МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В. Н. КАРАЗІНА
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КОМП'ЮТЕРНОЇ ФІЗИКИ ТА ЕНЕРГЕТИКИ
Кафедра комп'ютерної фізики

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи
за освітньо-кваліфікаційним рівнем магістр

«Генеративний дизайн дифракційних решіток за допомогою моделі ансамблю випадкових дерев прийняття рішень»

Виконав: студент 2 курсу, групи НФ-64
Дегтяр О. А.
Спеціальність: 105 – «Прикладна фізика та наноматеріали»
Освітньо-наукова програма: «Комп'ютерна фізика»
Керівник: PhD, доцент Овчаренко А. І.
Рецензент:

Харків – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н. Каразіна

Навчально-науковий інститут комп’ютерної фізики та енергетики

Кафедра комп‘ютерної фізики

Рівень вищої освіти (освітньо-кваліфікаційний рівень) магістр

Спеціальність 105 Прикладна фізика та наноматеріали

Освітня програма Комп‘ютерна фізика

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Костянтин НЄМЧЕНКО

(підпис)

«____» _____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дегтярю Олександру Андрійовичу

1. Тема роботи: «Генеративний дизайн дифракційних решіток за допомогою моделі ансамблю випадкових дерев прийняття рішень»

керівник роботи: Овчаренко Антон Ігорович, доцент.

Затверджено наказом по університету від 23 лютого 2024 року №4201-5/144.

2.Строк подання студентом роботи 20 травня 2024 року

3.Перелік питань, які потрібно розробити:

1. Огляд літератури присвяченої застосуванню машинного навчання для задач фотоніки
2. Аналіз роботи ансамблів випадкових дерев для у фотоніці
3. Підготовка та дослідження набору даних
4. Проведення оцінки різних моделей і передбачення їх ефективності

4. План роботи

№ з/п	Назви етапів роботи
1	Огляд теоритичноного матеріалу і літератури
2	Постановка задачі
3	Введення основних досліджуваних понять
4	Аналіз моделей Decision trees та random forest
5	Дослідження методів оцінки результатів моделі
6	Аналіз методу RIGID для оберненого дизайну
7	Підготовка даних та їх дослідження
8	Оцінка моделей для передбачення їх ефективності
9	Завершення написання дипломної роботи (висновки та література)
10	Подача науковому керівнику
11	Відгук, підготовка до захисту
12	Попередній захист
13	Захист

5.Дата видачі завдання 23 лютого 2024 року

Студент _____ Олександр ДЕГТЯР
(підпис)

Керівник роботи _____ Антон ОВЧАРЕНКО
(підпис)

Перелік скорочень та абревіатур

AEM – Штучні електромагнітні матеріали (Artificial electromagnetic materials)

CEMS – Обчислювальне електромагнітне моделювання Computational electromagnetic simulation)

DNN – Глибокі нейроні мережі (deep neural networks)))

g – Геометрія

g^*– Бажана геометрія

s – Відгук електромагнітного розсіювання

s^* – Бажане s

f – Пряма модель

f^{-1} – Обернена модель

РЕФЕРАТ

Дегтяр Олександр «Генеративний дизайн дифракційних решіток за допомогою моделі ансамблю випадкових дерев прийняття рішень». Кваліфікаційна робота магістра з прикладної фізики. – Харків: ХНУ імені В. Н. Каразіна, 2024. – 41 с.

Робота складається з вступу, трьох розділів, висновків. Робота містить 14 рисунків, 1 таблицю, 31 використане літературне джерело.

Мета кваліфікаційної роботи – застосування моделей дерев рішень і random forest для оцінки ефективності відхилення, з подальшим їх використанням для оберненого дизайну.

КЛЮЧОВІ СЛОВА: МІШИНЕ НАВЧАННЯ, МЕТАПОВЕРХНІ, МТЕА РЕШІТКА, RANDOM FOREST, ДЕРЕВО РІШЕНЬ, ДИЗАЙН.

ABSTRACT

Dehtiar Oleksandr “Generative design of diffraction gratings with random forests”. Master's thesis in applied physics. – Kharkiv. – V. N. Karazin KhNU, 2024. – 41 p.

The work consists of an introduction, three sections, and conclusions. The work contains14 figures, 1 table, 31 used literary sources.

The purpose of the qualification work is to application of decision tree and random forest models to evaluate the efficiency of deviation, with their subsequent use for reverse design.

