КОМП'ЮТЕРНІ НАУКИ

УДК 004.8 https://doi.org/10.31713/vt2202337

Зубик Л. В., к.пед.н., доцент (Київський національний університет ім. Тараса Шевченка, м. Київ, zubyk.liudmyla@knu.ua)**, Зубик Я. Я., старший викладач** (Національний університет водного господарства та природокористування, м. Рівне**,** zubykjj@nuwm.edu.ua)

ЕФЕКТИВНІСТЬ АНСАМБЛЕВИХ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ОБРОБКИ ВЕЛИКИХ ДАНИХ

Ансамблеве навчання як добре відомий варіант машинного навчання традиційно вважається надійним й ефективним. У цих алгоритмах використовується більша кількість слабких учнів, наприклад, дерева рішень, об'єднаних для створення більш потужного сигналу.

Варіантами ансамблевого навчання є випадкові ліси, інші беговані (бутстрап-агреговані) та бустингові (форсовані) класифікатори. Вони продукують ознаковий простір, який може бути підрізаний з метою скорочення переважання перепідгонки.

Стаття спрямована на порівняння ефективності найбільш популярних ансамблевих методів машинного навчання та визначення специфіки їх використання у наукових дослідженнях.

***Ключові слова*: машинне навчання; ансамблеві методи; дерева рішень; випадковий ліс; бегінг; бустинг.**

Постановка проблеми. Моделі машинного навчання зазвичай потерпають від наступних типових ситуацій: присутність помилок, спричинених недосконалими припущеннями; помилки, викликані чутливістю до малих змін у тренувальній множині; наявність помилок, викликаних дисперсією спостережених значень, наприклад, непередбачуваними змінами або допущеними у вимірюваннях помилками. Мінімізувати зміщення/дисперсію допомагають ансамблеві методи, що спрямовані на поєднування множини слабких учнів, які базуються на одному й тому ж навчальному алгоритмі, з метою створення сильнішого учня, чия результативність краща ніж у будь-якого з окремо взятих учнів.

Аналіз досліджень. Пропагуванням методів ансамблевого навчання займалися і займаються зараз багато науковців, серед них: Майкл Кірнс (1989) [1], Роберт Шапіре (1990) [2], Лео Брейман (1998), Чжі-Хуа Чжоу (2012) та інші.

До методів ансамблевого навчання належить насамперед бегування. Бегування (пакетування) або агрегування бутстрапівських вибірок – це ефективний спосіб скорочення дисперсії у прогнозах.

Інший варіант машинного навчання – випадковий ліс. Дерева рішень загальновідомі тим, що вони схильні до перепідгонки, що збільшує дисперсію прогнозів. Для вирішення цієї проблеми був розроблений метод випадкового лісу з метою породження ансамблевих прогнозів із нижчою дисперсією. Випадковий ліс має спільні риси з бегінгом у сенсі тренування індивідуальних оцінювачів. Перевага використання випадкового лісу полягає в додатковому вбудованні другого рівня випадковості.

Окрім вищезгаданого, існує низка бустингових алгоритмів, найбільш популярним серед яких є адаптивний бустинг AdaBoost.

Постановка завдання. У роботі планується порівняти різні варіанти ансамблевого навчання: випадкові ліси, беговані (бутстрап-агреговані) класифікатори та бустингові (форсовані) класифікатори.

Для вирішення цієї мети необхідно пройти наступні етапи дослідження:

1. Проаналізувати різні способи продукування ознакового простору, який може бути підрізаний з метою скорочення переважання перепідгонки.

2. Порівняти ефективність найбільш популярних ансамблевих методів машинного навчання та визначити специфіку їх використання у наукових дослідженнях.

