Vol. 6, No. 1, 2024

Олексій Веретюк[1], Назарій Андрущак[2]

[1] Кафедра систем автоматизованого проектування, Національний університет Львівська політехніка, вул. С. Бандери 12, Львів, Україна, E-mail:oleksii.v.veretiuk@lpnu.ua, ORCID 0009-0009-2340-4458
[2] Кафедра систем автоматизованого проектування, Національний університет "Львівська політехніка", вул. С. Бандери, Львів, Україна, E-mail:nazariy.a.andrushchak@lpnu.ua, ORCID 0000-0002-8248-404X

МЕТОДИ ТА МОДЕЛІ МАШИННОГО НАВЧАННЯ В ХІМІЇ ТА МАТЕРІАЛОЗНАВСТВІ НА ПРИКЛАДІ ЕКСПЕРИМЕНТУ З ДИФУЗІЄЮ РОЗЧИНЕНОЇ РЕЧОВИНИ

Отримано: березень 12, 2024 / Переглянуто: березень 20, 2024 / Прийнято: квітень 01, 2024

https://doi.rg/

Анотація: Машинне навчання є логічним продовженням автоматизованих процесів за допомогою обчислювальних систем. В той час як велика кількість різних сфер діяльності людини булипокращені створенням програмного забезпечення з використанням алгоритмічнихпідходів, велика кількість інших задач залишається не вирішеною, адже створення алгоритмів для них є майже неможливим. До таких сфер можна віднести хімічні та фізичні дослідження. Емпіричний підхід все ще є одним з найважливіших підходів у досягненні результатів, адже для багатьох досліджень все ще не існує чіткого математичного апарату. Машинне навчання є тим рішенням, яке дозволяє заощадити ресурси й пришвидшити процес дослідження. Проведення експериментів це завжди збір даних про результати. Алгоритми машинного навчання дозволяють використати цю інформацію для побудови моделі здатної передбачати результати експериментів або властивості нових сполук. В межах цієї статті, на прикладі даних отриманих з експериментів з дифузією розчиненою речовини, перевіряється ефективність низки алгоритмів, як стандартних так й ансамблевих алгоритмів, з обмеженоюкількістю вхідних даних. В результаті були отримані дані про ефективність запропонованих алгоритмів, які буливизначені за допомогою формул середньоквадратичної похибки, а також середньої абсолютної відсоткової похибки. Наведені приклад й опис процесу побудови моделей машинного навчання різного типу.

Ключові слова: машинне навчання, scikit-learn, хімія, матеріалознавство, обмежена кількість тренувальних даних, дифузія розчиненої речовини

Вступ

Останні роки використання машинного навчання відіграє все більш важливу роль в наукових дослідженнях, зокрема у хімії та матеріалознавстві [1] для вирішення різноманітних задач, таких як визначення властивостей елементів та сполук [2, 3], визначення властивості кристалізації сполуки [4], автоматичне опрацювання результатів експерименту й передбачення його результату [5]. Одне з ключових застосувань машинного навчання в цих галузях є відкриття нових хімічних сполук та матеріалів. Зокрема, генерації молекулярних структур та інших властивостей елементів. На відміну від підходів *ab-initio* моделювання [6], які дозволяють визначення структур майбутніх кристалів на основі перших принципів без використання експериментального введення, проте вимагають великих обчислювальних потужностей на суперкомп'ютерах (обчислення може тривати місяцями), алгоритми машинного навчання можуть покращити точність й ефективність експериментів та емпіричних досліджень, використовуючи результати минулих досліджень в якості тренувальних

даних. Такий підхід дозволяє заощадити велику кількість як матеріальних так і часових ресурсів, адже дослідження потенційних властивостей великої кількості сполук, наприклад можливості кристалізуватись, вимагає використання значної кількості обчислювальних ресурсів. В той самий час як натренована модель машинного навчання здатна обробити тисячі або десятки тисяч різних сполук в короткий термін.

Проте машинне навчання не є ідеальним інструментом, цей підхід вимагає великої кількості даних для того щоб відсоток вдалого прогнозування був достатньо великим для ефективного застосування моделей та алгоритмів. Це є великою перепоною для застосування машинного навчання, адже не кожне дослідження чи експеримент має велику кількість даних для побудови ефективної моделі.

