СЕКЦІЯ «ІНФОРМАТИКА»

УДК 004.85:519.6

DOI 10.31651/2076-5886-2021-1-58-68

PACS 02.70.Wz, 07.05.Kf, 07.05.Mh, 07.05.Tp

КОНОНЕНКО Вероніка Вікторівна,
студентка спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького

КРАСНОШЛИК Наталія Олександрівна,
кандидат технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: wlik007@ukr.net
ORCID 0000-0003-4661-6997

ВИКОРИСТАННЯ МЕТОДІВ БУСТІНГУ ДЛЯ ЗАДАЧ МАШИННОГО НАВЧАННЯ

У роботі проведено порівняння методів бустінгу при розв'язанні задач машинного навчання. Розглянуто бустінг над деревами рішень, який являє собою ансамблевий алгоритм і залишається одним з найбільш ефективних і популярних методів машинного навчання. Описано ідею методів градієнтного і адаптивного бустінгу та бібліотек XGBoost і CatBoost. Використано методи градієнтного бустінгу, AdaBoost, XGBoost і CatBoost у середовищі Jupyter Notebook для розв'язання задач класифікації та регресії. Проведено порівняння якості отримуваних прогнозів та часу навчання алгоритмів при розв'язанні задачі медичної діагностики, задачі кредитного скорінгу, задачі прогнозування кількості оренди велосипедів. Встановлено, що найменший час навчання і найкращі результати для задач бінарної класифікації демонструє модель XGBoost. Для задачі регресії кращі результати продемонструвала модель CatBoost, але при цьому час її навчання завжди на порядок більший у порівняні з іншими моделями.

Ключові слова: *машинне навчання, бустінг, градієнтний бустінг, XGBoost, CatBoost.*

Вступ

Машинне навчання є підрозділом досить великої області науки, що вивчає штучний інтелект. Алгоритми, що відносяться до даного напрямку, використовуються при розв'язанні задач, для яких часто складно або неможливо знайти явний алгоритм розв'язання, наприклад: медична діагностика, створення алгоритмів фільтрації реклами і спаму, прогноз погоди, прогнозування економічних і соціальних процесів, детектування об'єктів на фото або відео, розпізнавання тексту, мови та ін..

Серед сучасних методів машинного навчання одними з досить ефективних є методи, які використовують ансамблі алгоритмів. У процесі роботи алгоритми цього класу генерують відразу кілька прогнозів за допомогою кожного окремого алгоритму, що входить в ансамбль. Підсумковий розв'язок задачі одержують шляхом інтеграції набору отриманих розв'язків певним методом, наприклад, методом голосування.

До часто використовуваних ансамблів алгоритмів відносять бустінг, що

обумовлено його досить активним розвитком і високою якістю одержуваних результатів. Бустінг (англ. boosting – покращення) – це процедура послідовного побудови ансамблю алгоритмів машинного навчання, коли кожен наступний алгоритм прагне компенсувати недоліки композиції всіх попередніх алгоритмів.

Мета роботи полягає у застосуванні і порівнянні різних методів бустінгу при розв'язанні задач машинного навчання.

Виклад основного матеріалу

1. Методи бустінгу

Ансамбль алгоритмів – це метод, який використовує кілька алгоритмів з метою отримання кращої ефективності прогнозування, ніж можна було б отримати від кожного алгоритму машинного навчання окремо. В основі таких методів лежить ідея навчання декількох (базових) класифікаторів на одній і тій самій навчальній вибірці та комбінації їх прогнозів для нових тестових даних. Математичним обґрунтування цієї ідеї є теорема Кондорсе про журі присяжних (the Condorcet's jury theorem), що датується XVIII століттям [1-2].

Таким чином, маючи кілька слабких класифікаторів, можна об'єднати їх прогнози і досягти більш високої точності класифікації об'єктів з тестової вибірки. Серед найбільш відомих ансамблевих методів класифікації виділяють: беггінг (bagging), бустінг (boosting), стекінг (stacking).

Бустінг – метод побудови ансамблю алгоритмів, при якому базові алгоритми навчаються послідовно і кожен наступний алгоритм ансамблю застосовується до результатів на виході попереднього.