Типові випадки використання ансамблевих методів. Моделі машинного навчання зазвичай потерпають від наступних типових ситуацій:

– *зміщення*: помилка спричинена недосконалими припущеннями. Коли зсув занадто високо, це означає, що алгоритм машинного навчання не зміг розпізнати важливі зв'язки між ознаками й наслідками, тобто ми маємо недовизначеність у системі;

– *дисперсія*: помилка викликана чутливістю до малих змін у тренувальній множині. Коли дисперсія висока, це означає, що алгоритм занадто адаптований до тренувальної підмножини, відтак навіть мінімальні зміни у тренувальній підмножині можуть призвести до отримання кардинально різних прогнозів. Має місце наступний

випадок: замість того, щоб моделювати загальні закономірності у тренувальній підмножині, алгоритм помилково приймає шум за сигнал;

– *шум*: помилка викликана дисперсією спостережених значень, наприклад, непередбачувані зміни або ж помилки у вимірюваннях. Це непереборна помилка, яка не може бути пояснена жодною моделлю.

Розглянемо тренувальну підмножину спостережень $\{x_i\}_{i=1..n}$ і відповідно результати $\{y_i\}_{i=1..n}$. Припустимо, що існує функція $f[x]$ така, що $y=f[x]+\varepsilon$, де ε – це білий шум з $E[\varepsilon_i]=0$ і $E[\varepsilon_i^2]=\sigma_\varepsilon^2$. Потрібно оцінити функцію $f'[x]$, яка найкраще відповідає $f[x]$ у сенсі мінімізації дисперсії помилки оцінювання $E(y_i- f'[x_i]^2]$ (середньоквадратична помилка не може дорівнювати нулю через шум, який представлений як σ_ε^2). Ця середньоквадратична помилка може бути подана як

$$E\left[\left(y_i-\hat{f}\left[x_i\right]\right)^2\right]=\left[E\left[\hat{f}\left[x_i\right]-f\left[x_i\right]\right]\right]^2+V\left[\hat{f}\left[x_i\right]\right]+\sigma_\varepsilon{}^2. \quad (1)$$

Ансамблевий метод – це метод, який поєднує множину слабких учнів, які базуються на одному й тому ж навчальному алгоритмі з метою створення сильнішого учня, чия результативність краща, ніж у будь-якого з окремо взятих учнів. Ансамблеві методи допомагають мінімізувати зміщення/дисперсію.

Агрегація бутстрапів. Бегування (пакетування) або агрегування бутстрапівських вибірок – це ефективний спосіб скорочення дисперсії у прогнозах. Це працює наступним чином: по-перше, потрібно згенерувати *N* тренувальних підмножин даних за допомогою випадкового відбору із *поверненням*. По-друге, виконати припасування *N* оцінювачів, по одному на кожну окрему тренувальну підмножину. Ці оцінювачі підганяються незалежно один від одного, отже, моделі можуть бути підігнані паралельно. По-третє, ансамблевий прогноз – це *просте середнє арифметичне* індивідуальних прогнозів з *N* моделей. У випадку категоріальних змінних ймовірність того, що спостереження належить класу, визначається часткою оцінювачів, що класифікують це спостереження як члена цього класу (*мажоритарним голосуванням, тобто більшістю голосів*). Коли базовий оцінювач може робити прогнози з імовірністю передбачення, бегований класифікатор може набувати середнього значення ймовірностей.

Скорочення дисперсії. Головна перевага бегування полягає в тому, що воно зменшує дисперсію прогнозів, тим самим допомагаючи

вирішувати проблему перепідгонки. Дисперсія у бегованому прогнозуванні ($\varphi_i[c]$) є функцією числа бегованих оцінювачів (*N*), середньої дисперсії передбачення, що виконується одним оцінювачем (σ), та середньої кореляції між їх прогнозами (ρ):

$$V\left[\frac{1}{N}\sum_{i=1}^{N}\varphi_i[c]\right]=\frac{1}{N^2}\sum_{i=1}^{N}\left(\sum_{i=1}^{N}\sigma_{i,j}\right)=\frac{1}{N^2}\sum_{i=1}^{N}\left(\sigma_i^2+\sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j}\right)=$$

$$=\frac{1}{N^2}\sum_{i=1}^{N}\left(\bar{\sigma}^2+\sum_{j\neq i}^{N}\bar{\sigma}^2\bar{\rho}\right)=\frac{\sigma^2+(N-1)\bar{\sigma}^2\bar{\rho}}{N}=\bar{\sigma}^2\left(\bar{\rho}+\frac{1-\bar{\rho}}{N}\right), \qquad (2)$$