Методи та моделі машинного навчання об'єднані в межах різних бібліотек, які доступні у вигляді відкритого коду. Популярними на сьогодні бібліотеками є scikit-learn, keras, tensorflow. Бібліотека scikit-learn є однією з найпопулярніших відкритих бібліотек машинного навчання для Python, пропонуючи різноманітні інструменти для класифікації, регресії, кластеризації, а також інших завдань машинного навчання. Вона містить ефективні реалізації багатьох алгоритмів навчання, таких як метод опорних векторів, дерева рішень, випадкові ліси та багато інших. Бібліотека також надає інструменти для попередньої обробки даних, відбору ознак та оцінки моделей. Keras– це високорівневий інтерфейс для побудови та навчання нейронних мереж у мові програмування Python. Він надає зручні інструменти для швидкого створення, налаштування та навчання різноманітних моделей глибокого навчання. TensorFlow– це відкрита платформа для розробки та навчання моделей глибокого навчання та штучного інтелекту. Вона надає потужні інструменти для створення складних нейронних мереж, обробки великих обсягів даних та вирішення різноманітних завдань машинного навчання.

Також важливою допоміжною бібліотекою є RDKit, яка дає доступ до різних операцій й обчислень в галузі хімії, що може бути корисним при підготовці даних, на яких буде тренуватись модель.

Прикладом задачі, яка може бути вирішена за допомогою цих інструментів є передбачення ефективної енергії активації дифузії в межах експериментів з дифузією розчиненою речовини. На основі відкритого набору даних можна побудувати моделі машинного навчання, з використанням різних алгоритмів та порівняти їх ефективність. Також крім стандартних алгоритмів існує низка ансамблевих алгоритмів, моделі яких теж були побудовані й оцінені.

Постановка проблеми

Незважаючи на те, що для того, щоб отримати натреновану модель машинного навчання й почати її використовувати достатньо імпортувати декілька модулів з бібліотеки, основна проблема лежить в тому, як зробити цю модель ефективною. Основними проблемами є розуміння різних алгоритмів машинного навчання, їхніх відмінностей й специфіки для того, щоб вибрати найоптимальніший алгоритм для вирішення відповідного роду проблеми. Іншим викликом є дані, які використовуються, як для тренування моделі так й для подальшої валідації.

Машинне навчання це підхід, який повністю залежить від наявності коректних даних в великій кількості, адже конфліктні дані або їх недостатня кількість призведе до того, що модель не буде працювати консистентно, а її прогнозування будуть не вдалими. Варто сказати, що основною проблемою в цьому випадку зазвичай виступає кількість даних. Отже, це цілком прогнозовано, що треба буде працювати з малою відносно потреб алгоритмів кількістю даних. Проте є підходи, які дозволяють підвищити ефективність моделей навіть при недостатній кількості даних.

Іншою проблемою є те, що не існує універсального підходу для обрання алгоритму побудови моделі. Основним способом вибору є тренування декількох моделей на різних алгоритмах й вибір відповідно до результатів передбачень на тестових даних.

Виклад основного матеріалу

Машинне навчання – це галузь штучного інтелекту, яка зосереджена на створенні статистичних алгоритмів, які можуть автоматично навчатися з даних й узагальнювати їх без явних інструкцій. Покращення ефективності таких алгоритмів відбувається відповідно до надання нових даних, які дозволяють робити передбачення більш точними, тобто без додаткових інструкцій та змін в програмному забезпеченні. Існують декілька базових типів машинного навчання: з вчителем, без вчителя й навчання з підкріпленням. Навчання з вчителем –викликає найбільший інтерес. Цей тип навчання представляє собою математичну модель з визначеними входами та бажаними виходами, тобто вимагаються такі дані, в яких є як змінні, які мають бути проаналізовані, так і результат, який виходить з значень або поєднання цих змінних. Це є чудовим рішенням для експериментальних наук, як хімія та матеріалознавство, адже є як інформація про входи: параметри середовища, типи сполук, так й виходи у вигляді результату експерименту.

До основних алгоритмів навчання з вчителем, які можуть використовуватись як для регресії так й для класифікації, окрім алгоритму лінійної регресії, який є тільки регресійним, можна віднестилінійну регресію [7], метод найближчих сусідів (k-NearestNeighbors)[8], метод опорних векторів (SupportVectorMachines)[9], навчання на основі дерев рішень (DecisionTrees)[10], нейронні мережі (NeuralNetworks)[11].

Лінійна регресія – найпоширеніший метод машинного навчання для прогнозування значення змінної на основі інших змінних, використовуючи лінійну функцію.