Таким чином, метод бустінга реалізує послідовну композицію алгоритмів навчання, в якій кожен алгоритм повинен компенсувати помилки, допущені композицією попередніх алгоритмів (див. рис. 1).

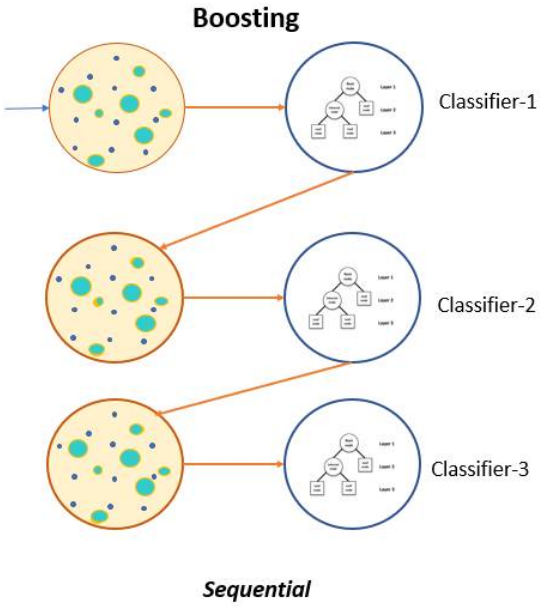

Рис. 1 Ідея методу бустінгу [4]

У основі ідеї бустінга лежить питання: чи може набір «слабких» навчальних алгоритмів дати у підсумку «сильний» алгоритм? Різні алгоритми навчання помиляються на різних прикладах. Тоді перший алгоритм в композиції буде навчатися

на всьому наборі, другий – на тих прикладах, на яких перший припустився помилки, третій – на тих прикладах, де помилилися перший і другий алгоритми, і т. д. При цьому кожному об'єкту вибірки, залежно від величини допущеної на ньому помилки, присвоюється вага, яка впливає на ймовірність вибору об'єкту для навчання наступного алгоритму композиції [3].

На сьогоднішній день бустінг залишається одним з найбільш популярних і широковживаних методів машинного навчання. Основні причини це – простота, універсальність, гнучкість (можливість побудови різних модифікацій), і висока узагальнююча здатність. Головну роль у популяризації бустінгу зіграли різні змагання з машинного навчання, особливо які проводяться на платформі kaggle.com.

Бустінг над деревами рішень вважається одним з найбільш ефективних методів з точки зору якості класифікації. При його застосуванні можна спостерігати сутеве зменшення частоти помилок на тестової вибірці по мірі нарощування композиції. На прикладі бустінга стало зрозуміло, що досить складні композиції алгоритмів демонструють гарну якість, якщо їх правильно налаштовувати. Згодом феномен бустінга отримав теоретичне обґрунтування. Виявилося, що зважене голосування не збільшує ефективну складність алгоритму, а лише згладжує відповіді базових алгоритмів. Кількісні оцінки узагальнюючої здатності бустінга формулюються в термінах відступу. Ефективність бустінга пояснюється тим, що в міру додавання базових алгоритмів збільшуються відступи навчальних об'єктів. Причому бустінг продовжує розмежовувати класи навіть після досягнення безпомилкової класифікації навчальної вибірки [1].

1.1 Адаптивний алгоритм бустінгу AdaBoost

Алгоритм AdaBoost (від Adaptive Boosting) – алгоритм машинного навчання, запропонований Йоав Фройнд (Yoav Freund) і Робертом Шапіро (Robert Schapire). Це мета-алгоритмом, який в процесі навчання будує композицію з базових алгоритмів для покращення їх ефективності. AdaBoost є алгоритмом адаптивного бустінгу в тому розумінні, що кожен наступний класифікатор будується для об'єктів, які погано класифікуються попередніми класифікаторами.

AdaBoost викликає слабкий класифікатор у циклі. Після відповідного виклику оновлюються значення вагів, які визначають важливість тих чи інших об'єктів з навчальної вибірки для класифікації. На кожній ітерації ваги кожного невірно класифікованого об'єкта зростають, таким чином новий класифікатор «фокусує свою увагу» на цих об'єктах [5].