де $\sigma_{i,j}$ – це коваріація передбачень за оцінювачами *i, j*;

$$\sum_{i=1}^{N}\bar{\sigma}^2=\sum_{i=1}^{N}\sigma^2 \Leftrightarrow \bar{\sigma}^2=N^{-1}\sum_{i=1}^{N}\sigma_i^2;\ \text{і}$$

$$\sum_{j\neq i}^{N}\bar{\sigma}^2\bar{\rho}=\sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j} \Leftrightarrow \bar{\rho}=\left(\bar{\sigma}^2N(N-1)\right)^{-1}\sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j}\,. \qquad (3)$$

Наведене вище рівняння показує, що бегування ефективне лише в тій мірі, в якій $\bar{\rho}<1$; тоді як $\bar{\rho}\to 1\Rightarrow V\left[\frac{1}{N}\sum_{i=1}^{N}\varphi_i[c]\right]\to\bar{\sigma}^2$. Одна з цілей послідовного бутстрапування полягає в тому, щоб проводити відбори зразків якомога більш незалежно, тим самим зменшуючи ρ, що має знизити дисперсію бегованих класифікаторів. На рис. 1 наведено діаграму середньоквадратичного відхилення бегованого прогнозування як функції $N\in[5,30],\quad \bar{\rho}\in[0,1],\quad \bar{\sigma}=1.$

Поліпшена точність. Розглянемо бегований класифікатор, який виконує прогноз на *k* класах мажоритарним голосуванням серед *N* незалежних класифікаторів. Можна позначити всі прогнози як *{0,1}*, де *1* відповідає правильному передбаченню. Точність класифікатора - це ймовірність *p* позначити прогноз як *1*. У середньому ми отримаємо N_p прогнозів, промаркованих як *1* з дисперсією N_p *(1 - p)*. Мажоритарне голосування дає правильне передбачення, коли спостерігається найбільш прогнозований клас. Наприклад, для *N = 10* і *k = 3* бегований класифікатор зробив правильне передбачення, коли спостерігався клас *A*, і віддані голоси були *[A, B, C] = [4, 3, 3]*.