KEYWORDS: MACHINE LEARNING, METASURFACES; META GRID, RANDOM FOREST, DECISION TREE; DESIGN.

ЗМІСТ

Вступ 8

Розділ 1. Генеративний дизайн дифракційних решіток 10

1.1 Мета-гратки 11

1.2 Прямий і обернений дизайн 12

Розділ 2. Методи машинного навчання для прямого і оберненого проєктування 15

2.1 Decision trees and random forest 15

2.2 Методи оцінки результатів 19

2.3. Обернений дизайн за допомогою RIGID 21

Розділ 3 Результати тренування моделей 25

3.1. Підготовка та дослідження набору даних 25

3.2 Оцінка різних моделей для передбачення ефективності 30

Висновки 35

Список літератури 37

ВСТУП

В останні роки, було запропоновано і випробувано багато різних варіантів застосування статистичних методів машинного навчання для швидкої оцінки властивостей різних структур шляхом високорівневої інтерполяції існуючого набору даних [1, 2]. Багато з них базуються на великих глибинних мережах, які обіцяють підвищену точність, проте за рахунок великого набору тренувальних даних [3]. Отримання такого набору може бути дуже обтяжуючим завданням, особливо, коли мова заходить про необхідність 3D моделювань, то того ступеня, що є непрактичним. Одним з класичних методів в машинному навчанні є так звані дерева рішень і їх ансамблі (random forests). Вони не намагаються знайти перетворення з простору вхідних даних на меншу абстрактну площину, а будують внутрішню компактну репрезентацію наявного набору для швидкої інтерполяції. Очікувано, точність простого дерева буде нижчою в середньому за повноцінну нейронну мережу, проте її простота має деякі переваги. По-перше, швидкість навчання вимірюється секундами, або долями секунд. По-друге, ансамблі дерев, які самі по собі невеликими (за обсягом пам’яті) і швидкими (за часом виконання) моделі, які мають відносно низьку точність, показують на практиці значно кращі середні результати за рахунок взваженного усереднення всіх відповідей. По-третє, дерева рішень входять в клас інтерпретованих моделей. Це означає, що є можливість зазирнути "всередину", щоб прослідкувати, яким чином була отримана та чи інша відповідь. Останнє дуже корисне при побудові зворотньої моделі, коли вхідні і вихідні вектори змінюються місцями. Наостанок, четверте, для певних задач з відносно опуклим параметричним простором, ансамблі дерев рішень можуть демонструвати високу точність оцінки при використанні невеликих наборів тренувальних даних [4, 5, 6].

Метою цієї роботи, є дослідження можливості адекватності застосування моделей дерев рішень і random forest, з подальшим їх використанням для оберненого дизайну структур, для задачі відхилення світла мета-граткою.

Раніше, цей підхід був випробуваний для задач моделювання випромінювачів [27] та оптичних і акустичних метаматеріалів [4]. Задача, яка розглядається в цій роботі, якісно відрізняється від розглянутих раніше і представляє великий практичний інтерес [7, 8, 9]

1. ГЕНЕРАТИВНИЙ ДИЗАЙН ДИФРАКЦІЙНИХ РЕШІТОК

Дифракцією називаються явища, зумовлені здатністю хвиль змінювати напрям поширення при зустрічі з різкими неоднорідностями – невеликими непрозорими перешкодами або з отворами у непрозорому екрані.

Можливість практичного спостереження дифракції залежить від розмірів перешкод: вони повинні бути співвимірними з довжиною хвилі.

Дифракційна гратка — це сукупність багатьох дуже вузьких щілин, розділених проміжками. Найпростіша дифракційна гратка являє собою плоску прозору пластинку, на поверхні якої створена система однакових вузьких прозорих смуг (щілин), розділених однаковими непрозорими проміжками (рис.1). Відстань d між сусідніми щілинами називається періодом гратки, і є одним з її основних параметрів. Часто замість d використовують сталу дифракційної гратки $n = 1/d$ – кількість щілин на одиницю довжини гратки. Ще одним параметром гратки є загальна кількість щілин N. Для дифракційних граток високого класу $d \sim 1$ мкм (10^{-6} м) і $N \sim 10^5$.