Метод найближчих сусідів –алгоритм машинного навчання та регресії. Алгоритм базується на принципі схожості об’єктів. Коли використовується як регресійни алгоритм, результат базується на значенні середнього арифметичного та медіани бажаної змінної, а у випадку класифікації використовується мода.

Метод опорних векторів –інший алгоритм, який може використовуватись для класифікації та регресії. Особливістю цього алгоритму, що він добре працює зі складними даними в малих кількостях. Цей алгоритм базується на пошуку такої гіперплощини, яка найкраще розділяє класи, тобто відстань від площини до класів має бути якнайбільша.

Дерево рішень – цей алгоритм базується на принципі побудови дерева, де вузол є проміжним рішенням й та відповідає за вирішення того, який шлях буде обраний наступний відносно значення, яке було передано у вузол, або термінальним рішенням, яке представляє фінальне передбачення.

Нейронні мережі –це концепція, яка наслідує поведінку людського мозку при прийнятті рішень. Цей підхід представляє собою шари нейронів, які пов’язані між собою, й складаються з вхідного шару, захованого шару й вихідного шару. Вхідний шар складається з вузлів, кількість яких відповідає кількості ознак у даних, ці вузли пов’язанізв’язками з вузлами наступного шару. Кожен зв’язок має відповідну вагу, яка визначає важливість вхідного сигналу для конкретного нейрона. Кожен нейрон має в собі певний обчислювальний апарат, який має видати значення відповідно до величин вхідних сигналів, яке буде застосовано для активації наступного нейрона, якщо значення попередніх відповідають умові активації. Цей процес продовжується допоки черга не дійде до вихідного шару, який має згенерувати фінальний прогноз.

Реалізувати алгоритм можна на будь якій мові програмування, проте сьогодні вже існує велика кількість високорівневих бібліотек, які надають доступ до різних алгоритмів, моделей та засобів машинного навчання. Однією з найпопулярніших є scikit-learn написаний мовою Python.

Scikit-learn– це відкрита бібліотека для машинного навчання, яка надає широкий набір інструментів для класифікації, регресії, кластеризації, підбору параметрів, підтримки вимірювання та візуалізації результатів, а також для роботи з наборами даних.

В якості даних для тренування моделі машинного навчання були взяті відкриті дані з експериментів з дифузією розчиненою речовини [12]. Властивостями, які використовуються для передбачення, є матеріали, з яких складається носій та розчин, а також властивості цих матеріалів, як наприклад температура плавлення розчиненого елемента. Властивість, яка має бути передбачена,

це ефективна енергія активації дифузії, тобто та енергія, яка потрібна для того, щоб атоми розчинника могли пройти через кристалічну решітку носія з одного місця в інше. Варто сказати, що цей набір даних доволі малий по мірках машинного навчання всього 408 записів. Для знаходження числового параметра треба використати регресійні алгоритми. Для вирішення задачі передбачення енергії активації дифузії були обрані такі алгоритми: лінійна регресія, метод найближчого сусіда, метод опорних векторів з трьома різними ядрами, кожен з яких представляють певну математичну функцію: лінійну, поліноміальну та радіально базисну, а також алгоритм дерев рішень.

На лістингу 1 представлена реалізація коду, який випадковим чином розділяє дані на тестові на тренувальні, ініціалізує, тренує й оцінює моделі за допомогою формули середньоквадратичної похибки. В цьому прикладі тренується виключно алгоритм методу опорних векторів з радіально базисною функцією.

```
frompathlibimportPath
fromsklearn.model_selectionimporttrain_test_split
fromsklearn.metricsimportmean_squared_error
fromsklearn.svmimport SVR
importpandasaspd
dsd_data = pd.read_csv(Path("materialscience_datasets/Dilute_Solute_Diffusion_with_features.csv"))
strat_train_set, strat_test_set = train_test_split(dsd_data, test_size=0.2, random_state=42)
# Getdatawithoutlabels
train_features = strat_train_set.drop("Enorm (eV)", axis=1)
train_features = train_features.drop("E_raw (eV)", axis=1)
train_features = train_features.drop("Materialcompositions 1", axis=1)
train_features = train_features.drop("Materialcompositions 2", axis=1)
# Labels
train_e_labels = strat_train_set["Enorm (eV)"].copy()
train_eraw_labels = strat_train_set["E_raw (eV)"].copy()
# Gettestfeatureswithoutlabels
test_features = strat_test_set.drop("Enorm (eV)", axis=1)
test_features = test_features.drop("E_raw (eV)", axis=1)
test_features = test_features.drop("Materialcompositions 1", axis=1)
test_features = test_features.drop("Materialcompositions 2", axis=1)
# Labels
test_e_labels = strat_test_set["Enorm (eV)"].copy()
test_eraw_labels = strat_test_set["E_raw (eV)"].copy()
# Modelstrainingandestimating
RBFSVR_model = SVR(kernel="rbf", C=41, gamma='scale')
RBFSVR_model.fit(train_features, train_eraw_labels)
predictions = RBFSVR_model.predict(train_features)
print(f'SVR RBF RMSE: {mean_squared_error(train_eraw_labels, predictions, squared=False)}')
predictions = RBFSVR_model.predict(test_features)
print(f'SVR RBF TEST RMSE: {mean_squared_error(test_eraw_labels, predictions, squared=False)}')
```

