https://doi.org/10.31891/2307-5732-2025-359-13
УДК 004.8:004.93:004.94:528.8

ГАРКУША ІГОР
Національний технічний університет «Дніпровська політехніка»
https://orcid.org/0000-0003-1190-1501
e-mail: garkusha.i.m@nmu.one
ІВАНОВ ДЕНИС
Національний технічний університет «Дніпровська політехніка»
https://orcid.org/0000-0001-8660-0928
e-mail: ivanov.d.v@nmu.one

ПЕРСПЕКТИВИ ВИКОРИСТАННЯ ТА ОЦІНКА ШВИДКОДІЇ БІБЛІОТЕКИ MLPACK В ЗАДАЧАХ ОБРОБКИ ДАНИХ ДИСТАНЦІЙНОГО ЗОНДУВАННЯ ЗЕМЛІ

У статті проводиться огляд реалізації та порівняння швидкодії певних алгоритмів машинного навчання сучасними бібліотеками – mlpack та scikit-learn. Розглянуті алгоритм пошуку k-найближчих сусідів (k-NN) та класифікація з вчителем за алгоритмом випадкового лісу (Random Forest). В якості прикладу датасету для порівняння алгоритму k-NN пошуку обраний датасет Covertype з відомого репозиторію університету Каліфорнії в Ірвайні. Датасет для порівняння швидкодії класифікації на базі алгоритму Random Forest власно створений на основі мультиспектральних даних космозйомки, отриманих з апарату дистанційного зондування Землі Sentinel-2A. Наведені певні особливості використання mlpack та іншої бібліотеки Armadillo, яку використовує mlpack. Зокрема описуються особливості завантаження форматованих даних функціями цієї бібліотеки. В роботі розглянутий процес підготовки датасету для навчання класифікатору. Наведені узагальнені кроки обробки наборів даних для навчання, наведений перелік типів земного покриття території зйомки та пропонуються рішення, щодо покращення розділення даних у просторі ознак. У статті наведена точність класифікації з вчителем алгоритмом Random Forest із певними гіперпараметрами моделі та дана оцінка часу на тренування і класифікацію набору даних для тестування. Дані для тестування склали 20% від загальної кількості даних сформованого за певними ознаками датасету. Результатами дослідження виступає час класифікації у програмах, які створені мовами програмування C++ та Python і які використовують відповідно бібліотеки mlpack та scikit-learn.

Ключові слова: *mlpack, scikit-learn, Sentinel-2, Random Forest, машинне навчання, класифікація з вчителем.*

GARKUSHA IGOR
IVANOV DENYS
Dnipro University of Technology

PROSPECTS OF USE AND THE SPEED ASSESSMENT OF THE MLPACK LIBRARY IN REMOTE SENSING OF THE EARTH DATA PROCESSING TASKS

The article reviews the implementation and comparison of the performance of certain machine learning algorithms by modern libraries – mlpack and scikit-learn. The k-nearest neighbor (k-NN) search algorithm and supervised classification using the Random Forest algorithm are considered. As an example of a dataset for comparing the k-NN search algorithm, the Covertype dataset from the well-known repository of the University of California, Irvine was chosen. The dataset for comparing the performance of classification based on the Random Forest algorithm was created on the basis of multispectral multiband image data from the Sentinel-2A Earth remote sensing device. Certain features of using mlpack and another Armadillo library, which mlpack uses, are presented. In particular, the features of loading formatted data by the functions of this library are described. The paper considers the process of preparing a dataset for training a classifier. The generalized steps of processing datasets for training are presented, a list of land cover types of the survey area is given, and solutions are proposed to improve data resolution in feature space. The article presents the classification accuracy with the Random Forest algorithm with certain model hyperparameters and gives an estimate of the time for training and classification of the test dataset. The data for testing amounted to 20% of the total amount of data of the dataset formed according to certain features. In the process of creating the dataset, calculations of vegetation (multispectral) indices were involved, in particular: normalized difference vegetation index (NDVI), normalized difference water index (NDWI), inverse Red-Green index (simple ratio G/R), as well as the principal components PC1 and PC2, which were calculated using the Principal Component Analysis. The results of the study are the classification time in programs created in the C++ and Python programming languages and using the mlpack and scikit-learn libraries, respectively.

Keywords: *mlpack, scikit-learn, Sentinel-2, Random Forest, machine learning, supervised classification.*

Стаття надійшла до редакції / Received 09.09.2025
Прийнята до друку / Accepted 01.11.2025

Постановка проблеми

При використанні в останні роки великої кількості методів машинного навчання (machine learning, ML), технологій штучного інтелекту при обробці великих об'ємів даних для різних галузей суспільства, важливу складову грає не тільки отримання високої якості та достовірності результатів, а й час, що витрачається на обробку даних. Програмне забезпечення (ПЗ) для побудови ML-систем будується на базі або із залученням таких передових фреймворків, як TensorFlow, PyTorch, бібліотек scikit-learn, Apache Spark MLlib та інших. Саме ці фреймворки та бібліотеки займають високі рейтинги серед розробників, які будують подібне ПЗ. У всіх з них є одна особливість – переважна їх більшість гарно адаптовані, або створені під інтерпретовану мову програмування Python. Залучаються додаткові обчислювальні можливості таких Python-бібліотек як NumPy та SciPy, що створені мовою програмування C/C++ під потреби саме програмного коду мовою Python. Але часто потрібно для підвищення працездатності ПЗ та більш ефективного використання ресурсів залучати інші мови програмування та якось адаптувати використання вже відомих бібліотек при побудові ПЗ. Тому є актуальним

розвиток бібліотек та фреймворків для рішення ML-задач суто компільованими мовами програмування, наприклад, такими як: C/C++, Go, Rust та іншими.

В роботі розглянуто певне порівняння бібліотеки для ML-задач mlpack (machine learning pack), що створена мовою C++ та бібліотеки scikit-learn, яка активно використовується в розробках мовою програмування Python.

Аналіз досліджень та публікацій

Перша версія mlpack вийшла у 2011 році та була описана в роботі [1]. Бібліотека побудована на базі класів-шаблонів C++ та має можливість використання і в інших мовах програмування (наприклад, Python, R, Go). Сама бібліотека залучає для виконання основної обчислювальної роботи іншу відому бібліотеку лінійної алгебри Armadillo [2, 3]. В свою чергу Armadillo використовують для операцій лінійної алгебри з векторами та матрицями. Armadillo використовує в своїй роботі різні реалізації алгоритмів таких відомих бібліотек як BLAS/OpenBLAS та LAPACK. Таким чином, фактично бібліотека mlpack використовує поєднання множини бібліотек Armadillo, BLAS/OpenBLAS та LAPACK.