Опишемо базовий алгоритм для задачі побудови бінарного класифікатора. Розглядаємо задачу класифікації на два класи $Y = \{-1, +1\}$. Припустимо, що базові алгоритми $b_1, ..., b_T$ також повертають лише дві відповіді -1 і +1. Вектор вагів об'єктів $\mathrm{W}^l = (\mathrm{w}_1, ..., \mathrm{w}_2)$, навчальна вибірка X^l. Нехай

$$Q(b, W^l) = \sum_{i=1}^{l} w_i \left[y_i \cdot b(x_i) < 0 \right]$$

– стандартний функціонал якості алгоритму класифікації b. Апроксимуємо порогову функцію втрат $[z < 0]$ за допомогою експоненти $E(z) = exp(-z)$.

Кроки алгоритму AdaBoost будуть наступні [5]:

1. Ініціалізація ваги об'єктів: $w_i = 1/i, i = 1, ..., l$;

2. Для всіх $t = 1, ..., T$, поки не виконаний критерій зупинки.

2.1 Знаходимо класифікатор $b_t : X \to \{-1,+1\}$ який мінімізує зважену похибку класифікації

$$b_t = \arg\min_b Q(b, W^l)$$

2.2 Перераховуємо коефіцієнти зваженого голосування для алгоритму класифікації b_t:

$$\alpha_t = \frac{1}{2}\ln\frac{1-Q(b,W^l)}{Q(b,W^l)}$$

2.3 Перерахунок вагів об'єктів: $w_i = w_i \cdot \exp(-\alpha_t \cdot y_i \cdot b_t(x_i)), i = 1,...,l$.

2.4 Нормування вагів об'єктів: $w_0 = \sum_{j=1}^{l} w_j$, $w_i = w_i / w_0$, $i = 1,...,l$.

3. Повертаємо: $a(x) = sign\left(\sum_{i=1}^{T} \alpha_i \cdot b_i(x)\right)$.

Така техніка послідовного навчання нагадує градієнтний спуск, лише замість зміни параметрів одного алгоритму для мінімізації функції втрат, AdaBoost додає моделі в ансамбль, поступово покращуючи його.

1.2 Градієнтний бустінг

Іншим дуже популярним бустінговим алгоритмом, принцип роботи якого дуже схожий на розглянутий AdaBoost, є градієнтний бустінг. Даний алгоритм працює послідовно додаючи до минулих моделей нові так, щоб виправлялися помилки, допущені попередніми предикторами. Метод градієнтного бустінгу відрізняється від адаптивного тим, що, на відміну від AdaBoost, який змінює ваги на кожній ітерації, градієнтний намагається навчати нові моделі за залишковою похибкою попередніх моделей (рухаючись до мінімуму функції втрат).

Градієнтний бустінг починає роботу з побудови початкової моделі і коригує її, крок за кроком створюючи послідовність моделей. Кожна модель у послідовності створюється на основі залишків моделі, яка отримується на попередньому кроці. Залишки моделі по суті використовуються у якості цільової змінної. Тобто розв'язується задача [6]:

$$F = \sum_{i=1}^{m} L\left(y_i, a(x_i) + b_i\right) \to \min \qquad (1)$$

де $a(x_i)$ – початково побудований алгоритм, b_i – наступний алгоритм, який будується і здійснює коригування відповідей $a(x_i)$ до вірних. Таким чином, поліпшується функціонал $\varepsilon_i = y_i - a(x_i)$.

Вираз (1) слід розглядати як мінімізацію функції $F\left(b_1,...,b_m\right)$, отже, необхідно вибрати ефективний метод її мінімізації. У даному випадку функція від багатьох змінних максимально зменшується в напрямку свого антиградієнта:

$$-\left(L'\left(y_1, a(x_1)\right),...,L'\left(y_m, a(x_m)\right)\right) .$$

Отже, відповіді алгоритму b_i можуть бути визначені наступним чином:

$$b_i = -L'\left(y_i, a(x_i)\right), i = 1,...,m .$$

Налаштування алгоритму відбувається на навчальній вибірці:

$$\left(x_i, -L'\left(y_i, a(x_i)\right)\right)_{i=1}^{m} .$$

Звідси визначається і назва методу – градієнтний бустінг. Далі, після побудови відповідей алгоритму b_i, будується третій алгоритм, який коригує зазначену суму. А далі процес продовжується. Це приводить до одного з основних параметрів градієнтного бустінгу – кількості ітерацій бустінгу N. Параметр впливає на точність одержуваних результатів.