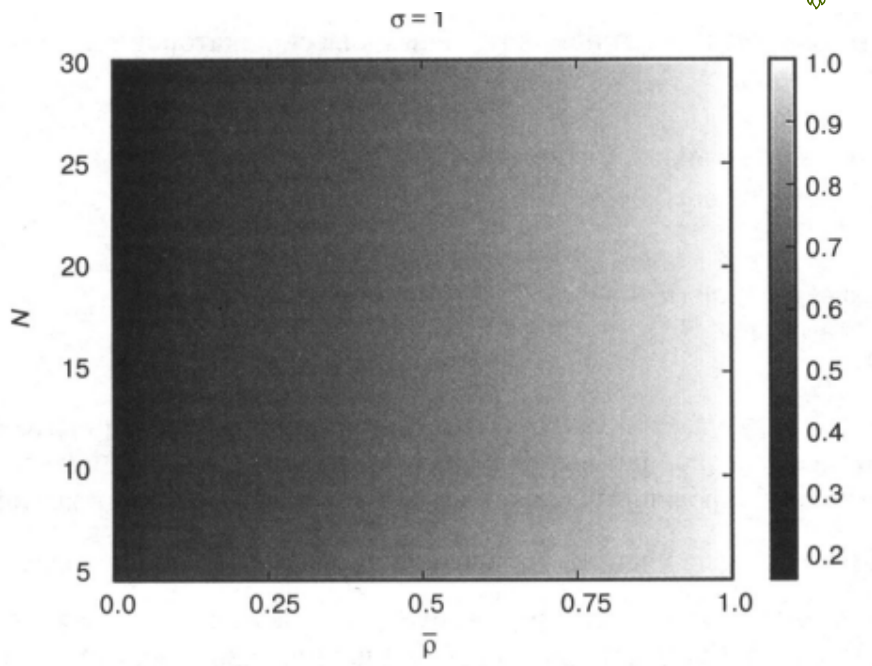

Рис. 1. Стандартне відхилення бегованого прогнозу

Проте бегований класифікатор зробив неправильне передбачення, коли спостерігався клас *A*, і віддані голоси були *[A, B, C] = [4, 1, 5]*. Достатньою умовою є те, що сума цих міток дорівнює *X > N/2*. Необхідною (але не достатньою) умовою є те, що *X > N/k* , що відбувається з ймовірністю

$$P\left[X > \frac{N}{k}\right] = 1 - P\left[X \le \frac{N}{k}\right] = 1 - \sum_{i=0}^{\lfloor N/k \rfloor} \binom{N}{i} p^i (1-p)^{N-i}. \qquad (4)$$

Із цього випливає, що для досить великого *N*, скажімо *N>p(p-1/k)*$^{-2}$*, p>1/k =>P[X>N/k]>p*, отже, точність бегованого класифікатора перевищує середню точність індивідуальних класифікаторів.

Це достатньо сильний аргумент на користь бегування будь-якого класифікатора загалом у випадку, коли це дозволяють обчислювальні можливості. Однак, на відміну від бустингу, бегування не може покращити точність слабких класифікаторів: якщо індивідуальні учні є слабкими класифікаторами (p<<1/k), мажоритарне голосування, як і раніше, показуватиме слабку результативність (хоча і з нижчою дисперсією).

Рис. 2 ілюструє ці факти. Оскільки легше досягти p<<1 ніж p>1/k, бегування має більше шансів бути успішним у скороченні дисперсії, ніж у скороченні зміщення.

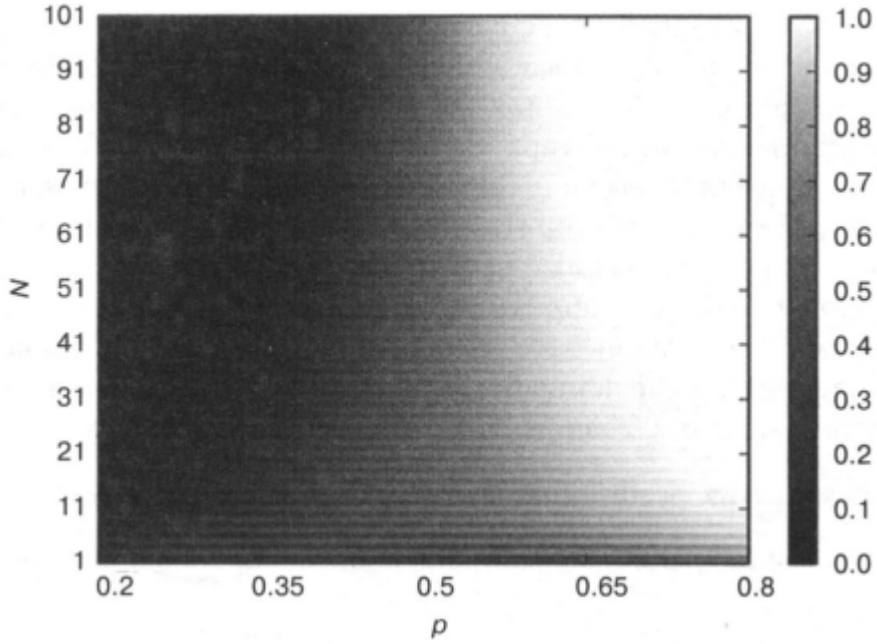

Рис. 2. Правильність бегованого класифікатора як функція точності індивідуального оцінювача (*P*), числа оцінювачів (*N*) та *k = 2*