В роботі [1] описані основні результати порівняння mlpack з відомими аналогами. Наприклад, порівнюється час проведення розрахунків на різних датасетах між mlpack та таким ПЗ як: Weka (Waikato Environment for Knowledge Analysis, мова розробки – Java), Shogun (мова розробки – C++), MATLAB (мови розробки – C/C++, MATLAB), mlpy (мови розробки – Python, C/C++) та scikit-learn (мови розробки – Python, Cython, C/C++). Оскільки праця була опублікована у 2013 році, то в процесі виконання поточного дослідження виконана спроба повторної оцінки вже сучасних версій mlpack та scikit-learn на більш сучасній апаратурі.

У праці [1] в якості вхідних даних використаний відомий набір ML-даних Covertype, що надається репозиторієм університету Каліфорнії в Ірвайні [4, 5]. В [1, с. 803] подано порівняння обробки набору даних Covertype алгоритмами, що реалізують пошук *k*-найближчих сусідів (*k*-nearest neighbors, *k-NN*). Як стверджують автори [1], станом на 2013 рік ефективність використання mlpack для цього алгоритму була вище за часом виконання майже в 45 разів в порівнянні із використанням scikit-learn! За результатами в [1], час *k-NN* пошуку програмою з mlpack для датасету Covertype складав 14,34 секунди, а програмою на базі scikit-learn він склав 651,63 секунди (~11 хвилин).

Також в ході дослідження, порівняння проводилося між класифікацією за алгоритмом випадкового лісу (Random Forest) Python-програмою із залученням бібліотеки scikit-learn та C++-програмою із залученням бібліотеки mlpack. Слід зазначити, що реалізація Random Forest в mlpack була додана наприкінці березня 2018 року починаючи з версії 3.0.

Вперше алгоритм Random Forest запропонований у 2001 році в роботі [6]. Це ансамблевий ML-метод для класифікації та регресії. Останні роки його можна зустріти в ПЗ для обробки даних дистанційного зондування Землі (ДЗЗ) з космосу, наприклад, в продуктах QGIS та ESA SNAP. Тому його порівняння між mlpack (C++) та scikit-learn (Python) для побудови власних програм розрахунків та скриптів обробки великих наборів даних саме ДЗЗ є дуже актуальним.

Формулювання цілей статті

Метою роботи є порівняльна оцінка швидкодії виконання алгоритму *k-NN* пошуку засобами бібліотеки mlpack в програмах, що розроблені мовою програмування C++ та засобами бібліотеки scikit-learn в Python-програмах, а також порівняння швидкодії та точності класифікатору Random Forest, що реалізований в цих бібліотеках на прикладі мультиспектральних даних космозйомки апарату ДЗЗ Sentinel-2A.

Виклад основного матеріалу

В роботі розглядалися бібліотека mlpack версії 4.3.0 (входить за замовченням до складу репозиторію ПЗ операційної системи Ubuntu 24.04) та scikit-learn версії 1.7.1. Слід зауважити, що операційні системи, які відносяться до GNU/Linux-сумісних, найчастіше використовуються фахівцями у складі обчислювальних вузлів хмарних оточень (наприклад, в хмарних оточеннях Amazon, Google, Microsoft та ін.).

Узагальненими кроками підготовки та обробки наборів даних в дослідженні були такі.

1. Проведення попереднього аналізу структури датасету. На цьому кроці дивляться на структуру файлу даних або готують дані під певний формат. Дивляться на заголовок даних, з яких атрибутів-ознак вони складаються та визначаються (за потреби) з даними, які будуть виступати в якості цільового вектору. Часто дані досліджень представляються у форматованих текстових файлах з роздільниками полів (пробіл або табуляція, або кома, або крапка з комою) – в CSV-форматі. Є й інші формати, наприклад, HDF-формат. При використанні mlpack краще готувати дані в CSV-форматі з роздільниками пробілом або комою. В поточній версії mlpack багаторазово перевантажена функція mlpack::data::Load( ) за замовченням не розуміє роздільник крапка з комою. Тому в цьому варіанті потрібно, наприклад, використовувати для завантаження даних із файлів з іншими роздільниками функції бібліотеки Armadillo.

2. Виконання завантаження даних з врахуванням формату файлу та інформації про вміст. При використанні бібліотек Python набули популярність функції з різноманітних модулів та пакетів бібліотек такі, як: csv.reader( ), numpy.loadtxt( ), pandas.read_csv( ) та ін. При використанні mlpack дослідники можуть залучати як багаторазово перевантажену функцію mlpack::data::Load( ), так і, за потреби, виклик перевантаженої функції бібліотеки Armadillo – Mat<eT>::load( ). Зауважимо, що кожна матриця даних, як об'єкт в Armadillo, має таку функцію завантаження. При використанні в якості роздільника, наприклад, крапки з комою, викликом Mat<eT>::load( ) огортають виклик іншої перевантаженої функції Armadillo: csv_name( ). У якості третього параметру, в цьому випадку, залучають аргумент-опцію csv_opts::semicolon. Наприклад:

```
#include <mlpack.hpp>
. . .
using namespace arma;
using namespace mlpack;
. . .
mat dataset;
field<std::string> header(7); // наприклад, в data.csv – 7 полів: N;R;G;B;PC1;PC2;Class
dataset.load(csv_name("data.csv", header, csv_opts::semicolon));
```

3. Проведення передобробки даних до їх використання в основному процесі обробки. Найчастіше на цьому кроці виконують певні перетворення отриманих даних: відділення потрібної множини даних від основної їх частини за певним критерієм, транспонування матриці даних (за потреби), розділення даних на дані для формування матриці ознак та на дані для цільового вектору, нормалізація даних (за потреби в залежності від задачі та даних). Важливою особливістю mlpack є те, що бібліотека використовує матриці, які представлені в реалізації через бібліотеку Armadillo. Тому при обробці слід враховувати, що на відміну від NumPy (який використовується у scikit-learn), дані в таких матрицях зберігаються з орієнтацією на стовпці, а не на рядки! Тобто матрицю додатково транспонують. Приклади цього можна побачити у авторів в [3]. При використанні функції mlpack::data::Load( ) це враховується, але при використанні функцій завантаження Armadillo про це потрібно подбати розробнику – робити транспонування після завантаження. Оскільки в роботі, в єдиному файлі даних, що завантажувався функціями Armadillo, містилася матриця ознак разом із цільовим вектором, то для їх розділення можна обрати такий підхід:

```
// формуємо цільовий вектор labels:
rowvec rlabels = dataset.col(dataset.n_cols - 1).t( );
Row<size_t> labels = arma::conv_to<arma::Row<size_t>>::from(rlabels.row(0));
// видаляємо останній стовпець
dataset.shed_cols(dataset.n_cols - 1, dataset.n_cols - 1);
// отримаємо транспоновану матрицю ознак dataset:
dataset=dataset.t( );
```

В цьому прикладі показаний фрагмент тексту програмного коду, який враховує, що дані цільового вектору розміщені в останньому стовпці завантаженого ранішу датасету (пункт 2 вище) та до транспонування.