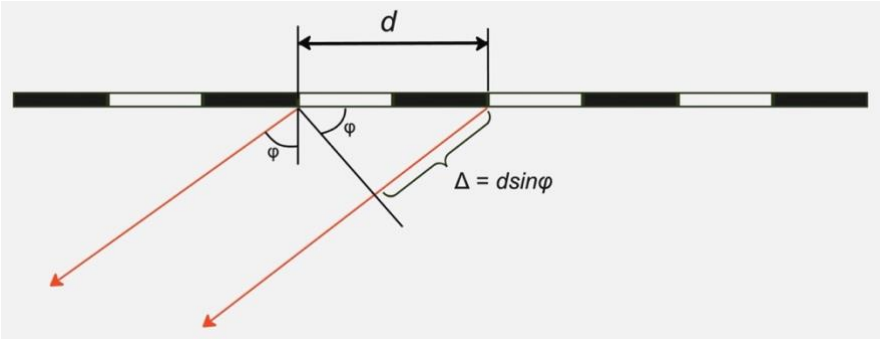


Рис. 1 Дифракційна гратка [10]

1.1 Мета-гратки

Концепція високо контрастних діелектричних структур на субхвилі призвела до розробки надтонких оптичних компонентів. Змінюючи склад матеріалу, тобто змінюючи шпаруватість решітки, можна створювати штучні матеріали з нетрадиційними властивостями. Такі матеріали називають метаповрхнямі. Метаповерхні служать нанорозмірними фазовими решітками та мають широке застосування в спектроскопії, інтегральній оптиці, оптичній фільтрації та голографії. Серед найраніших прикладів були метаповрхнямі, що складалися з серії стовпчиків з діоксиду титану, які разом створювали ефективне середовище, що підтримувало яскравий профіль показника заломлення [11].

З розвитком досліджень у цій області виявилось, що ефективність такого підходу, по суті, обмежена, особливо коли розглядається екстремальне маніпулювання хвилями. Наприклад, у канонічній задачі управління плоскою хвилею було виявлено, що фазово-градієнтний підхід фундаментально страждає від компромісу між ефективністю і кутом повороту.

Для перетворення довільного хвильового фронту можна використовувати інший шлях проектування метаповерхонь, який називається метарешітками. Він забезпечуватиме унітарну ефективність без необхідності глибоких субхвильових елементів і дозволить відмовитися від дискретизації неперервної фази або профілю імпедансу. Використовуючи фізику решітки, в якій локальний період вибирає дискретний набір порядків дифракції, і розширюючи її хвильове керування, розглядаючи складні розсіювачі з метаматеріалу як елементи, що утворюють решітку.

1.2 Прямий і обернений дизайн

Ми зосередимося на загальній парадигмі дизайну, яку називають "прямим дизайном", яка охоплює широкий спектр існуючих методів. Прямий дизайн є простою та широко використовуваною стратегією, яка передбачає, що у дизайнера є механізм для оцінки властивостей даного Artificial electromagnetic materials (AEM)[3]. Мовою математики, нам потрібно оцінити $s = f(g)$, для деякого довільного значення g, де f - так звана "пряма модель", яка пов'язує властивості матеріалу, позначеного s, з його конструкцією, позначеною g. Зазвичай g і s є числовими векторами, які представляють налаштування конструкції та електромагнітні властивості AEM відповідно. Враховуючи можливість оцінити $f(g)$ для довільного g, ми можемо потім систематично оцінювати можливі конструкції до тих пір, поки не досягнемо певного бажаного рівня згоди з деякими цільовими властивостями AEM.

Один із головних факторів, який впливає на прямий дизайн, - це час, необхідний для оцінки $f(g)$: тобто час, потрібний для оцінки кандидатського дизайну. Загалом, швидка оцінка підтверджує, що більше дизайнів можуть бути оцінені за той самий період часу, що призводить до швидшого розв'язання рівняння, або виявлення кращих дизайнів протягом фіксованого часу. Існують три загальні механізми для оцінки $f(g)$, у порядку збільшення ступеня складності: теорія, обчислювальне електромагнітне моделювання (computational electromagnetic simulation, CEMS) і експеримент [3]. Як ми тут визначаємо, теоретичний механізм - це той, де існує математичний вираз у замкнутій формі для $s = f(g)$, що безпосередньо пов'язує дизайн з електромагнітними властивостями AEM через відносно малий і передбачуваний обсяг обчислень. Тому теоретичний механізм зазвичай є найшвидшим і найбажанішим рішенням. CEMS передбачає числове розв'язання рівнянь Максвелла, що зазвичай вимагає значно більше обчислень - і, отже, часу - ніж теоретична оцінка. Нарешті, експериментальний метод вимагає, щоб кожен кандидатський дизайн був

виготовлений і характеризований в лабораторії, і, таким чином, він значно повільніший і дорожчий, ніж CEMS. Виготовлення є найменш бажаним підходом.

