовірності віднесення до різних класів
- Модель може будувати і нелінійний кордон, якщо на вхід подати поліноміальні ознаки

недоліки:

- Погано працюють в задачах, в яких залежність відповідей від ознак складна, нелінійна
- в більшості випадків програє за якістю класифікації моделям які в якості базового алгоритму використовуються дерева ухвалення рішень

Компроміс:

- доволі часто можна суттєво покращити якість классифікації за допомогою генерування нових ознак (англ. Feature engineering). Однак на це необхідні людські ресурси

Випадковий ліс.

Порівнювати випадковий ліс будемо перш за все з іншими алгоритмами що використовують дерева ухвалення рішень.

Як бачимо по результатам випадковий ліс програє бустинговим алгоритмам по всім метрикам.

Одна з переваг випадкового лісу над іншими алгоритмами що використовують дерева ухвалення рішень – менша схильність до перенавчання (англ. Overfitting) що є наслідком переваги беггінга над бустингом.

Переваги:

- має високу точність передбачення, на більшості завдань буде краще лінійних алгоритмів.
- практично не чутливий до викидів в даних через випадкове семплювання.
- Маштабування: дерева можна будувати паралельно на декількох машинах
- не чутливий до масштабування (і взагалі до будь-яких монотонних перетворень) значень ознак.
- однаково добре обробляє як безперервні, так і дискретні ознаки.
- рідко перенавчається, на практиці додавання дерев майже завжди тільки покращує композицію, але на валідації, після досягнення певної кількості дерев, крива навчання виходить на асимптоту.
- добре працює з пропущеними даними; зберігає хорошу точність, якщо більша частина даних пропущена.

Недоліки:

- на відміну від одного дерева, результати випадкового лісу складніше інтерпретувати.
- немає формальних висновків (p-values), доступних для оцінки важливості змінних.
- алгоритм схильний до перенавчання на деяких завданнях, особливо на зашумлених даних
- великий розмір вихідних моделей. Потрібно $O(NK)$ пам'яті для зберігання моделі, де K - число дерев.
- Погано працюють з розрідженими даними

Градієнтний бустинг.

Як бачимо бустингові алгоритми отримали найкащі показники за якістю классифікації та середні показники за іншими метриками.

Спочатку запишемо загальну характеристику градієнтного бустинга, а потім порівняємо дві різні реалізації(XGBoost та LightGBM).

Переваги:

- Демонструють найкращі показники якості классифікації серед классичних алгоритмів машинного навчання.
- Добре працюють “із коробки” (показують гарний результат і без тюнингу параметрів)
- добре працює з пропущеними даними; зберігає хорошу точність, якщо більша частина даних пропущена.
- не чутливий до масштабування (і взагалі до будь-яких монотонних перетворень) значень ознак.

Недоліки:

- Маштабованість – оскільки побудова кожного наступного базового класифікатора потребує наявністі попередніх класифікаторів то виникають труднощі при маштабуванні.
- чутливий до викидів у даних: оскільки кожен класифікатор зобов'язаний виправляти помилки попередників. Таким чином, метод занадто залежить від викидів у даних.

Тепер порівняємо безпосередньо дві використані реалізації градієнтного бустинга - (XGBoost та LightGBM)

По якості классифікації – обидві реалізації показали дуже хороші результати, але, все-таки XGBoost дещо попереду і необхідно це відзначити.

По швидкодії – тут також лідує XGBoost, та має відрив на 25-30 відсотків.

По розміру моделі – тут явно лідує LightGBM та має достатньо велику перевагу у два рази.

По швидкості навчання тєж лідує LightGBM

Загальний висновок

В роботі розглянуті алгоритми, які є часто вживаними класичними алгоритмами машинного навчання для вирішення задачі класифікації по табличним даним, хоча вони і не накривають весь величезний клас таких алгоритмів. У цій роботі я не розглядав алгоритми на базі нейронних мереж, які також ефективно розв’язують задачі класифікації — зокрема тому, що саме

дослідженням ефективності нейронних мереж різного вигляду останнім часом присвячена величезна кількість робіт. Окрім того, під табличними я маю на увазі дані, де кожний запис – це деяка характеристика об’єкта яка має певну інтерпретацію, а для таких даних алгоритми із класу нейронних мереж часто програють класичним (англ. shallow machine learning).

Алгоритми реалізовано та продемонстровано результати порівняння за відповідними метриками на наборі модельних задач. Показано, що кожен алгоритм має свої переваги та недоліки, і вибір “оптимального” залежить від даних та від поставлених задач.

Посилання

1. Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning. - MIT Press, 2016. P 96 – 152.
2. Hastie, Trevor, Trevor Hastie, Robert Tibshirani, and J H. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York: Springer, P 337 – 380, P 587 – 601.
3. Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006, P 205 – 207.
4. Воронцов К.В. Математические методы обучения по прецедентам. – 2007. – 141 с. [Електронний ресурс]. – URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf
5. Клюшин Д. А. Логістична регресія. – 6 с. [Електронний ресурс]. – URL: http://om.univ.kiev.ua/users_upload/15/upload/file/pr_lecture_08.pdf
6. Открытый курс машинного обучения. Тема 10. Градиентный бустинг [Електронний ресурс]. – URL: https://habr.com/ru/company/ods/blog/327250/
7. Breiman L. Bagging Predictors / L. Breiman // Machine Learning. – 1996. – P. 123–140.
8. Вандерплас Д. Python для сложных задач: наука о данных и машинное обучение. – СПб.: Питер, 2018. – 576 с.