4. Виконання розбиття набору даних на піднабори для тренування та тестування або для тренування, валідації та тестування. Крок виконують в залежності від потреби для використання цих даних певними обчислювальними методами. Найчастіше піднабори даних потребують різні класифікатори та штучні нейронні мережі. В mlpack для розділення даних використовують функцію Split( ) з простору імен data. Наприклад, розділення датасету на навчальний (80%) та тестувальний (20%) буде виглядати так:

```
mat trainDataset, testDataset;
Row<size_t> trainLabels, testLabels;
data::Split(dataset, labels, trainDataset, testDataset, trainLabels, testLabels, 0.2);
```

5. Виконання основних перетворень з даними, наприклад, навчання класифікатору, побудова моделей регресії або подібне.

6. Збереження отриманої обчислювальної моделі (наприклад, натренованого класифікатору) у файл певного формату (варіанти залежать від бібліотеки, яка використовується – файл може бути в бінарному форматі або в певному текстовому, наприклад, в XML-форматі).

7. Перевірка точності отриманої моделі на тестових даних та розрахунок певних метрик (наприклад, метрик класифікації).

8. Внесення за потреби змін в гіперпараметри моделі та повторення певних пунктів від 4 до 7.

Після створення обчислювальної моделі її залучають до процесу обробки основних даних, що подібні навчальному та тренувальному піднаборам.

Набір даних Covertype міститься у форматованому текстовому файлі. Матрицю ознак *X* (samples) складають рядки із всіма до передостаннього стовпця. Розмір матриці ознак 581 012 рядків та 54 стовпця. Останній стовпець у файлі даних – код певного класу даних. Його значення складають цільовий вектор-стовпець *y* (target). Значення в *y* представляють коди атрибута Cover_Type від 1 до 7, які описують тип лісової рослинності: Spruce/Fir (ялина, 1), Lodgepole Pine (сосна Лоджпол, 2), Ponderosa Pine (сосна Пондероса, 3), Cottonwood/Willow (тополя/верба, 4), Aspen (осика, 5), Douglas-fir (Дугласія тисолиста, 6), Krummholz (Крумхольц, 7) [5].

У праці [1] зазначена конфігурація обчислювальної апаратури, на базі якої проводився порівняльний аналіз результатів експериментів: процесор AMD Phenom II X6 1100T 3,3 ГГц, ОЗП: 8 Гб.

В поточному дослідженні було виконано порівняння швидкодії виконання програмного коду на базі mlpack та Python-коду із scikit-learn на процесорі AMD Ryzen 7 Pro 7735U 2,7 – 4,5 Ггц (8 ядер, 16 потоків), ОЗП: 32 Гб. Програмний код реалізовував *k-NN* пошук для множини з 581 012 вимірювань за сімома класами з Cover_Type.

Для виконання реалізації порівняння в mlpack був залучений клас-шаблон mlpack::NSModel з параметром типу mlpack::NearestNeighborSort. При побудові моделі вказується алгоритм, який буде впливати на пошук. Для сумісності за замовченням з подібною функціональністю із scikit-learn, вказувався алгоритм на

базі *k*-вимірного дерева (k-d tree, використаний параметр mlpack::NSModel::TreeTypes::KD_TREE). Далі відбувалася побудова моделі з гіперпараметром $k = 3$. Для сумісності із алгоритмом scikit-learn, додатково вказувався параметр для алгоритму на базі *k*-вимірного дерева – leaf_size = 15. Слід зазначити, що документація по mlpack не містить детального опису подібних кроків. Розробники радять використовувати приклади коду, що надаються в репозиторії проєкту. В ході дослідження для побудови цієї частини програми був використаний програмний код, який побудований на основі прикладів авторів mlpack з частини тестування проєкту. Наприклад (використані всі вимірювання датасету без розбиття на піднабори):

```
. . .
#include <mlpack.hpp>
#include <mlpack/core.hpp>
#include <mlpack/methods/neighbor_search.hpp>
#include <mlpack/methods/neighbor_search/ns_model.hpp>
#include <mlpack/methods/neighbor_search/neighbor_search.hpp>
. . .
using namespace arma;
using namespace mlpack;
using namespace mlpack::tree;
. . .
using KNNModel = NSModel<NearestNeighborSort>;
util::Timers timers;
KNNModel* knn = new KNNModel(KNNModel::TreeTypes::KD_TREE, false);
knn->LeafSize( ) = 15;
arma::mat refTrainDataset(dataset);
knn->BuildModel(timers, std::move(refTrainDataset), SINGLE_TREE_MODE, 0.05);

arma::Mat<size_t> resultingNeighbors;
arma::mat resultingDistances;
arma::mat refTestDataset(dataset);
double time_start = (double)clock( );
knn->Search(timers, std::move(refTestDataset), 3, resultingNeighbors, resultingDistances);
double time_stop = (double)clock( );
delete knn;
. . .
```

В програмному коді мовою Python використовувався клас sklearn.neighbors.NearestNeighbors бібліотеки scikit-learn, який реалізує навчання без вчителя для реалізації пошуку найближчих сусідів.

У підсумку мали наступний час виконання *k-NN* пошуку на зазначеній вище апаратній конфігурації на базі процесору AMD Ryzen 7 Pro 7735U та множині даних з Covertype (кількість вимірювань в пошуку – 581 012):

– mlpack: 2,6 секунди;

– scikit-learn: 12,5 секунд.

Таким чином, бачимо значне покращення в працездатності реалізації *k-NN* пошуку в scikit-learn та досить швидку роботу алгоритму в mlpack. Розрив за часом виконання зменшився в порівнянні із даними дослідження [1]. Зменшення розриву в часі виконання явно визвано не тільки більшою працездатністю процесору AMD Ryzen 7 Pro 7735U. Відмітимо, що за даними PassMark Software показник Single Thread Rating для цих моделей процесорів складає:

– AMD Phenom II X6 1100T: 1500;

– AMD Ryzen 7 Pro 7735U: 3127.