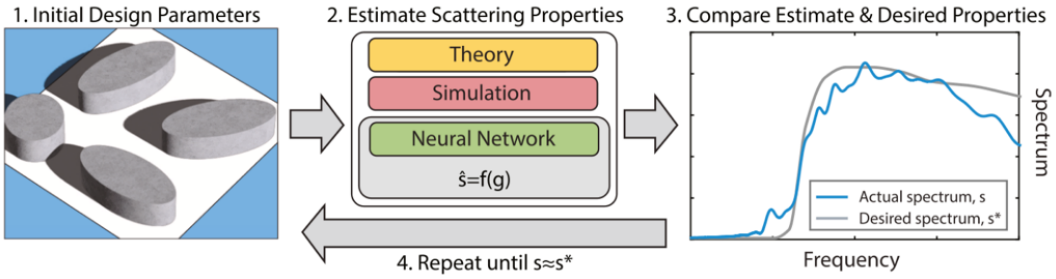


Рис.2. Робочий процес прямого дизайну ілюструється за допомогою метаповерхні, яка складається з повторюваних суперклітинок резонаторів у формі циліндра розміром 2x2. Кожен резонатор у суперклітинці має налаштований напівмінорний та напівосновний радіуси, а також спільну висоту і періодичність суперклітинки, що становить 10-вимірний вектор дизайну, g. Нашою метою є ідентифікація деякого g такого, щоб AEM виробляв спектр поглинання, позначений s*, зображений на правому вкладенні. На кроці (1) робочого процесу ми ідентифікуємо деякий кандидатський дизайн (або дизайни), а потім на кроці (2) оцінюємо їх властивості, використовуючи пряму модель метаповерхні. Малюнок було взято зі статі [3].

Процес знаходження g за умови s^* називається оберненою задачею (протилежність до прямої задачі), і може бути позначений як $g = f^{-1}(s)$. Загалом ми можемо використовувати глибокі нейронні мережі (DNN) для моделювання f^{-1} так само, як і для прямого процесу f, тренуючи DNN з s як вхідним значенням, а g як вихідним. У цій стратегії ми використовуємо точно ті ж самі дані, але просто подаємо s як вхідні дані до DNN, і потім тренуємо модель для точних прогнозів g. На жаль, цей підхід часто не працює при вирішенні обернених задач через їх неоднозначність, що призводить до поганого навчання

багатьох моделей машинного навчання, зокрема глибокі нейроні мережі (domain neural networks, DNN). Неоднозначні задачі - це ті, які порушують певні критерії, називані умовами Гадамара більш детально з умовами можна ознайомитись [12,13]. Хоча ми тут не розглядаємо конкретно неоднозначність, ми коротко перераховуємо умови Гадамара. Гадамар висунув концепцію добре поставленої проблеми і визначив, що вона повинна відповідати трьом критеріям:

H1: Існування (існує рішення);

H2: Унікальність (існує тільки одне рішення);

H3: Стабільність (рішення залежить неперервно від початкових умов). Обернена недобре поставлена проблема тоді є такою, яка порушує будь-яку з умов H1 - H3. Переважна більшість прямих проблем AEM є добре поставленими, оскільки ми знаємо, за дизайном проблеми, що існує точно один набір розсіювання, s, відповідний кожному дизайну AEM. Навпаки, коли вказується бажане розсіювання, s^*, для деякої довільної системи AEM, можливо, що два (або більше) різних дизайнів дають розсіювання, які близько йому відповідають, порушуючи умову H2. У літературі критерій H2 часто називається одне-до-багатьох, і це дійсно широко визнана проблема в багатьох дослідженнях AEM.

2. МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ПРЯМОГО І ОБЕРНЕНОГО ПРОЄКТУВАННЯ

2.1 Decision trees and random forest

Дерево рішень — це модель, яка в певному сенсі схожа на модель прийняття рішень людиною. Наприклад, коли ми переходимо дорогу, наш мозок уявно "відповідає на запитання" такого типу: який світлофор горить? чи є поруч рухомі транспортні засоби? яка відстань до них? і так далі. Ці запитання можуть утворювати ієрархію. Наприклад, якщо горить червоне світло, ми приймаємо рішення "не переходити дорогу", а при зеленому сигналі "задаємо додаткові запитання" про безпеку переходу вулиці.