Наступним важливим параметром є η або швидкість навчання (learning rate). Суть його полягає у тому, що зміщення аргументу в F відбувається тільки на частину вектору для того щоб зберегти баланс між точністю і швидкістю збіжності алгоритму:

$$a_{t+1}(x) = a_t(x) + \eta \cdot b_t \ ,$$

де даний параметр визначається у межах $\eta \in (0,1]$.

Наразі вважається, що градієнтний бустінг над деревами рішень є один з найбільш універсальних і сильних методів машинного навчання, відомих на сьогоднішній день.

1.3 XGBoost і CatBoost

Екстремальний градієнтний бустінг (XGBoost – eXtreme Gradient Boosting) – це удосконалена реалізація градієнтного бустінгу. Цей алгоритм має високу прогнозну здатність і в десять разів швидше будь-яких інших методів градієнтного бустінгу. Крім того, включає в себе різні регуляризації, що зменшують перенавчання і покращують загальну продуктивність. Тобто, це програмна бібліотека з відкритим кодом, яка використовує системну оптимізацію та удосконалення алгоритму (див. рис. 2).

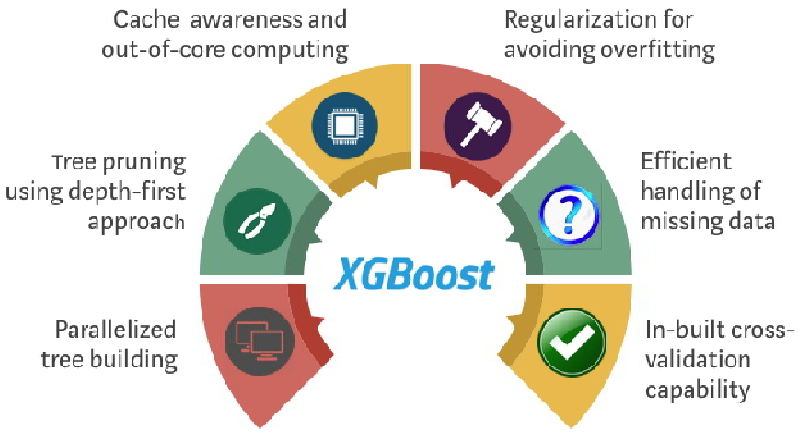

Рис. 2 Як XGBoost оптимізує стандартний градієнтний бустінг [7]

У XGBoost використовується системна оптимізація та покращення алгоритму [7]. Системна оптимізація передбачає наступне:

- *Паралелізація*. У XGBoost побудова дерев ґрунтується на паралелізації. Для покращення часу роботи порядок циклів змінюється: ініціалізація проходить при зчитуванні даних, потім виконується сортування, яке використовує паралельні потоки.
- *Відсікання гілок дерева*. У градієнтному бустінгу критерій зупинки для розбиття дерева залежить від критерію від'ємної втрати у точці розбиття. XGBoost використовує параметр максимальної глибини `max_depth` замість цього критерію і починає зворотне відсікання.

– *Апаратна оптимізація*. Алгоритм був розроблений таким чином, щоб він оптимально використовував апаратні ресурси. Це досягається шляхом створення внутрішніх буферів у кожному потоці для зберігання статистики градієнта.

А покращення алгоритму полягають у наступному:

– *Регуляризація*. Алгоритм штрафує складні моделі, використовуючи як регуляризацію Lasso (L1), так і Ridge-регуляризацію (L2), для того, щоб уникнути перенавчання.
– *Робота з розрідженими даними*. Алгоритм спрощує роботу з розрідженими даними, у процесі навчання заповнюючи пропущені значення в залежності від значення втрат.
– *Метод зважених квантилів.* XGBoost використовує його для того, щоб найбільш ефективно знаходити оптимальні точки поділу у разі роботи зі зваженим набором даних.
– *Крос-валідація*. Алгоритм використовує свій власний метод крос-валідації на кожній ітерації. Тобто, не потрібно окремо програмувати цей пошук і визначати кількість ітерацій бустінгу для кожного запуску.