Показник Multithread Rating складає:

– AMD Phenom II X6 1100T: 3923;

– AMD Ryzen 7 Pro 7735U: 18698.

Таким чином за даними PassMark Software за показником CPU Mark Rating модель процесора AMD Ryzen 7 Pro 7735U по потужності випереджає AMD Phenom II X6 1100T в ~4,7 рази. Але різниця у виконанні програми на базі scikit-learn значно покращилася – стала скоріше в ~52 рази!

Продовженням порівняння бібліотек стало використання реальних даних мультиспектральної зйомки з апарату ДЗЗ Sentinel-2A та класифікатору, що використовує алгоритм Random Forest.

В якості вхідних даних використаний фрагмент датасету [7] для задачі класифікації території Південної Африки на базі наземних вимірювань, що є у відкритому доступі. Датасет містить 4366 полігони у системі координат проєкції EPSG:32734 (WGS 84 / UTM zone 34S). Полігональні ділянки були завірені дослідниками роботи [7] протягом сезону врожайності на початку 2017 року. Як зазначається в [7], датасет містить зразки вимірювань на місцевості, які використовуються для створення маски сільськогосподарських угідь та карти типів культур Південної Африки. Ці дані збиралися за принципами ініціативи JECAM (Joint Experiment for Crop Assessment and Monitoring) під проєкт Sen2-Agri (Sentinel-2 for Agriculture), що фінансується ESA (European Space Agency). Акцентується, що дані збираються у вигляді не точок, а саме полігонів. Датасет

розповсюджується у векторному форматі ESRI Shapefile, який є фактично стандартом де-факто у світі геоінформаційних систем (ГІС). Полігони датасету географічно покривають певні ділянки частини території Південної Африки, яка також покривається двома тайлами мультиспектральних зйомок з супутників серії Sentinel-2 – тайли 34HCH та 34HDH.

Для експерименту по дослідженню роботи бібліотеки mlpack та порівняння її можливостей із scikit-learn, датасет був обмежений тільки границями тайлу 34HCH та став містити 2012 полігонів. Нехай множина цих полігонів, що потрапили у межі тайлу 34HCH, має позначення *P*.

Згідно атрибуту LC (Land Cover) з таблиці атрибутів *P*, полігональні ділянки відповідали таким типам земного покриття:

– Shrub land (чагарникові угіддя);
– Water bodies (водойми);
– Forest (ліси);
– Build-up surface (забудовані території);
– Bare soil (гола земля);
– Perennial crops (багаторічні культури);
– Barley (ячмінь);
– Oilseed crop (олійні культури);
– Wheat (пшениця);
– Grasses & other fodder crops (трави та інші кормові культури);
– Fallows (пари);
– Lupins (люпин);
– Grassland & meadows (луки та пасовища).

В якості мультиспектрального космознімку в досліджені був обраний продукт зйомки з супутника Sentinel-2A рівня обробки L1C за тайлом 34HCH:

S2A_MSIL1C_20170809T081601_N0205_R121_T34HCH_20170809T084800

Продукт доступний для завантаження зі сховища Google Cloud Storage за певними правилами доступу. Через центральний сервіс CDSE (Copernicus Data Space Ecosystem) доступні оновлені версії продукту за вказану дату та час зйомки.

На рис. 1 представлені межі зйомки та зменшене композитне кольорове зображення, яке сформоване за даними каналів 4-3-2 (RGB) мультиспектрального сканера MSI (Multi-Spectral Instrument) [8] супутника Sentinel-2A. Зйомка території виконана 09 серпня 2017 року та відповідає року завершення формування *P*.

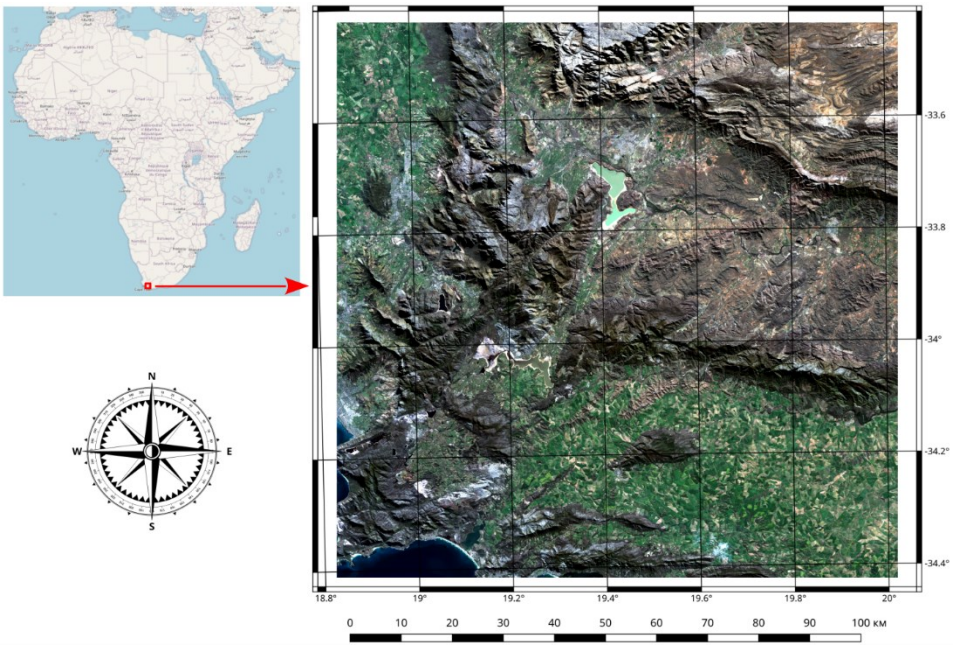


Рис. 1. Композитне RGB-зображення території дослідження за даними мультиспектрального сканера MSI супутника Sentinel-2A. Дата зйомки: 09.08.2017

Подальший процес дослідження складався з двох великих частин:

– підготовки датасету для навчання класифікатору та попереднього аналізу майбутніх класів;

– розробки робочих прототипів програм для проведення тренування класифікатору та отримання карт класифікації території дослідження за визначеними класами.