Лістинг 1. Реалізація алгоритму методу опорних векторів

Важливим етапом створення моделі є процес налаштування алгоритмів. Для цього використовують гіперпараметри. Гіперпараметри– це параметри, значення яких використовуються при навчанні, наприклад максимальна глибина дерева. Для того щоб знайти найкращі гіперпараметри для обраних алгоритмів був використаний клас GridSearchCV, який представляє

собою пошук грубою силою крізь задані параметри з використанням крос-валідації, а також пошук параметрів за допомогою звичайного циклу. На лістингу 2 наведена реалізація пошуку найкращих параметрів.

```
fromsklearn.model_selectionimportGridSearchCV
importnumpyasnp
C_range = list(range(1, 1000))
param_grid = dict(gamma=['scale'], C=C_range)
grid = GridSearchCV(SVR(kernel='rbf'), param_grid=param_grid, cv=10, scoring='neg_root_mean_squared_error')
grid.fit(train_features, train_eraw_labels)
print(
   "Thebestparametersare %s with a scoreof %0.2f"
   % (grid.best_params_, grid.best_score_)
        )
for n inrange(1, 20):
KNearest_model = KNeighborsRegressor(n_neighbors=n)
KNearest_model.fit(train_features, train_eraw_labels)
predictions = KNearest_model.predict(train_features)
print(f'KNearest n={n} RMSE: {mean_squared_error(train_eraw_labels, predictions, squared=False)}')
```

Лістинг 2. Приклад пошуку найкращих гіперпараметрів для двох алгоритмів

Якщо узагальнити, то пошук найкращих параметрів –це перебір можливих варіантів гіперпараметрів у визначених межах. Як показано на лістингу 2 у випадку методу опорних векторів був використаний клас GridSearchCV з бібліотеки scikit-learn, який обрав найкращі параметри з наданих. Проте, ці гіперпараметри можна шукати за допомогою власних методів, наприклад для алгоритму найближчих сусідів був використаний цикл, який перебирає кількість *n_neighbors*, тобто кількість сусідів, які використовуються при передбаченні, далі достатньо просто порівняти значення похибок й обрати найкращий варіант.

Також важливим етапом в підготовці моделі є крос-валідація. Як відомо дані поділяються на тренувальні й тестові. Тестові дані використовуються лише на фінальному етапі побудови моделі й мають дати фінальну оцінку якості тренування. Також тестові дані не завжди генерують випадково з одного набору даних, дуже часто цей набір буде створюватись вручну, щоб зменшити його схожість з тренувальними даними, для найкращої оцінки ефективності. Отже, тільки тренувальні дані використовуються під час знаходження гіперпараметрів, тому є загроза того, що ці параметри будуть добре працювати лише з тренувальними даними. Процес крос-валідації лежить в розділені тренувальних даних на додаткові набори даних, одна частина, яких відповідає за тренування, а інша за тестування. Такі набори даних генеруються декілька разів на кожен варіант гіперпараметрів, що дозволяє уникнути проблеми з залежністюгіперпараметрів на тренувальнихданих.

В якості оцінки алгоритму використовується середньоквадратична похибка(*RMSE*) [13], яка може бути обчислена за допомогою формули 1:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n-1}(y_i - \hat{y}_i)^2} \tag{1}$$

де $n-$ кількість записів, $i-$ індекс, $\hat{y}$ – передбачене значення, $\hat{y}$ – істинне значення, яке шукається.