Ще одним прикладом методу бустінгу є CatBoost.

CatBoost – відкрита програмна бібліотека, розроблена компанією Яндекс, яка реалізує унікальний запатентований алгоритм побудови моделей машинного навчання, що використовує одну з оригінальних схем градієнтного бустінгу. Основне API для роботи з бібліотекою реалізовано для мови Python, також існує реалізація для мови програмування R.

У липні 2017 року компанія Яндекс виклала бібліотеку з алгоритмом CatBoost у вільний доступ з відкритою ліцензією Apache 2.0.

Переваги використання CatBoost наступні [8]:

– CatBoost дозволяє проводити навчання на декількох GPU.

– Бібліотека дозволяє отримати відмінні результати з параметрами за замовчуванням, що скорочує час, необхідний для налаштування гіперпараметрів.

– Забезпечує підвищену точність за рахунок зменшення перенавчання. Навчені моделі CatBoost можна експортувати у Core ML для виведення на пристроях (iOS).

– Реалізовано можливість обробляти пропущені значення.

– Може використовуватися для задач регресії та класифікації.

2. Приклади задач машинного навчання

У якості прикладів задачі бінарної класифікації розглянемо задачу прогнозування у пацієнта хвороби Паркінсона і задачу кредитного скорінгу.

Для прогнозування у пацієнта хвороби Паркінсона скористаємось набором даних «Parkinsons Data Set» [9]. Цей датасет складається з ряду біомедичних вимірювань голосу від 31 людини, 23 з яких мають хворобу Паркінсона. Кожен стовпець у таблиці є певним виміром голосу, а кожен рядок відповідає одному із 195 записів голосу від цих осіб (існує близько шести записів на одного пацієнта) і 23 характеристики голосу.

У якості ще одного прикладу розглянемо задачу визначення кредитної платоспроможності (кредитного скорінгу). Нехай є дані про клієнтів, які звернулися за кредитом. Тут об’єктами є клієнти, а ознаками можуть бути їхня стать, рівень доходу, освіта, інформація про те, є чи немає у них позитивної кредитна історія і т. д. Цільовою ознакою (відповіддю) виступає інформація про те, повернув клієнт кредит в банк чи ні. За цими даними потрібно навчитися передбачати, чи поверне кредит новий клієнт, який звернувся у банк. Таким чином, мова йде про задачу класифікації: потрібно визначити, до якого класу належить клієнт: позитивного (кредит буде повернений) або негативного (кредит не буде повернутий).

Скористаємось набором даних «Credit Approval Data Set» [10]. Цей датасет містить інформацію про 690 клієнтів, з яких 307 виплатили кредит, а 383 – ні. Опис кожного клієнта містить 14 ознак (6 числових і 8 категорійних). Оскільки цей файл стосується кредитних карт, то для захисту конфіденційності даних всі імена та значення ознак замінені деякими символами.

У якості прикладу задачі регресії розглянемо задачу прогнозування кількості велосипедів, які взято на прокат за день.

Будемо працювати з датасетом «Bike Sharing Dataset Data Set» [11], у якому по днях записана календарна інформація та погодні умови, що характеризують автоматизовані пункти прокату велосипедів, а також число орендованих у цей день велосипедів, яке потрібно спрогнозувати.

3. Порівняння методів бустінгу

Для проведення обчислювальних експериментів використано середовище Jupyter Notebook.

Виконаємо попередню обробку даних з датасету «Parkinsons Data Set», фрагмент якого представленого на рис. 3. Для цього необхідно виконати нормування ознак так, щоб кінцеві значення знаходилися в інтервалі від -1 до 1 і розділити вибірку на тренувальну, валідаційну і тестову.

```
In [2]: df = pd.read_csv('parkinsons.data')
        df.head()
```

Out[2]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425

5 rows × 24 columns

Рис. 3 Набір даних «Parkinsons Data Set»

Для оцінки результатів будемо використовувати функцію accuracy_score( ) з бібліотеки sklearn, тобто обрахуємо долю правильних відповідей на тестовій вибірці. Результати застосування різних алгоритмів бустінгу для прогнозування наявності хвороби Паркінсона представлено у табл.1.