В якості початкових ознак даних для проведення класифікації, обрані значення в пікселях зображень мультиспектральних каналів зйомки: NIR (Near-infrared, ближній інфрачервоний, зображення $I_N(x,y)$), Red (червоний, зображення $I_R(x,y)$), Green (зелений, зображення $I_G(x,y)$) та Blue (синій, зображення $I_B(x,y)$). Дані каналів представлені з просторовим розрізненням 10 метрів та в подальшому були сформовані у вигляді

єдиного чотирьохканального зображення (стекованого n-канального зображення *I(x,y,n)*) формату GeoTIFF з проведенням нормалізації за документацією ESA Sentinel-2 в одиниці спектрального відбиття у верхніх шарах атмосфери (top-of-atmosphere (TOA) spectral reflectance): всі значення були розділені на 10 000. Розмір зображення 10 980 x 10 980 піксель. Загальна кількість пікселів у початковому чотирьохканальному зображені склала 482 241 600.

Для подальшої підготовки датасету, з *I(x,y,n)* маскуванням по полігональним ділянкам множини *P* були отримані значення пікселів з N, R, G та B каналів та сформований окремий датасет *A*, що містив 2 983 731 вимірювань по відповідним ознакам.

Наступним кроком став процес отримання діаграм розсіювання даних з *A* для пар різних каналів за ознаками N, R, G та B. Для візуалізації діаграм розсіювання (рис. 2) була використана Python-бібліотека seaborn, яка побудована на базі іншої відомої бібліотеки Matplotlib. Для скорочення часу візуалізації, *A* була проріджена до 1300 вимірювань (взяті перші 100 вимірювань з кожного класу). Назви класів відповідають переліченим вище типам земного покриття території зйомки.

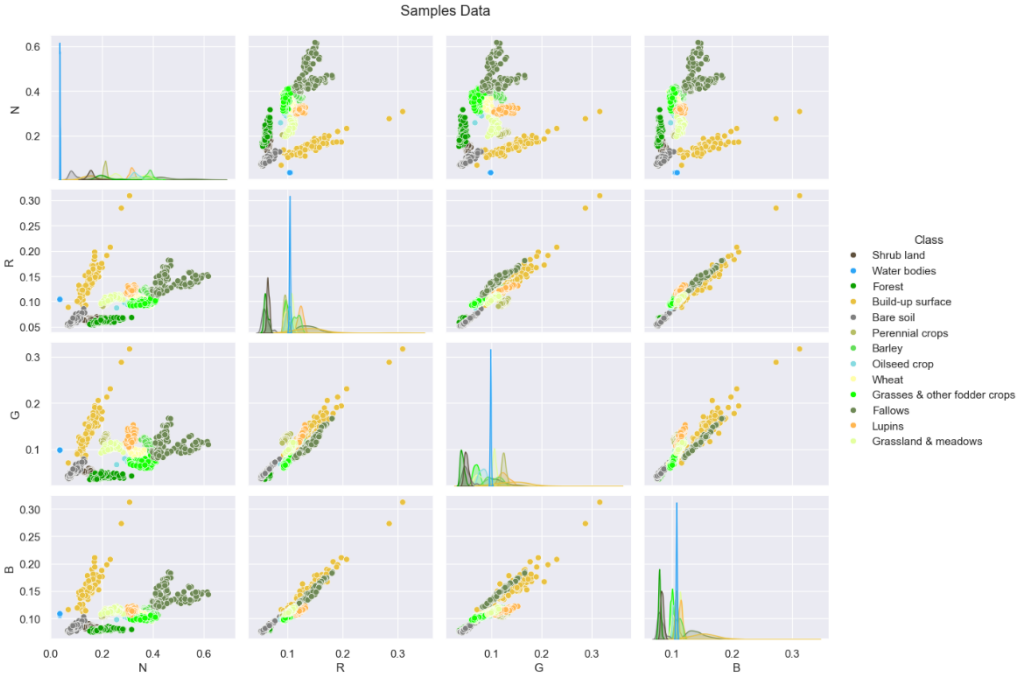


Рис. 2. Діаграми розсіювання для даних з різних класів, що отримані за полігонами з роботи [7] та продукту рівня обробки L1C мультиспектральної зйомки Sentinel-2A за тайлом 34HCH (дата зйомки: 09.08.2017)

Як бачимо з рис. 2, багато вимірювань з різних класів є досить сильно корельовані та перетинаються у просторі ознак N, R, G, B.

Для поліпшення віднесення вимірювань до певних класів, над *A* були виконані такі кроки.

1. Кількість вимірювань знов скорочена – обрано кожне 4-те вимірювання з *A*. Це дозволило скоротити множину з 2 983 731 до 745 933 для пришвидшення подальших операцій попереднього аналізу множини даних.

2. Оскільки аналіз проводився тільки за одним космознімком в період великої вегетації рослинності, то для порівняння роботи класифікаторів з різних бібліотек, дані класів Perennial crops, Barley, Oilseed crop, Wheat, Grasses & other fodder crops, Fallows, Lupins та Grassland & meadows були об'єднані в єдиний клас Vegetation (рослинність).

3. За даними трьох зображень $I_N(x,y)$, $I_R(x,y)$, $I_G(x,y)$, створені додаткові зображення *ind1(x,y)*, *ind2(x,y)*, *ind3(x,y)*, що представляли розраховані мультиспектральні індекси [9] відповідно: NDVI (Normalized Difference Vegetation Index, (N-R)/(N+R)), NDWI (Normalized Difference Water Index, (G-N)/(G+N)) та зворотній Red-Green індекс (simple ratio G/R).

4. За даними семи зображень $I_N(x,y)$, $I_R(x,y)$, $I_G(x,y)$, $I_B(x,y)$, *ind1(x,y)*, *ind2(x,y)*, *ind3(x,y)*, методом головних компонент (PCA – Principal Component Analysis) отримані нові додаткові зображення *pc1(x,y)* та *pc2(x,y)*, що містили множини даних без лінійної кореляції – результати розрахунку головних компонент PC1 та PC2.

5. Проведено об'єднання в оновлений стек мультиспектрального зображення *I(x,y,n)* шести зображень: $I_N(x,y)$, $I_R(x,y)$, $I_G(x,y)$, $I_B(x,y)$, *pc1(x,y)*, *pc2(x,y)*.

6. Отримано оновлену множину *X* із 745 933 вимірювань за 6-ти класами та із 6-ти ознаками: N, R, G, B, PC1, PC2.

На рис. 3 представлена узагальнена схема передобробки та підготовки датасету до проведення класифікації. Діаграми розсіювання для кінцевої *X* по ознакам PC1 та PC2 представлені на рис. 4. Для скорочення часу візуалізації та поліпшення сприйняття відображення були взяті перші 100 вимірювань з кожного класу.