Схожу ідею можна використовувати для класифікації кредитоздатних осіб банком. Ця задача відома як кредитний скорінг. Працівник банку задає запитання клієнту, і за його відповідями на певному етапі вирішує, чи доцільно видати кредит такому клієнту. Такий же процес відбувається, наприклад, при постановці діагнозу у лікаря. Подібний процес прийняття рішень звичайно візуалізується у формі деревоподібної структури (блок-схеми Рис.3)

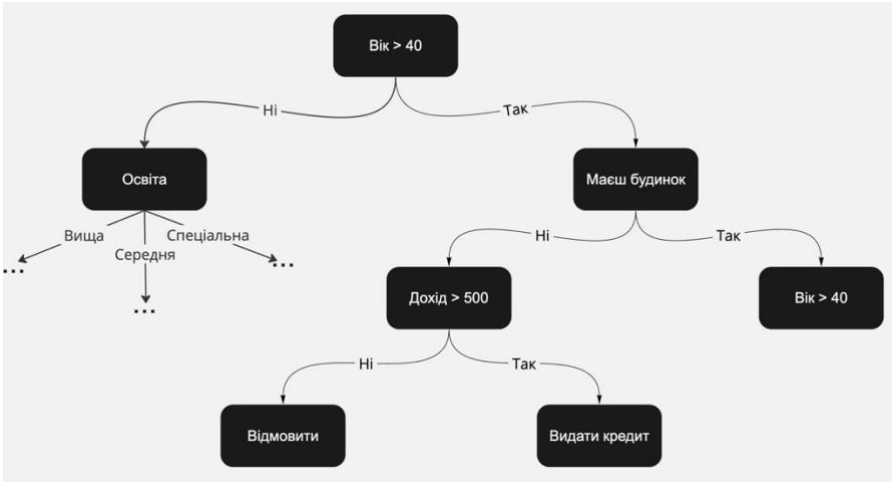


Рис.3. Блок схема ілюструю банкову систему прийняття рішень

Ентропія - фундаментальне поняття в теорії інформації, введене засновником цієї теорії, Клодом Шенноном. Інтуїтивно ентропію можна розуміти як "ступінь невизначеності" або "ступінь хаосу" у системі. Чим більш впорядкованою і визначеною є система, тим менша її ентропія. Задача дерева рішень полягає в приведенні порядку до набору значень цільової змінної. Отже, їхня мета - максимальне зменшення ентропії.

Математично ентропія системи з N можливими станами обчислюється за формулою Шеннона.

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

де p_i – вірогідність і-го стану системи.

В якомусь сенсі поняття ентропії обернено поняттю інформації: чим менша ентропія
(невизначеність), тим більше у нас інформації про систему. Введемо поняття приросту інформації (information gain) як зменшення ентропії.

$$IG = S_0 - S_1$$

де S_0- ентропія початкової системи, S_1- сумарна ентропії систем шо утворилися.

Розглянемо метод бутстрепа. Маємо вибірку X розміру N. Будемо рівномірно і рівноймовірно вибирати N об'єктів з цієї вибірки з поверненням. Це означає, що ми вибираємо N разів довільний об'єкт вибірки (з ймовірністю $\frac{1}{N}$), при цьому кожного разу вибираємо з одного і того ж множини об'єктів. Зрозуміло, що через повернення серед отриманих N об'єктів, загалом кажучи, будуть повтори. Позначимо нову вибірку X_1. Повторюючи описану процедуру M разів, можемо отримати вибірки $X_1, X_2, \ldots, X_M$, причому їх може бути скільки

завгодно, залежно від наших цілей. Звернімо увагу, що початкова вибірка могла бути не дуже великого розміру, щоб оцінювати за нею будь-які характеристики генеральної сукупності. Метод бутстрэпу дозволяє отримати достатню кількість вибірок, не залучаючи до цього жодного нового об'єкта!