Таблиця 1. Бустінг для задачі медичної діагностики

Алгоритм бустінгу	Доля правильних відповідей (accuracy score)	Час навчання
Адаптивний бустінг AdaBoost	0.8813	119 ms
Градієнтний бустінг	0.9573	116 ms
XGBoost	0.8983	37.8 ms
CatBoost	0.9152	3.86 s

Виконаємо попередню обробку даних з датасету «Credit Approval Data Set», фрагмент якого представленого на рис. 4. Для цього видалимо пропущені значення та закодуємо категорійні ознаки.

Застосуємо розглянуті методи бустінгу до задачі кредитного скорінгу, отримані результати представлено у табл. 2.

Таблиця 2. Бустінг для задачі кредитного скорінгу

Алгоритм бустінгу	Доля правильних відповідей (accuracy score)	Час навчання
Адаптивний бустінг AdaBoost	0.8594	76 ms
Градієнтний бустінг	0.8978	69 ms
XGBoost	0.8933	34 ms
CatBoost	0.8910	2.17 s

```
df = pd.read_csv('crx.data', header=None, names=['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8',
                 'A9', 'A10', 'A11', 'A12', 'A13', 'A14', 'A15', 'A16'])
df.head()
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+

Рис. 4 Набір даних «Credit Approval Data Set»

Виконаємо попередню обробку даних з набору даних «Bike Sharing Dataset Data Set» », фрагмент якого представленого на рис. 5. Для цього видалимо стовпчики atemp та windspeed(mph), і виконаємо нормування ознак.