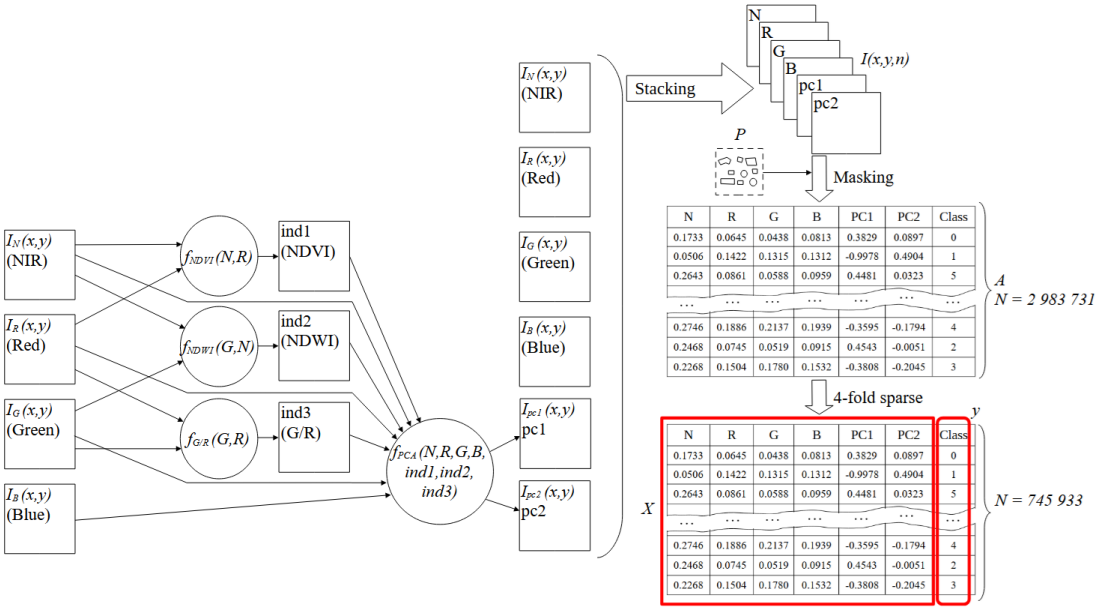


Рис. 3. Схема процесу отримання датасету для навчання класифікатору

Таким чином з рис. 4 бачимо, що додаткові ознаки – головні компоненти PC1 та PC2 значно покращують розрізнення вимірювань, які мають відношення до вказаних класів.

З отриманого датасету X 80% даних були відокремлені для навчання класифікатору Random Forest, а інші використані для тестування створеної моделі класифікатору. Процедури навчання класифікатору та тестування навчаної моделі проводилися для порівняння за допомогою створеного програмного коду на базі mlpack та окремо scikit-learn. На рис. 5 представлені результати цього процесу. Гіперпараметрами для класифікатору по 6-ти класам виступали: кількість дерев в лісі – 10, мінімальний розмір листа – 1, максимальна глибина пошуку – 15. Багатопоточність була увімкнена.

Таким чином, за отриманими результатами по часу на навчання та часу на класифікацію лідером на платформі з процесором AMD Ryzen 7 Pro 7735U була програма на базі scikit-learn. Загалом можна зробити висновок, що реалізація класифікатору Random Forest в mlpack поки що уступає за часом виконання реалізації із scikit-learn.

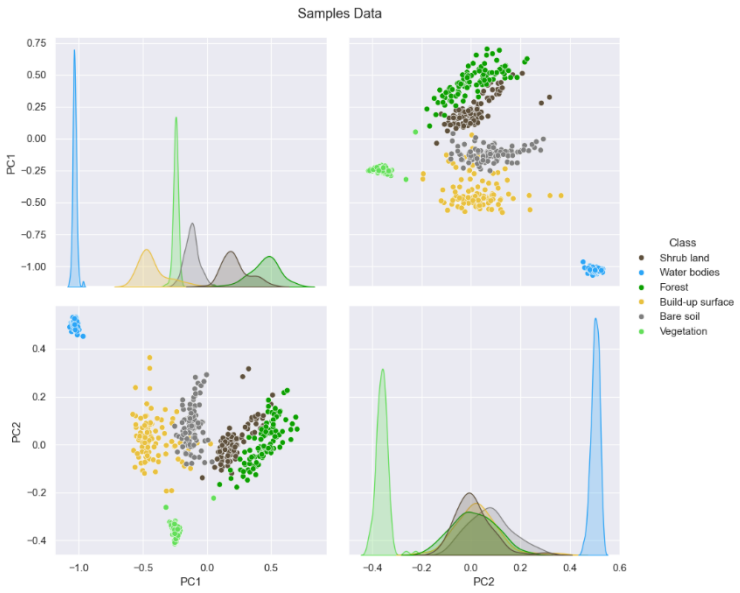
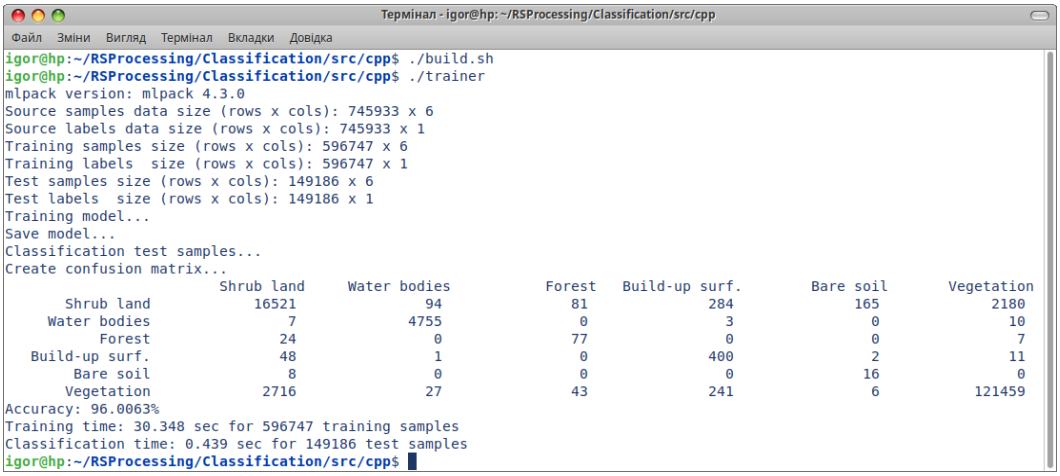