Поговоримо про метод бегінга (bagging = **b**ootstrap **agg**regat**ing**) який був запропонований Лео Брейманом у 1996 році [15]. Це один з перших ансамблевих методів у машинному навчанні. Ідея полягає в тому, що з навчальної вибірки X методом бутстрепа генерується M вибірок $X_1, \dots, X_M$. Навчимо на кожній вибірці свій класифікатор $a_i(x)$. Вихідний класифікатор побудуємо як середню відповідь усіх класифікаторів:

$$a(x) = \frac{1}{M}\sum_{i-1}^{M} a_i(x).$$

Метод Random Forest – це бегінг над деревами рішень [16]. Окрім того метод використовує випадкові підпростори. Основна ідея виглядає так

- Обираємо кількість дерев у лісі: N
- Кожне з N дерев будуємо на вибірці $X_n(n = \overline{1, N})$ отриманої за допомогою бутстрепа при цьому дерево будується не на усьому ознаковому просторі а тільки з використанням $d < D$ ознак. Ці d ознак обираються випадково. За допомогою досліджень було виявлено, що для задач класифікації краще обирати $d = \sqrt{D}$, а в задачах регресії $d = \frac{D}{3}$.
- Вихідний класифікатор отримується шляхом агрегування відповідей N дерев

Порівняємо два різні класифікатори із пакету sckiti-learn мови програмування Python DecisionTreeClassifier [22, 23] та RandomForestClassifier [24, 25, 26].

DecisionTreeClassifier і RandomForestClassifier - це дві різні моделі машинного навчання, які використовуються для класифікаційних завдань на основі дерев прийняття рішень. Проте вони мають деякі ключові відмінності:

DecisionTreeClassifier:

- Базується на одному дереві прийняття рішень.
- Рішення приймається шляхом проходження по дереву від кореня до листя, де кожен вузол представляє умову за одним з ознак, а листя містять прогнози для класів.
- Модель може мати схильність до перенавчання на навчальних даних, особливо якщо дерево занадто глибоке.

RandomForestClassifier:

- Базується на ансамблі дерев прийняття рішень.
- Random forest створює кілька (зазвичай багато) дерев прийняття рішень і комбінує їх прогнози для отримання остаточного прогнозу.
- Кожне дерево будується на основі підмножини навчальних даних (з повтореннями) та підмножини ознак (зазвичай квадратний корінь від загальної кількості ознак).
- Використовується метод беггінгу (bootstrap aggregating) для зменшення розсіювання моделі та покращення узагальнюючої здатності.
- Зазвичай більш стійкий до перенавчання, ніж одиничне дерево прийняття рішень.

Загалом RandomForestClassifier є розширенням та покращенням DecisionTreeClassifier. Він часто використовується у випадках, коли потрібна більш точна класифікація, і в тому випадку, якщо є ймовірність перенавчання з одиночним деревом прийняття рішень. Проте він може потребувати більше ресурсів для навчання та прогнозування, ніж одиночне дерево прийняття рішень.

2.2 Методи оцінки результатів

Для оцінки моделей використовувався набір metrics із пакету scikit-learn. А саме така метрика як Confusion matrix. У галузі машинного навчання, а конкретно у проблемі статистичної класифікації, матриця плутанини (Confusion matrix), також відома як матриця помилок, - це конкретна таблиця, яка дозволяє візуалізувати ефективність алгоритму.

Зазвичай кожен ряд матриці представляє екземпляри у фактичному класі, тоді як кожний стовпець представляє екземпляри у передбаченому класі [17].

Це особливий вид таблиці узгодженості з двома вимірами ("фактичний" та "передбачений") та однаковими наборами "класів" у обох вимірах (кожне поєднання виміру та класу є змінною у таблиці узгодженості).

Звернемо увагу, що при порівнянні фактичного набору класифікації з передбаченим набором класифікації може бути 4 різних результати. Перше, якщо фактична класифікація позитивна и передбачена класифікація також позитивна (1,1), це називається правильним позитивним результатом, оскільки позитивний зразок правильно ідентифікується класифікатором. Друге, якщо фактична класифікація позитивна, а передбачена класифікація негативна (1,0), це називається помилковим негативним результатом, оскільки позитивний зразок неправильно ідентифікується класифікатором як негативний. Третє, якщо фактична класифікація негативна, а передбачена класифікація позитивна (0,1), це називається помилковим позитивним результатом, оскільки негативний зразок неправильно ідентифікується класифікатором як позитивний. Четверте, якщо фактична класифікація негативна и передбачена класифікація також негативна (0,0), це називається правильним негативним результатом, оскільки негативний

зразок правильно ідентифікується класифікатором. Таким чином результати для матриці плутанини можна класифікувати як (правильні позитивні, помилкові негативні, помилкові позитивні та правильні негативні). Ці чотири результати можуть бути сформульовані у вигляді 2×2 матриці плутанини (Рис. 4)наступним чином

		Predicted condition	
	Total populati = P + N	**Positive (PN**	**Negative (PI**
Actual Condition	**Negative (N)**	True Negativ	False Positiv
	Positive (P)	False Negati	True Positive

Рис. 4 Таблиця матриці плутанини [18, 19]

Усі правильні прогнози розташовані на діагоналі, тому легко візуально перевірити таблицю на помилки прогнозу, оскільки значення поза діагоналлю будуть їх представляти. Підсумовуючи 2 стовпці матриці плутанини, також можна здогадатися загальну кількість позитивних (P) та негативних (N) зразків у вихідному наборі даних, тобто

$$P = TP + FN$$

$$N = FP + TN$$

Розглянемо ще декілька метрик за якими можна охарактеризувати класифікатори.