В Таблиці 1 представлені результати різних алгоритмів з тренувальними даними, але без змін базових параметрів. Як видно з таблиці, найкраще себе показали алгоритм найближчого сусіда, методи опорних векторів з поліноміальною та радіально базисною функцією. Дерева рішень

показали результат похибки нуль, це свідчить про те, що цей алгоритм при стандартних параметрах схильний до перенавчання, тобто добре розуміє тренувальні дані, але не зможе дати точні передбачення на тих, які не є в тренувальному наборі. Серед цих алгоритмів вартує взяти ті, які найкраще себе показали та знайти найкращі гіперпараметри для моделі, також варто це зробити й для алгоритму дерева рішень, щоб точно зрозуміти чи може він бути ефективним.

Таблиця 1

Порівняльний аналіз похибки передбачення на тренувальних даних при базових параметрах алгоритмів

	Лінійна регресія	Метод найближчих сусідів	Метод опорних векторів (лінійна функція)	Метод опорних векторів (поліноміальна функція)	Метод опорних векторів (радіально базисна функція)	Дерева рішень
RMSE (тренувальні дані)	0.419	0.27	0.437	0.236	0.215	0.0

В Таблиці 2 представлені результати різних алгоритмів, як з тренувальними так і з тестовими даними (дані, які не були доступні в тренувальному наборі даних). Ці результати відрізняються від Таблиці1 тим, що алгоритми використовували найоптимальніші гіперпараметри. Варто зауважити, що найкращими гіперпараметрами для методу опорних векторів з поліноміальною функцією виявились стандартні параметри, тому похибка лишилась така сама як у Таблиці 1. Інші ж алгоритми показали приріст точності, а алгоритм дерев більше немає проблеми перенавчання. Також в цій таблиці є дані про похибку передбачення моделі на тестових даних.

Таблиця 2

Порівняльний аналіз похибки передбачення на тренувальних та тестових даних

	Лінійна регресія	Метод найближчих сусідів	Метод опорних векторів (лінійна функція)	Метод опорних векторів (поліноміальна функція)	Метод опорних векторів (радіально базисна функція)	Дерева рішень
RMSE (тренувальні дані)	0.419	0.22	0.437	0.236	0.082	0.109
RMSE (тестові дані)	0.488	0.318	0.55	0.586	0.203	0.303

Іншими способами покращення моделей є так звані ансамблеві алгоритми [14], що полягають в комбінуванні декількох базових моделей для отримання кращих прогнозів ніж може забезпечити кожна модель окремо. Це може бути досягнуто за допомогою різних підходів, таких як голосування (наприклад, багатокласова класифікація), зважування (де ваги призначаються кожній моделі) або навіть навчання базових моделей на різних підмножинах даних (бутстреп агрегація). Такі ансамблеві методи можуть покращувати стійкість та точність моделі, а також зменшувати ризик перенавчання. Інші популярні ансамблеві методи включають в себе багінг, випадковий ліс, градієнтний та адаптивний бустінг.

Багінг (Bagging, від BootstrapAggregating)[15] – це ансамблевий метод машинного навчання, який використовується для покращення стійкості та точності моделі. Основна ідея полягає в тому,

щоб навчити кілька однакових базових моделей на різних підмножинах даних, вибраних за допомогою випадкового вибору з повторенням.

Адаптивний бустінг (AdaptiveBoosting)[16] – це інший ансамблевий метод машинного навчання, який також використовує декілька базових моделей для покращення точності класифікації. Проте, він використовує інший підхід до комбінування моделей порівняно з багінгом, а саме використовується послідовне навчання слабких класифікаторів, при цьому кожна нова модель фокусується на прикладах, які були неправильно класифіковані попередніми моделями.

Градієнтнийбустінг[17] – це ансамблевий метод машинного навчання, який використовується для рішення задач як класифікації, так і регресії. Він базується на ітеративному навчанні послідовної серії базових моделей, зазвичай дерев рішень, кожна з яких коригує попередній прогноз наступної моделі. Градієнтнийбустінгдобре працює з великою кількістю даних та може робити точні прогнози навіть у випадку складних взаємозв'язків між ознаками.

Випадковий ліс (RandomForest)[18] – це ансамблевий метод машинного навчання, який використовується для класифікації та регресії. Він базується на ідеї побудови кількох дерев рішень та комбінування їх прогнозів для отримання кінцевого результату.

Як базова модель передбачення для алгоритмів багінгу та адаптивного бустінгу був обраний алгоритм який показав найменшу похибку на тренувальних та тестових даних – метод опорних векторів з радіально базисною функцією. Для градієнтного бустінгу й випадкового лісу використовуються дерева рішень, як базова модель.