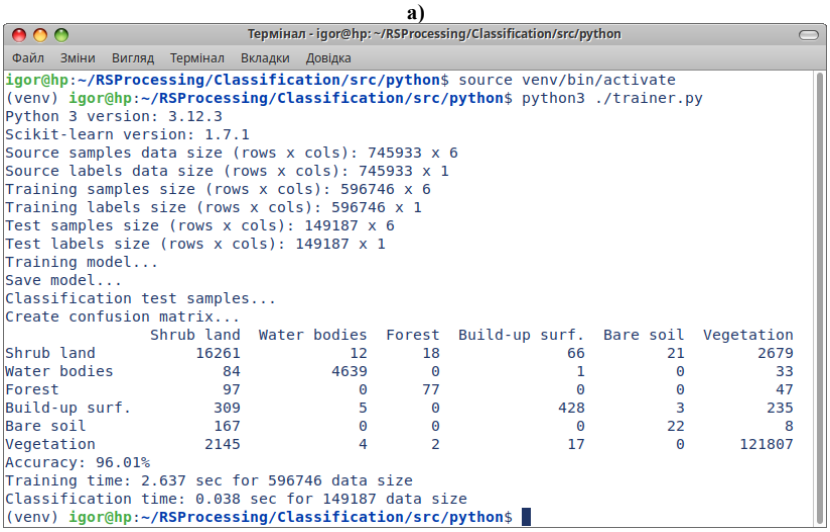
Рис. 4. Діаграми розсіювання для даних шести класів за розрахованими ознаками PC1 та PC2

Точність класифікації тестових даних склала ~96% та є досить високою в обох програмах.

Після підготовки моделей класифікатора Random Forest, розроблені програмні рішення на базі бібліотек GDAL (Geospatial Data Abstraction Library) і mlpack мовою програмування C++ та пакетів numpy і rasterio мовою програмування Python, які дозволили отримати результат класифікації багатоканального космознімку із супутника Sentinel-2A території дослідження (рис. 1) за навченими моделями. Одна з карт класифікації представлена на рис. 6.



а)



б)

Рис. 5. Результати навчання класифікатору Random Forest та тестування його моделі в GNU/Linux-сумісній ОС на процесорі AMD Ryzen 7 Pro 7735U: а) – реалізація на mlpack; б) – реалізація на scikit-learn

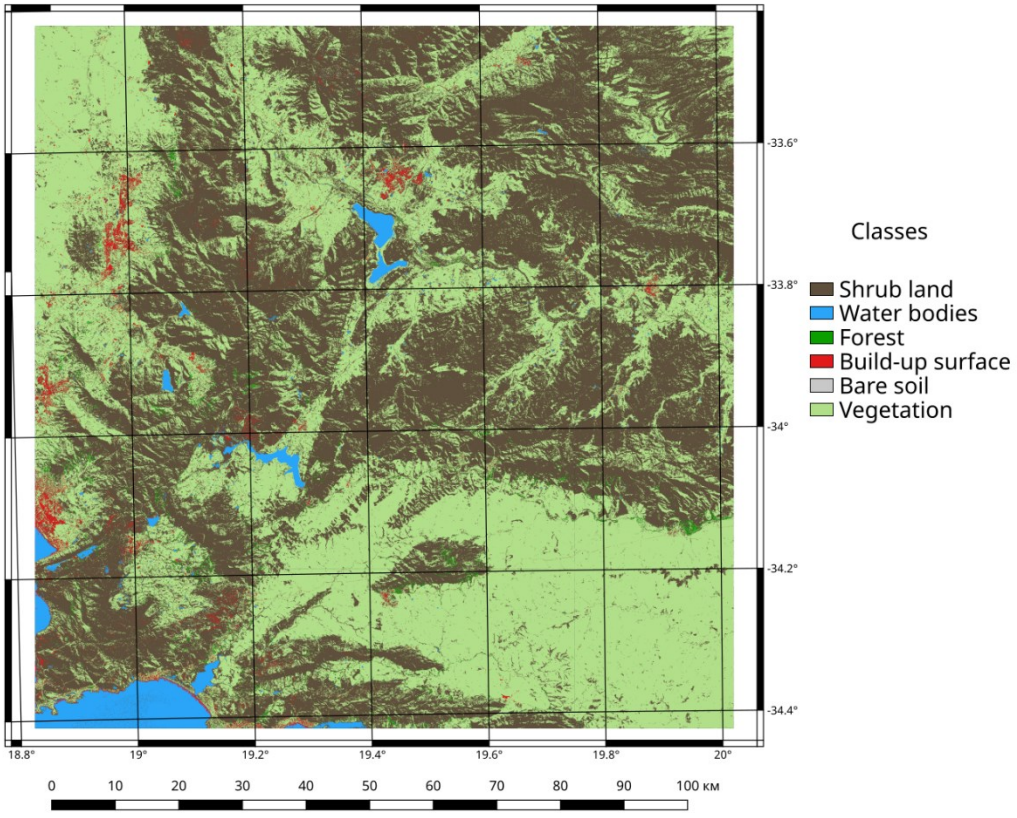


Рис. 6. Карта класифікації типів земного покриття території дослідження

Бібліотеки GDAL (в програмі мовою C++) та rasterio (в програмі мовою Python) були залучені для виконання задачі читання 6-тиканального геоприв'язаного зображення *I(x,y,n)* та задачі збереження карти класифікації. Для більшої адекватності порівняння часу класифікації між mlpack та scikit-learn, дані всіх каналів були повністю завантажені в пам'ять. Операції введення/виведення не враховувалися під час порівняння та опрацьовувалися значення вже у підготовлених комірках пам'яті.

Загальна кількість вимірювань для класифікатору Random Forest, кожне з яких має значення за 6-ти ознаками (N, R, G, B, PC1, PC2), становила 120 560 400. Ця кількість вимірювань і є кількістю пікселів растрової геоприв'язаної карти класифікації (рис. 6).

Час, витрачений на класифікацію Random Forest за навчаними моделями на обладнанні із центральним процесором AMD Ryzen 7 Pro 7735U склав:

– mlpack – 163,726 секунди (~2,7 хвилини) при виконанні в одному потоці;

– mlpack – 26,2 секунди при виконанні в багатопоточному режимі (16 потоків);

– scikit-learn – 72,457 секунди (~1,2 хвилини) при виконанні в одному потоці;

– scikit-learn – 143,071 секунди (~2,4 хвилини) при виконанні в багатопоточному режимі (16 потоків).

Для використання mlpack в багатопоточному режимі розробники радять використовувати при збірці власних програм опцію увімкнення підтримки OpenMP (Open Multi-Processing). У випадку із scikit-learn при визначені гіперпараметрів в конструкторі класу RandomForestClassifier є відповідний параметр n_jobs, який контролює використання кількості ядер процесора під час виконання навчання моделі й подальшої класифікації.