Надмірність спостережень. Розглянемо випадок, коли спостереження не можна вважати однаково розподіленими та взаємно незалежними. Надмірні спостереження здійснюють згубні впливи на бегування. Зразки, що вилучаються з поверненням, з більшою ймовірністю будуть практично ідентичними, навіть якщо вони не мають спільних спостережень. Це робить $p \approx 1$, і бегування не скоротить дисперсію, незалежно від *N*. Наприклад, якщо кожне спостереження в *t* промарковано відповідно до повернення між *t* і *t+100*, то ми повинні відібрати 1% спостережень у розрахунку бегованого оцінювача, але не більше.

Рекомендуються три альтернативні рішення, одне з яких полягає в установці *max_samples=out ['tW'].mean()* у реалізації класу бегованого класифікатора в бібліотеці *sklearn*. Більш якісним рішенням є застосування методу спадкового бутстрапівського відбору.

Ще один згубний вплив надмірності спостережень полягає в тому, що буде роздута позапакетна точність. Це відбувається через те, що випадковий відбір з поверненням зразків поміщає в тренувальну підмножину зразки, які дуже схожі на ті, що поза пакетом. У такому разі правильна стратифікована *k*-блочна

перехресна перевірка без перетасування перед розбиттям покаже набагато меншу точність на тестовій підмножині, ніж та, яка була оцінена поза пакетом.

З цієї причини при використанні цього класу бібліотеки *sklearn* рекомендується встановити *stratifiedKFold(n_splits=k, shuffle=False)*, перехресно перевірити бегований класифікатор та проігнорувати результати позапакетної точності. Низьке число *k* краще від високого, оскільки надмірне розбиття знову помістить у тестову підмножину зразки, занадто схожі на ті, що використовуються в тренувальній підмножині.

Випадковий ліс. Дерева рішень загальновідомі тим, що вони схильні до перепідгонки, що збільшує дисперсію прогнозів. Для вирішення цієї проблеми був розроблений метод випадкового лісу (англ. *random forest, RF*) для породження ансамблевих прогнозів із нижчою дисперсією.

Випадковий ліс має деякі спільні риси з бегінгом у сенсі тренування індивідуальних оцінювачів незалежно один від одного на бутстрапованих підмножинах даних. Ключова відмінність від бегінгу полягає в тому, що у випадковому лісі вбудовується другий рівень випадковості: під час оптимізації кожного вузлового дріблення оцінюватиметься лише випадкова підвибірка (без повернення) атрибутів для подальшого декорелювання оцінювачів.

Як і бегінг, випадковий ліс зменшує дисперсію прогнозів без перепідгонки (доти, поки $\overline{\rho} < 1$). Інша перевага полягає в тому, що випадковий ліс оцінює важливість ознак. Ще одна перевага полягає в тому, що випадковий ліс надає оцінки позапакетної точності, однак у застосунках, спрямованих на обробку великих даних, вони, швидше за все, будуть завищені. Але як і бегінг, випадковий ліс не обов'язково демонструватиме більш низьке зміщення, ніж індивідуальні дерева рішень.

Якщо велика кількість зразків надмірна (не є однаково розподіленими і взаємно незалежними), все одно матиме місце перепідгонка: випадковий відбір із поверненням побудує велику кількість практично ідентичних дерев (p≈1), де кожне дерево рішень перепідігнано (недолік, завдяки якому дерева рішень є сумно відомими). На відміну від бегінгу, випадковий ліс завжди задає розмір бутстрапованих вибірок відповідно з розміром тренувальної підмножини даних.

Деталізуємо, як можна вирішити проблему перепідгонки випадкових лісів у бібліотеці *sklearn*. Загалом наведені рішення можуть бути застосовані до будь-якої реалізації:

1. Встановити для параметра *max_features* менше значення, щоб досягти розходження між деревами.
2. Встановити параметр регуляризації *min_weight_fraction_leaf* рівним досить великому значенню (наприклад, 5%) для того, щоб позапакетна точність сходилася до позавибіркової (*k*-блокової) правильності.
3. Використовувати оцінювач *BaggingClassifier* на базовому оцінювачі *DecisionTreeClassifier*, де *max_samples* встановлений рівним середньої унікальності (*avgU*) між вибірками.