Перша метрика називається precision_score [17] - це співвідношення

$$\text{precision}_{\text{score}} = \frac{TP}{(TP + FP)}$$

де *TP* - кількість правильно визначених позитивних випадків, а *FP*- кількість помилково визначених позитивних випадків. Precision - інтуїтивно визначає здатність класифікатора не маркувати позитивний випадок, як негативний. Найкраще значення - 1, а найгірше - 0.

Друга метрика recall_score [17] - це співвідношення

$$\text{recall}_{\text{score}} = \frac{TP}{(TP + FN)}$$

де *TP* - кількість правильно визначених позитивних випадків, а *FN* - кількість неправильно визначених негативних випадків. Recall - інтуїтивно визначає здатність класифікатора знаходити всі позитивні випадки. Найкраще значення - 1, а найгірше - 0.

Також розглянемо метрику яка може характеризувати точність наших класифікаторів під назвою score. Вона передбачає середню точність даних.

$$score = \frac{TP + TN}{(TP + TN + FP + FN)}$$

2.3. Обернений дизайн за допомогою RIGID

Ми прагнемо до швидкого зворотного проектування метаматеріалів для досягнення заданих якісних функціональних характеристик. Це зворотне завдання є складним через його невирішуваність та наявність неунікальних рішень. Попередні роботи здебільшого зосереджувалися на методах, заснованих на глибокому навчанні, які вимагають великої кількості даних, тривалого тренування та налаштування гіперпараметрів, і є неінтерпретованими [28, 29]. Щоб подолати ці обмеження, пропонується використати метод Random-forest-based Interpretable Generative Inverse Design (RIGID) [4], який є одноетапним методом зворотного проектування для швидкого генерування дизайнів

метаматеріалів із заданими функціональними характеристиками. На відміну від більшості існуючих методів, використовуючи інтерпретованість випадкового лісу, можна усунути потребу в тренуванні зворотної моделі, яка б відображала відповіді на дизайни. На основі ймовірності задоволення цільових вимог, отриманої з тренованої прямої моделі, можна вибирати дизайн-рішення, використовуючи методи Монте-Карло [30] за ланцюгами Маркова [31]. Метод RIGID функціонує як генеративна модель, яка захоплює умовний розподіл задовольняючих рішень, виходячи з заданої цілі проектування. Ефективність та результативність метода RIGID була продемонстрована на прикладах проектування акустичних та оптичних метаматеріалів, де доступні лише невеликі набори даних (менше 250 навчальних зразків) [4].