Таблиця 3

Порівняльний аналіз похибки передбачення на тренувальних та тестових даних ансамблевих алгоритмів

	Багінг	Адаптивний Бустінг	Градієнтний Бустінг	Випадковий Ліс
RMSE (тренувальні дані)	0.23	0.084	0.076	0.11
RMSE (тестові дані)	0.332	0.206	0.21	0.219

З Таблиці 3 можна побачити, що майже кожен з ансамблевих алгоритмів, є близьким до результату найбільш вдалого за результатами тестування алгоритму: методу опорних векторів з радіальною базисною функцією.

Тестові й тренувальні набори даних розділяються випадковим методом, отже існує ймовірність того, що результати можуть бути залежні від цього розподілу, незважаючи на використання підходу крос-валідації. Для того, щоб отримати фінальні результати тренувальні й тестові дані треба перегенерувати декілька разів й отримати похибку передбачення для кожної ітерації й порівняти результати алгоритмів.

Як можна побачити з результатів тестування, які представлені в Таблицях 4 та 5, кожен алгоритм зберігає близьке значення похибки передбачення при різних наборах тренувальних даних, які були розділені в межах ітерацій, що вказує на стабільність та чіткість налаштування алгоритмів. Варто сказати, що похибка передбачення в такому вигляді може не бути достатньо інформативною, щоб отримати кращу інформацію про якість моделі, можна застосувати формулу середньої абсолютної відсоткової похибки (*MAPE*) [19]. Для обчислення середньої абсолютної відсоткової похибки використовують формула (2):

$$MAPE(y,\hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}\frac{|y_i - \hat{y}_i|}{\max(\in, |y_i|)}, \quad (2)$$

де n– кількість записів, i– індекс, $\hat{y}$– передбачене значення, y– істинне значення, яке шукається, ϵ – мале позитивне значення, яке має запобігти можливому діленню на нуль.

Таблиця 4

Порівняльний аналіз похибки передбачення на тренувальних та тестових даних для декількох ітерацій

	Лінійна регресія	Метод найближчих сусідів	Метод опорних векторів (лінійна функція)	Метод опорних векторів (поліноміальна функція)	Метод опорних векторів (радіально базисна функція)	Дерева рішень
RMSE базове (тренувальні дані)	0.419	0.22	0.437	0.236	0.082	0.109
RMSE ітерація 1 (тренувальні дані)	0.423	0.232	0.458	0.258	0.084	0.100
RMSE ітерація 2 (тренувальні дані)	0.433	0.218	0.454	0.248	0.083	0.100
RMSE ітерація 3 (тренувальні дані)	0.425	0.213	0.453	0.233	0.084	0.101
RMSE базове (тестові дані)	0.488	0.318	0.55	0.586	0.203	0.303
RMSE ітерація 1(тестові дані)	0.472	0.321	0.551	0.431	0.198	0.291
RMSE ітерація 2(тестові дані)	0.425	0.296	0.446	0.501	0.221	0.241
RMSE ітерація 3(тестові дані)	0.483	0.368	0.489	0.318	0.270	0.300

Таблиця 5

Порівняльний аналіз похибки передбачення на тренувальних та тестових даних ансамблевих алгоритмів для декількох ітерацій

	Багінг	Адаптивний Бустінг	Градієнтний Бустінг	Випадковий Ліс
RMSE базове (тренувальні дані)	0.23	0.084	0.076	0.11
RMSE ітерація 1 (тренувальні дані)	0.227	0.086	0.081	0.100
RMSE ітерація 2 (тренувальні дані)	0.228	0.084	0.077	0.101
RMSE ітерація 3 (тренувальні дані)	0.225	0.085	0.084	0.108
RMSE базове (тестові дані)	0.332	0.206	0.21	0.219
RMSE ітерація 1 (тестові дані)	0.281	0.201	0.193	0.229
RMSE ітерація 2 (тестові дані)	0.323	0.226	0.178	0.181
RMSE ітерація 3 (тестові дані)	0.361	0.270	0.153	0.178

В Таблицях 6 та 7 представлені результати обчислення середньої абсолютної відсоткової похибки використовуючи розподіл даних для базової ітерації.