```
df = pd.read_csv('bikes_rent.csv',index_col=False, sep=',')
df.head()
```

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed(mph)	windspeed(ms)	cnt
0	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	4.805490	985
1	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	7.443949	801
2	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	7.437060	1349
3	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	4.800998	1562
4	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	5.597810	1600

Рис. 5 Набір даних «Bike Sharing Dataset Data Set»

Для оцінки результатів будемо використовувати функцію score( ) з бібліотеки sklearn, тобто коефіцієнт детермінації R^2.

$$R^2 = \frac{S_{reg}}{S_{tot}},$$

де $S_{reg} = \sum_{i=1}^{n} (\hat{y}_i - \overline{y}_i)^2$, $S_{tot} = \sum_{i=1}^{n} (y_i - \overline{y}_i)^2$, $\hat{y}_i$ – прогнозоване значення параметра, $\overline{y}_i$ – середнє арифметичне значення параметру.

Результати застосування різних алгоритмів бустінгу для прогнозування кількості орендованих велосипедів представлено у табл. 3.

Таблиця 3. Бустінг для прогнозування кількості оренди

Алгоритм бустінгу	Коефіцієнт детермінації R^2	Час навчання
Адаптивний бустінг AdaBoost	0.7294	116 ms
Градієнтний бустінг	0.8470	104 ms
XGBoost	0.8483	57 ms
CatBoost	0.8698	2.6 s

Висновки

У роботі описано ідею методів градієнтного і адаптивного бустінгу та бібліотек XGBoost і CatBoost. Було проведено порівняння якості отримуваних прогнозів та часу навчання алгоритмів при розв'язанні задачі медичної діагностики, задачі кредитного скорінгу, задачі прогнозування кількості оренди велосипедів.

Результати обчислювальних експериментів показали, що загалом моделі бустінгу демонструють досить хороші результати як для задач класифікації, так і для задач регресії. Для задач бінарної класифікації кращі результати демонструють моделі, що використовують градієнтний бусттінг. Найменший час навчання і найкращі результати демонструє модель XGBoost, завдяки своїй апаратній і програмній оптимізації при реалізації даної бібліотеки. Модель CatBoost продемонструвала кращі результати для задачі регресії, але при цьому час її навчання завжди на порядок більший у порівняні з іншими моделями.

Список використаної літератури:

1. Кашницкий Ю. С. Ансамблевый метод машинного обучения, основанный на рекомендации классификаторов / Ю. С. Кашницкий, Д. И. Игнатов // Интеллектуальные системы. Теория и приложения, 2015. – Т. 19. №4, С. 37–55.
2. Кривохата А. Г. Застосування ансамблевого навчання в задачах класифікації акустичних даних / Кривохата А. Г., Кудін О. В., Давидовський М. В., Лісняк А. О. // Вісник Запорізького національного університету. Фізико-математичні науки, 2018. – №1. – С. 48-60.
3. Бустинг (Boosting) – Loginom Wiki [Електронний ресурс]. – Режим доступу: https://wiki.loginom.ru/articles/boosting.html.
4. Ensemble Methods in Machine Learning: Bagging Versus Boosting [Електронний ресурс]. – Режим доступу: https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting.
5. Алгоритм AdaBoost [Електронний ресурс]. – Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=AdaBoost.
6. Пивкин, К.С. Моделирование покупательского спроса на предприятиях розничной торговли на основе методов машинного обучения. [Текст]: дис. ... канд. экон. наук: 08.00.13: Ижевск, 2018. – 220 с.
7. Алгоритм XGBoost: пусть он царствует долго [Електронний ресурс]. – Режим доступу: https://cutt.ly/9nATUGb.
8. Быстрый градиентный бустинг с CatBoost / Блог компании OTUS / Хабр [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/otus/blog/527554.
9. UCI Machine Learning Repository: Parkinsons Data Set [Електронний ресурс]. – Режим доступу: https://archive.ics.uci.edu/ml/datasets/parkinsons.
10. UCI Machine Learning Repository: Credit Approval Data Set [Електронний ресурс]. – Режим доступу: https://archive.ics.uci.edu/ml/datasets/Credit+Approval.
11. UCI Machine Learning Repository: Bike Sharing Dataset Data Set [Електронний ресурс]. – Режим доступу: http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset.

References:

1. Kashnickij Yu. S. Ansamblevyj metod mashinnogo obucheniya, osnovannyj na rekomendacii klassifikatorov / Yu. S. Kashnickij, D. I. Ignatov // Intellektualnye sistemy. Teoriya i prilozheniya, 2015. – T. 19. №4, C. 37–55 [in Russian].
2. Krivohata A. G. Zastosuvannya ansamblevogo navchannya v zadachah klasifikaciyi akustichnih danih / Krivohata A. G., Kudin O. V., Davidovskij M. V., Lisnyak A. O. // Visnik Zaporizkogo nacionalnogo universitetu. Fiziko-matematichni nauki, 2018. – №1. – S. 48-60 [in Ukraine].

3. Busting (Boosting) – Loginom Wiki. Retrieved from: https://wiki.loginom.ru/articles/boosting.html [in Russian].
4. Ensemble Methods in Machine Learning: Bagging Versus Boosting (2020). – Retrieved from: https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting.
5. Algoritm AdaBoost (2016). – Retrieved from: http://www.machinelearning.ru/wiki/index.php?title=AdaBoost [in Russian].
6. Pivkin, K.S. (2018) Modelirovanie pokupatelskogo sprosa na predpriyatiyah roznichnoj torgovli na osnove metodov mashinnogo obucheniya: dis. ... kand. ekon. nauk: 08.00.13: Izhevsk [in Russian].
7. Algoritm XGBoost: pust on carstvuet dolgo (2019). – Retrieved from: https://cutt.ly/9nATUGb [in Russian].
8. Bystryj gradientnyj busting s CatBoost / Blog kompanii OTUS / Habr (2020). – Retrieved from: https://habr.com/ru/company/otus/blog/527554 [in Russian].