- clf=DecisionTreeClassifier(criterion='entropy', max_features='auto', class_weight='balanced')

- bc = BaggingClassifier(base_estimator=clf, n_estimators = 1000, max_samples=avgU, max_features=l.)

4. Використовувати оцінювач *BaggingClassifier* на базовому оцінювачі *RandomForestClassifier*, де *max_samples* встановлений рівним середньої унікальності (*avgU*) між вибірками.

- clf=RandomForestClassifier(n_estimators=l, criterion='entropy', bootstrap=False, class_weight='balanced_subsample')

- bc=BaggingClassifier(base_estimator=clf, n_estimators=1000, max_samples=avgU, max_features=l.)

5. Модифікувати клас випадкового лісу для заміни стандартного бутстрапування на послідовне бутстрапування.

При припасуванні дерев рішень поворот ознакового простору в напрямку, що збігається з осями, як правило, скорочує кількість необхідних дереву рівнів. З цієї причини доцільно виконувати припасування випадкового дерева на PCA ознак, оскільки це може прискорити обчислення та дещо скоротити час перепідгонки. Крім того, аргумент *class_weight='balanced_subsample'* допоможе не допустити, щоб дерева неправильно класифікували міноритарні класи.

Бустинг. Історично питання об'єднання слабких оцінювачів для досягнення високоточного оцінювача вперше було поставлено Kearns and Valiant [1] і вирішено Schapire [2], який описав процедуру, що нині має назву бустинг (англ. *boosting* – форсування, посилення).

Загалом бустинг працює наступним чином:

– генерування однієї тренувальної підмножини шляхом випадкового відбору (з поверненням) у відповідності до деяких ваг вибірки (що ініціалізуються рівномірними вагами);

– налаштування одного окремого оцінювача за допомогою цієї тренувальної множини;

– якщо одиночний оцінювач досягає точності, що перевищує поріг прийнятності (наприклад, у бінарному класифікаторі вона дорівнює 50%, тобто, щоб класифікатор працював краще, ніж просто випадкове вгадування), то оцінювач залишається, інакше він відкидається;

– надається більшої ваги неправильно класифікованим спостереженням і меншої ваги правильно класифікованим спостереженням;

– повторюються попередні кроки до тих пір, поки не будуть отримані N оцінювачів;

– будується ансамблевий прогноз як середньозважене значення окремих прогнозів з *N* моделей, де ваги визначаються точністю індивідуальних оцінювачів.

Існує низка бустингових алгоритмів, з яких адаптивний бустинг AdaBoost є одним з найбільш популярних (Geron [3]).

Порівняння використання бегінгу і бустингу для вирішення задач, орієнтованих на обробку великих даних. Бегінг і бустинг суттєво відрізняються за наступними аспектами:

– Підгонка індивідуальних класифікаторів виконується послідовно.

– Слаборезультативні класифікатори відхиляються.

– На кожній ітерації спостереження зважуються по-різному.

– Ансамблевий прогноз по суті є середньозваженим значенням індивідуальних учнів.

Перевага бустингу в основному полягає в тому, що він скорочує як дисперсію, так і зміщення в прогнозах. Проте виправлення зміщення відбувається за рахунок більшого ризику перепідгонки. Можна стверджувати, що у застосунках, орієнтованих на обробку великих даних, бегінг зазвичай буде кращим, аніж бустинг. Бегінг вирішує завдання перепідгонки, тоді як бустинг вирішує завдання недопідгонки. Перепідгонка часто є більш серйозною проблемою ніж недопідгонка, тому що підігнати алгоритм машинного навчання занадто щільно до даних не важко через низьке співвідношення

сигнал/шум. Більш того, бегінг піддається розпаралелюванню, тоді як бустинг зазвичай вимагає послідовного виконання.