Таблиця 6

Порівняльний аналіз середньої абсолютної похибки передбачення на тренувальних та тестових даних

	Лінійна регресія	Метод найближчих сусідів	Метод опорних векторів (лінійна функція)	Метод опорних векторів (поліноміальна функція)	Метод опорних векторів (радіально базисна функція)	Дерева рішень
MAPE (тренувальні дані)	15.6%	6.3%	15.6%	6.7%	3.4%	3.3%
MAPE (тестові дані)	19.5%	10.2%	20.5%	16.1%	7.5%	9.8%

Таблиця 7

Порівняльний аналіз похибки передбачення на тренувальних даних ансамблевих алгоритмів

	Багінг	Адаптивний Бустінг	Градієнтний Бустінг	Випадковий Ліс
MAPE (тренувальні дані)	7.9%	3.4%	2.5%	3.5%
MAPE (тестові дані)	11.8%	7.8%	7.5%	7.2%

З представленої інформації в Таблицях 6-7, можна зробити висновок, що найкращим базовим алгоритмом для цієї задачі й цього набору даних є метод опорних векторів з радіально базисною функцією. Результати цього алгоритму співмірні, а в деяких випадках навіть кращі за ансамблеві алгоритми.

Висновки

У сьогоднішніх реаліях існує велика кількість різних підходів до побудови моделей машинного навчання для вирішення наукових задач. На основі реальних даних була вирішена задача побудови моделі передбачення ефективної енергії активації дифузії відповідно до параметрів розчинника й носія. Були отримані результати ефективності роботи різних алгоритмів як звичайних так й ансамблевих у вигляді похибок RMSE й MAPE. Найбільш ефективним виявився метод опорних векторів з радіально базисною функцією з середньоквадратичної похибкою при тренувальних та тестових даних 0.082 та 0.203 відповідно, та середньою абсолютної відсоткової похибкою 3.4% та 7.5%. Варто зауважити, що ансамблеві алгоритми, які використовували метод опорних векторів з радіально базисною функцією не покращили результатів передбачення, а у випадку багінгу було погіршення точності передбачень: 7.9% при тренувальних та 11.8% при тестових даних відповідно до метрики MAPE, та 0.23 й 0.332 відповідно до метрики RMSE. Це пов’язано з тим фактом, що метод опорних векторів вже є доволі сильною моделлю передбачення, а отже для її покращення треба використовувати інші підходи: збільшення тренувальних даних або нормалізація вихідних та вхідних параметрів.

Ці результати дозволяють проаналізувати, які алгоритми найкраще підходять для задач в умовах обмеженої кількості даних для тренування моделі.

Важливою частиною процесу побудови моделі є розуміння базових концепцій як наприклад перенавчання й знання того як уникнути цих проблем. Проте знання основ машинного навчання є недостатнім, адже для побудови справді ефективної моделі є необхідним розуміння сутності самих даних, адже тільки це здатне дати відповідь наскільки похибка є критичною, а отже наскільки реально модель може бути використана науковцями.

Перелік використаних джерел

[1] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science," Nature, vol. 559, no. 7715, pp. 547-555, 2018. [Online]. Available: https://doi.org/10.1038/s41586-018-0337-2

[2] V. Kulyk et al., "Prediction of hardness, flexural strength, and fracture toughness of ZrO_2 based ceramics using ensemble learning algorithms," Acta Metallurgica Slovaca, 2023. [Online]. Available: https://doi.org/10.36547/ams.29.2.1819

[3] A. Trostianchyn et al., "Boosting – based model for solving Sm-Coalloy's maximum energy product prediction task," Archives of Materials Science and Engineering, 2022. [Online]. Available: https://doi.org/10.5604/01.3001.0016.1191

[4] J. G. Wickerand R. I. Cooper, "Will it crystallise? Predicting crystallinity of molecular materials," Cryst Eng Comm, vol. 17, no. 9, pp. 1927-1934, 2015. [Online]. Available: https://doi.org/10.1039/C4CE01912A

[5] J. Kirmanetal., "Machine-learning-accelerated perovskite crystallization," Matter, vol. 2, no. 4, pp. 938-947, 2020. [Online]. Available: https://doi.org/10.1016/j.matt.2020.02.012

[6] R. A. Friesner, "Abinitio quantum chemistry: Methodology and applications," Proceedings of the National Academy of Sciences, vol. 102, no. 19, pp. 6648-6653, 2005.