9. UCI Machine Learning Repository: Parkinsons Data Set. – Retrieved from: https://archive.ics.uci.edu/ml/datasets/parkinsons.
10. UCI Machine Learning Repository: Credit Approval Data Set. – Retrieved from: https://archive.ics.uci.edu/ml/datasets/Credit+Approval.
11. UCI Machine Learning Repository: Bike Sharing Dataset Data. – Retrieved from: http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset.

KONONENKO Veronika,
student, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
KRASNOSHLYK Natalia,
candidate of Technical Sciences, Associate Professor, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

USING BOOSTING METHODS FOR MACHINE LEARNING PROBLEMS

Summary. Introduction. *Among the modern methods of machine learning, one of the most effective are the methods used by ensembles of algorithms. Frequently used ensembles of algorithms include boosting. Boosting is a method of building an ensemble of algorithms, in which the basic algorithms are learned sequentially and each subsequent algorithm of the ensemble is applied to the results of the previous one. Boosting on decision trees remains one of the most effective and popular methods of machine learning. The article describes the idea of gradient and adaptive boosting methods, XGBoost and CatBoost libraries, also considers their practical application for machine learning problems. Examples of the binary classification problems are the problem of predicting Parkinson's disease in a patient and the problem of credit scoring. As an example of the regression problem, consider the problem of predicting the number of bicycles rented per day.*

Purpose. *The purpose of this paper is to apply and compare different methods of boosting in solving machine learning problems.*

Results. *Gradient boosting, AdaBoost, XGBoost and CatBoost were used to solve classification and regression problems in the Jupyter Notebook environment. A comparison of the quality of forecasts and the time of learning algorithms in solving the machine learning problems. To assess the quality in the classification problem, the accuracy score on the test dataset was calculated, and in the regression problem - the coefficient of determination.*

The results of computational experiments have shown that, in general, boosting models show quite good results for both classification and regression problems. For binary classification problems, models using gradient boosting show the best results. The XGBoost model demonstrates the shortest training time and the best results, thanks to its hardware and software optimization in the implementation of this library. The CatBoost model has shown better results for the regression problem, but its training time is always an order of magnitude longer than other models.

Conclusion. *This article describes the idea of gradient and adaptive boosting methods, XGBoost and CatBoost libraries. A comparison of the quality of forecasts and the time of learning algorithms in solving the problem of medical diagnostics, the problem of credit scoring, the problem of predicting the number of bicycle rent was compared.*

The XGBoost model has been found to have the lowest learning time and best results for binary classification problems. For the regression problem, the CatBoost model showed the best results, but its learning time is always an order of magnitude longer than other models.

Keywords*: machine learning, boosting, gradient boosting, XGBoost, CatBoost.*

Одержано редакцією 15.04.2021 р.
Прийнято до публікації 10.06.2021 р.

УДК 519.688

DOI 10.31651/2076-5886-2021-1-69-84

PACS 02.60.-x, 02.60.Pn

ХАРЧЕНКО Віталій Вікторович,
студент спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького

ДІДКОВСЬКИЙ Руслан Михайлович,
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

СЕРДЮК Олександр Анатолійович,
кандидат економічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: serdyuk@ukr.net
ORCID 0000-0002-3919-4661

РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ ПЕРЕВІРКИ ПРОГРАМНОГО КОДУ НА НАЯВНІСТЬ ПЛАГІАТУ

У роботі описано аналіз передумов розробки та структуру розробленої серверної системи перевірки програмного коду, що є результатом виконання студентами практичних завдань з програмування, на наявність плагіату. Умови розробки системи складено на основі аналізу чотирьох наявних систем пошуку плагіату: MOSS, Codeqiury, Unicheck, CCFinderX. Для кожної з розглянутих систем визначено її переваги та недоліки, на основі чого визначено потрібні властивості розроблюваної системи. Визначено структуру серверної частини та потрібне програмне забезпечення й мову програмування для практичної реалізації. Визначено та наведено алгоритми для токенізації програмного коду та порівняння двох різних фрагментів коду з метою оцінки їх подібності. Реалізована у результаті система може використовуватись для надання API програмам пошуку плагіату у програмному коді.

***Ключові слова**: плагіат, плагіат у програмному коді, система перевірки на наявність плагіату, серверна система.*

Вступ

Запозичення коду серед студентів – це проблема, яка непокоїть багатьох викладачів університету, що викладають дисципліни, пов'язані з вивченням програмування. На жаль, викладачу важко особисто прослідкувати за унікальністю всіх робіт студентів. Окрім того, велику частку робочого часу доводиться витрачати на перевірку робіт, замість того, щоб цей час присвятити, наприклад, підготовці ще якісніших завдань для студентів та ознайомленню з новітніми трендами у