Вирішення задачі масштабованості. Як відомо, деякі популярні алгоритми машинного навчання масштабуються не дуже добре в залежності від розміру вибірки. Яскравим прикладом є метод опорних векторів (*support vector machines, SVM*). Якщо ви спробуєте виконати припасування оцінювача SVM на мільйоні спостережень, то може знадобитися тривалий час, доки алгоритм не стане збіжним. І навіть після цього немає жодної гарантії, що рішення є глобальним оптимумом або що він не буде перепідігнаний.

Один із підходів полягає в побудові бегованого алгоритму, де базовий оцінювач належить класу, який погано масштабується разом із розміром вибірки, наприклад SVM. При визначенні цього базового оцінювача можна запровадити жорстку умову ранньої зупинки. Наприклад, у реалізації опорно-векторних машин (SVM) у бібліотеці *sklearn* можна поставити низьке значення для параметра *max_iter*, наприклад, 100 000 ітерацій. За замовчуванням прийнято значення *max_iter=-1*, яке повідомляє оцінювачу про продовження виконання ітерацій доти, доки помилки не будуть вписуватися у допуски. З іншого боку, можна підвищити рівень допуску за допомогою параметра *tol*, який за замовчуванням має значення *tol=0,001*. Будь-який із цих двох параметрів призведе до ранньої зупинки. Можна також зупиняти рано й інші алгоритми за допомогою еквівалентних параметрів, таких як кількість рівнів у випадковому лісі (*max_depth*) або мінімальна зважена частка підсумкової суми ваг (всіх вхідних вибірок), зобов'язаних перебувати на листовому вузлі (*min_weight_fraction_leaf*).

Висновки. Два найбільш популярні ансамблеві методи, такі як бегінг та бустинг, працюють подібним чином, але бустинг може додавати нові моделі, які добре працюють там, де попередні моделі зазнають невдачі. Обидва згадані ансамблеві методи генерують кілька навчальних наборів даних шляхом випадкової вибірки, але лише бустинг визначає ваги для даних, щоб схилити терези на користь найскладніших випадків. Методи ансамблю приймають остаточне рішення шляхом усереднення N учнів або ж орієнтується на більшість з них, але це однаково зважене середнє для бегінгу і зважене середнє для бустингу, тобто з більшою вагою враховуються ті значення, які мають кращі показники на тренувальних даних. Обидва методи добре зменшують дисперсію та забезпечують більшу стабільність, але лише бустинг намагається зменшити зміщення. З

іншого боку, бегінг може вирішити проблему надмірного пристосування, тоді як бустинг може лише її посилювати. Враховуючи, що бегінгові алгоритми можуть бути розпаралелені, стає можливим трансформувати велику послідовну задачу в низку дрібніших, які виконуються одночасно. Рання зупинка збільшить дисперсію результатів від індивідуальних базових оцінювачів; проте це збільшення може бути компенсовано зменшенням дисперсії, пов'язаним з бегінговим алгоритмом. Це скорочення можна контролювати, додаючи нових незалежних базових оцінювачів. Таким чином, застосування бегінгу дозволяє отримувати більш швидкі оцінки на великих сукупностях даних.

1. Kearns, M. and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *In Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. 1989. New York : Association for Computing Machinery, 1989. Pp. 433–444. **2.** Schapire R. The strength of weak learnabllitp Machine Learning. *Kluwer Academic Publishers*. 1990. Vol. 5 No. 2. Pp. 197–227. **3.** Geron, A. Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 1st edition. O'Reilly Media. 2017. **4.** Gareth J., Witten D., Hastie T., and Tibshirani R. An Introduction to Statistical Learning: With Applications in R, 1st ed. Springer-Verlag. 2013. **5.** Hackeling G. Mastering Machine Learning with Scikit-Learn, 1st ed. Packt Publishing. 2014. **6.** Hastie T., Tibshirani R. and. Friedman. The Elements of Statistical Learning, 2nd ed. Springer-Verlag. 2016. **7.** Hauck T. Scikit-Learn Cookbook, 1st ed. Packt Publishing. 2014. **8.** Raschka S. Python Machine Learning, 1st ed. Packt Publis. 2015. **9.** Ensemble methods. URL: https://scikit-learn.org/stable/modules/ensemble.html (дата звернення: 25.05.2023). **10.** Random forest: many are better than one. URL: https://quantdare.com/random-forest-many-are-better-than-one/ (дата звернення: 25.05.2023). **11.** What is the difference between Bagging and Boosting? URL: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/ (дата звернення: 25.05.2023).