[7] D. Mauludand A. M. Abdulazeez, "A review on linear regression comprehensive in machine learning," Journal of Applied Science and Technology Trends, vol. 1, no. 4, pp. 140-147, 2020. [Online]. Available: https://doi.org/10.38094/jastt1457

[8] K. Taunk et al., "A brief review of nearest neighbor algorithm for learning and classification," in 2019 international conference oni ntelligent computing and control systems (ICCS), May 2019, pp. 1255-1260. IEEE. [Online]. Available: https://doi.org/10.1109/ICCS45141.2019.9065747

[9] B. Kumar, O. P. Vyas, and R. Vyas, "A comprehensive review on the variants of support vector machines," Modern Physics Letters B, vol. 33, no. 25, 1950303, 2019. [Online]. Available: https://doi.org/10.1142/S0217984919503032

[10] B. Charbutyand A. Abdulazeez, "Classification based on decision treeal gorithm for machine learning," Journal of Applied Science and Technology Trends, vol. 2, no. 01, pp. 20-28, 2021. [Online]. Available: https://doi.org/10.38094/jastt20165

[11] S. Smys, J. I. Z. Chen, and S. Shakya, "Survey on neural network architectures with deep learning," Journal of Soft Computing Paradigm (JSCP), vol. 2, no. 03, pp. 186-194, 2020. [Online]. Available: https://doi.org/10.36548/jscp.2020.3.007

[12] D. Morgan, "Machine Learning Materials Datasets," [Online]. Available: http://doi.org/10.6084/m9.figshare.7017254.v5

[13] T. O. Hodson, "Root means quare error (RMSE) or mean absolute error (MAE): When touse the mornot," Geoscientific Model Development Discussions, 2022, pp. 1-10. [Online]. Available: https://doi.org/10.5194/gmd-15-5481-2022

[14] T. G. Dietterich, "Ensemble learning," The handbook of brain the oryand neural networks, vol. 2, no. 1, pp. 110-125, 2002.

[15] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," Machine learning, vol. 40, pp. 139-157, 2000. [Online]. Available: https://doi.org/10.1023/A:1007607513941

[16] D. D. Margineantuand T. G. Dietterich, "Pruning adaptive boosting," in ICML, July 1997, vol. 97, pp. 211-218.

[17] L. Guelman, "Gradient boosting trees for auto insurance losscost modeling and prediction," Expert Systems with Applications, vol. 39, no. 3, pp. 3659-3667, 2012. [Online]. Available: https://doi.org/10.1016/j.eswa.2011.09.058

[18] A. Parmar, R. Katariya, and V. Patel, "A review on random forest: An ensemble classifier," in International conference on intelligent data communication technologies and internet of things (ICICI) 2018, 2019, pp. 758-763. [Online]. Available: https://doi.org/10.1007/978-3-030-03146-6_86

[19] A. DeMyttenaere, B. Golden, B. LeGrand, and F. Rossi, "Mean absolute percentage error for regression models," Neuro computing, vol. 192, pp. 38-48, 2016. [Online]. Available: https://doi.org/10.1016/j.neucom.2015.12.114

Oleksii Veretiuk [1] , Nazariy Andrushchak[2]

[1]Computer-Aided Design Department, Lviv Polytechnic National University, S. Bandery street 12, Lviv, Ukraine, E-mail:oleksii.v.veretiuk@lpnu.ua, ORCID 0009-0009-2340-4458
[2]Computer-Aided Design Department, Lviv Polytechnic National University, S. Bandery street 12, Lviv, Ukraine, E-mail:nazariy.a.andrushchak@lpnu.ua, ORCID 0000-0002-8248-404X

METHODS AND MODELS OF MACHINE LEARNING IN CHEMISTRY AND MATERIAL SCIENCE USING SOLUTE DIFFUSION EXPERIMENT

Recieved: March 12, 2024 / Revised: March 20, 2024 / Accepted: April 01, 2024

Abstract. Machine learning is a logical extension of automation using computer systems. While a large number of different areas of human activity have been improved by algorithmic software, a large number of other problems remain unsolved because creating an algorithm for them is almost impossible. One of these fields is science. The empirical approach is still main approach in achieving results, because for many studies there is still no clear mathematical apparatus. Machine learning is the solution that allows to save resources and speed up the research process. Conducting experiments always leads to collecting data about the results. Machine learning algorithms allow to use this information to build a model capable of predicting the results of experiments or the properties of new compounds. Within the scope of this article, the effectiveness of different algorithms, both standard and ensemble algorithms, is tested on the data obtained from experiments with solute diffusion. As a result, the effectiveness data of various algorithms were calculated using the formulas of root mean square error, as well as mean absolute percentage error. An example and description of the process of building different types of machine learning models are given.

Keywords: machine learning, scikit-learn, chemistry, material science, limited amount of data, solute diffusion