Висновки

В процесі дослідження можливостей використання бібліотеки mlpack та порівняння її швидкості із бібліотекою scikit-learn можна зробити висновок, що дійсно при певних умовах налаштування та на певних алгоритмах є певний виграш у швидкодії але подібне потребує окремих експериментів при використанні певних методів обробки чи класифікації. *k-NN* пошук, на який опиралися автори mlpack у своїх порівняннях, дійсно значно виграє на множині з датасету Covertype. Але, наприклад, використаний алгоритм Random Forest на великій кількості вимірювань (120 560 400) довів, що є доречним використання додаткової опції компіляції програми для залучення підтримки OpenMP-можливостей. Тобто у звичайному однопоточному режимі mlpack (версії 4.3.0) програє у швидкості реалізованому класифікатору Random Forest із пакунку scikit-learn. Несподіванкою, але фактом, виявилося зниження працездатності scikit-learn при використання багатопоточної моделі – приблизно вдвічі більше було витрачено часу на класифікацію бакатоканального зображення.

При порівнянні результатів класифікації (побудованих карт) з'ясувалося, що карти класифікації співпадають на 96%. Тобто можна казати, що результати класифікації з mlpack та scikit-learn за отриманими моделями навчання досить схожі. Як показали матриці невідповідності результатів класифікації тестових даних, більшість хибних віднесень було у класів Shrub land (чагарникові угіддя) та Vegetation (рослинність).

Для підвищення точності побудови подібних карт класифікації та більш чіткого розділення даних за класами у просторі ознак, фахівці радять використовувати набори різночасових мультиспектральних космознімків. В цьому дослідженні це не було самоціллю, оскільки досліджувалася ефективність залучення до побудови програм з обробки мультиспектральних даних бібліотеки mlpack та її певних можливостей.

Загалом можна відмітити, що бібліотека mlpack виступає гідним конкурентом по відношенню до бібліотеки scikit-learn та може бути ефективно використана для обробки, наприклад, великих об'ємів даних ДЗЗ мовою програмування C++.

Література

1. Curtin, R. R. MLPACK: A scalable C++ machine learning library / Curtin, R. R., Cline, J. R., Slagle, N. P., March, W. B., Ram, P., Mehta, N. A., & Gray, A.G. // The Journal of Machine Learning Research. – 2013. – Volume 14 – P. 801–805.

2. Conrad Sanderson, Ryan Curtin. Armadillo: An Efficient Framework for Numerical Linear Algebra // International Conference on Computer and Automation Engineering. – 2025. – P. 303–307. – DOI: https://doi.org/10.48550/arXiv.2502.03000.

3. Conrad Sanderson, Ryan Curtin. Practical Sparse Matrices in C++ with Hybrid Storage and Template-Based Expression Optimisation // Mathematical and Computational Applications. – 2019. – Volume 24, Issue 3, 70. – DOI: https://doi.org/10.3390/mca24030070.

4. Markelle Kelly, Rachel Longjohn, Kolby Nottingham. The UCI Machine Learning Repository [Електронний ресурс]. – Режим доступу: https://archive.ics.uci.edu.

5. Blackard, J. Covertype [Dataset]. UCI Machine Learning Repository [Електронний ресурс]. – 1998. – DOI: https://doi.org/10.24432/C50K5N.

6. Breiman, L. Random forests // Machine Learning. – 2001. – Volume 45. – P. 5–32. – DOI: https://doi.org/10.1023/A:1010933404324.

7. Agricultural Research Council (South Africa). In-situ data for supervised classification and product validation in Sen2-Agri over the South Africa area of interest for the 2017 growing season [Електронний ресурс] : [Dataset]. – GEO Knowledge Hub. – 2020. – Version 1.0. – Режим доступу: https://gkhub.earthobservations.org/records/5psrj-0rs82. – (Дата звернення 17.06.2025). – Назва з екрана.

8. Sentinel-2 Products Specification Document. – ESA, Thales Alenia Space, TAS team. – 2021. – REF: S2-PDGS-TAS-DI-PSD, Issue 14.9, 30/09/2021. – 552 p.

9. Index DataBase. A database for remote sensing indices [Електронний ресурс] : [веб-сайт]. – Режим доступу: https://www.indexdatabase.de/ – (Дата звернення 20.06.2025). – Назва з екрана.

References

1. Curtin, R. R. MLPACK: A scalable C++ machine learning library / Curtin, R. R., Cline, J. R., Slagle, N. P., March, W. B., Ram, P., Mehta, N. A., & Gray, A.G. // The Journal of Machine Learning Research. – 2013. – Volume 14 – P. 801–805.

2. Conrad Sanderson, Ryan Curtin. Armadillo: An Efficient Framework for Numerical Linear Algebra // International Conference on Computer and Automation Engineering. – 2025. – P. 303–307. – DOI: https://doi.org/10.48550/arXiv.2502.03000.

3. Conrad Sanderson, Ryan Curtin. Practical Sparse Matrices in C++ with Hybrid Storage and Template-Based Expression Optimisation // Mathematical and Computational Applications. – 2019. – Volume 24, Issue 3, 70. – DOI: https://doi.org/10.3390/mca24030070.

4. Markelle Kelly, Rachel Longjohn, Kolby Nottingham. The UCI Machine Learning Repository [Elektronnyi resurs]. – Rezhym dostupu: https://archive.ics.uci.edu.

5. Blackard, J. Covertype [Dataset]. UCI Machine Learning Repository [Elektronnyi resurs]. – 1998. – DOI: https://doi.org/10.24432/C50K5N.

6. Breiman, L. Random forests // Machine Learning. – 2001. – Volume 45. – P. 5–32. – DOI: https://doi.org/10.1023/A:1010933404324.

7. Agricultural Research Council (South Africa). In-situ data for supervised classification and product validation in Sen2-Agri over the South Africa area of interest for the 2017 growing season [Elektronnyi resurs] : [Dataset]. – GEO Knowledge Hub. – 2020. – Version 1.0. – Rezhym dostupu: https://gkhub.earthobservations.org/records/5psrj-0rs82. – (Data zvernennia 17.06.2025). – Nazva z ekrana.

8. Sentinel-2 Products Specification Document. – ESA, Thales Alenia Space, TAS team. – 2021. – REF: S2-PDGS-TAS-DI-PSD, Issue 14.9, 30/09/2021. – 552 p.

9. Index DataBase. A database for remote sensing indices [Elektronnyi resurs] : [veb-sait]. – Rezhym dostupu: https://www.indexdatabase.de/ – (Data zvernennia 20.06.2025). – Nazva z ekrana.