REFERENCES:

1. Kearns, M. and L. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *In Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. 1989. New York : Association for Computing Machinery, 1989. Pp. 433–444. **2.** Schapire R. The strength of weak learnabllitp Machine Learning. *Kluwer Academic Publishers*. 1990. Vol. 5 No. 2. Pp. 197–227. **3.** Geron, A. Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 1st

edition. O'Reilly Media. 2017. **4.** Gareth J., Witten D., Hastie T., and Tibshirani R. An Introduction to Statistical Learning: With Applications in R, 1st ed. Springer-Verlag. 2013. **5.** Hackeling G. Mastering Machine Learning with Scikit-Learn, 1st ed. Packt Publishing. 2014. **6.** Hastie T., Tibshirani R. and. Friedman. The Elements of Statistical Learning, 2nd ed. Springer-Verlag. 2016. **7.** Hauck T. Scikit-Learn Cookbook, 1st ed. Packt Publishing. 2014. **8.** Raschka S. Python Machine Learning, 1st ed. Packt Publis. 2015. **9.** Ensemble methods. URL: https://scikit-learn.org/stable/modules/ensemble.html (data zvernennia: 25.05.2023). **10.** Random forest: many are better than one. URL: https://quantdare.com/random-forest-many-are-better-than-one/ (data zvernennia: 25.05.2023). **11.** What is the difference between Bagging and Boosting? URL: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/ (data zvernennia: 25.05.2023).

Zubyk L. V., Candidate of Pedagogic Sciences (Ph.D.), Associate Professor (Taras Shevchenko National University of Kyiv, Kyiv, zubyk.liudmyla@knu.ua)**, Zubyk Y. Y., Senior Lecturer** (National University of Water and Environmental Engineering, Rivne, zubykjj@nuwm.edu.ua)

EFFICIENCY OF ASSEMBLE METHODS OF MACHINE LEARNING FOR BIG DATA PROCESSING

Ensemble learning, as a well-known variant of machine learning, is traditionally considered reliable and effective. These algorithms use a larger number of weak learners, such as decision trees, combined to create a stronger signal.

Variants of ensemble learning are random forests, other bootstrap-aggregated classifiers, and boosting (forced) classifiers. They produce a feature space that can be trimmed to reduce the prevalence of overfitting.

The article is aimed at comparing the effectiveness of the most popular ensemble methods of machine learning and determining the specifics of their use in scientific research.

Two most popular ensemble methods, such as bagging and boosting, are built independently for bagging, but boosting tries to add new models that do well where previous models was fail. Both mentioned ensemble methods generate several training data sets by random sampling, but only boosting determines weights for the data to tip the scales in benefit of the most difficult cases.

Ensemble methods make the final decision by averaging the N learners, or taking the majority of them. And we use one of the options: an equally weighted average for bagging or a weighted average for boosting with more weight to those with better performance on training data.

Both methods are good at reducing variance and provide higher stability, but only boosting tries to reduce bias. On the other side, bagging may solve the over-fitting problem, while boosting can increase it.

Given that begging algorithms can be parallelized, it becomes possible to transform a large sequential task into a number of smaller ones that are executed simultaneously. Thus, the application of bagging allows obtaining faster estimates on large data sets.

***Keywords*: machine learning; ensemble methods; decision trees; random forest; bagging; boosting.**