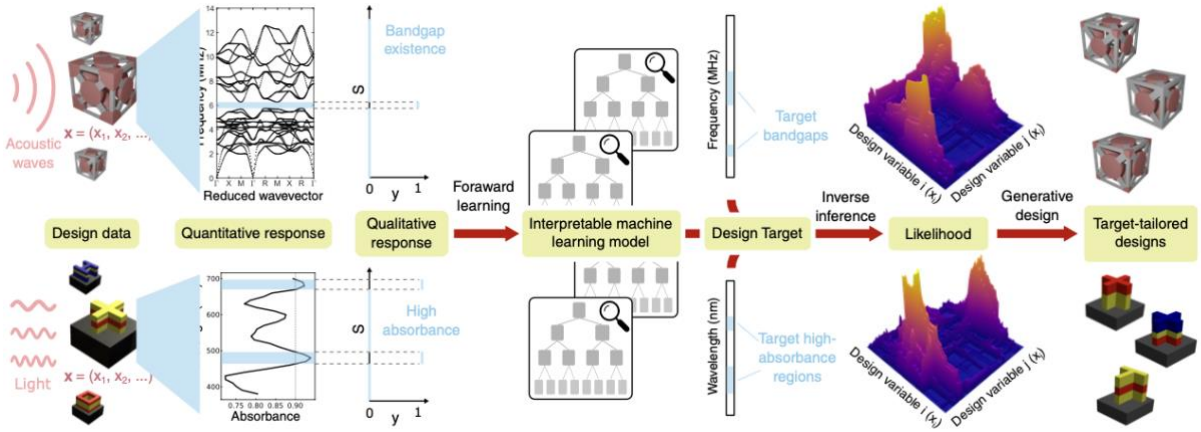


Рис. 5. Схематична діаграма методу RIGID. Спочатку ми тренуємо випадковий ліс на наборі даних дизайн-відповідь для вивчення прямої відповіді дизайну - прогнозування якісних відповідей (наприклад, існування забороненої смуги при будь-якій заданій хвильовій частоті) дизайнів. Потім, задавши ціль дизайну, ми можемо вивести ймовірність будь-якого дизайну, який задовольняє ціль, шляхом дослідження тренованого випадкового лісу. Нові дизайни зі зміненими відповідями можна генерувати шляхом вибірки простору дизайну на основі оцінки ймовірності [4].

Рис. 5 показує загальний вигляд цього методу. Зокрема, спочатку ми тренуємо випадковий ліс для прямого прогнозування. Потім, задавши ціль дизайну, ми можемо досліджувати тренований випадковий ліс, щоб вивести ймовірність будь-якого дизайну, який задовольняє ціль. Для генерації нових дизайнів, відповідних цілі, ми можемо вибирати з простору дизайну відповідно до ймовірності. У порівнянні з найбільш дослідженими методами на основі нейронних мереж, RIGID має значно менше витрат на навчання і настройку гіперпараметрів, працює більш надійно на наборах даних невеликого розміру (оскільки випадкові ліси менше схильні до перенавчання). Схоже на глибокі генеративні моделі, він може генерувати бажану кількість рішень, дозволяючи досліджувати альтернативні рішення, які можуть мати бажані властивості або функціональність поза тими, що розглядаються як цільові. Явне оцінювання ймовірності також надає інтерпретований характеристику ймовірності задоволення цілі дизайну та дозволяє здійснювати компроміс між експлуатацією та дослідженням при виборі згенерованих дизайнів.

Метод RIGID є надбудовою над методом DTGEN [27]. RIGID використовує паралельний аналіз різних дерев у лісі, і статистично зважує результати для отримання одного дизайну на виході. Цей підхід дозволяє генерувати комбінації параметрів з відповідними ймовірностями, які задовольняють вимогам, і потім оцінювати їх ефективність за допомогою прямої моделі випадкового лісу (RF). На виході ми отримуємо не конкретне значення параметрів, а інтервали з різними ймовірностями. Одне дерево в лісі дає один інтервал значень, інше – інший, але з іншою впевненістю. Інтервали можуть перетинатися або не перетинатися, і ми можемо тільки робити припущення, але з набагато більшою точністю.

У випадку з DTGEN підхід інший. Тут використовується лише одне дерево, і ліс не застосовується для оберненого дизайну. Замість цього, ліс використовується як пряма модель для розширення великого дерева. Генеративний підхід у DTGEN означає, що для побудови оберненої моделі, яка сама по собі є деревом, використовуються згенеровані дані. Це схоже на

тандемну модель у машинному навчанні, де поєднуються різні моделі для досягнення кращих результатів.

Для DTGEN Генеративний підхід означає використання згенерованих даних для побудови оберненої моделі, яка є деревом рішень. Це схоже на тандемну модель у машинному навчанні, де генерується великий набір даних для навчання великого дерева рішень.

Таким чином, RIGID використовує більш складний і паралельний підхід, тоді як DTGEN застосовує простіший, але ефективний метод з одним деревом. Обидва підходи по-різному інтерпретують і використовують термін "генеративний", забезпечуючи різні стратегії для досягнення оптимальних результатів у проектуванні матеріалів.

3. РЕЗУЛЬТАТИ ТРЕНУВАННЯ МОДЕЛЕЙ

3.1. Підготовка та дослідження набору даних

Для дослідження були використані дані з відкритого ресурсу MetaNet [20, 21] в яких було 31433 промодельовані структури, кожна з них зберігає певний набір даних таких як:

- Angle – кут під яким проходить хвиля
- Efficiency – ефективність за заданими параметрами
- pattern – дизайн решітки
- period – період решітки
- polarization – просторова орієнтація електричної складової частини електромагнітної хвилі
- thickness – товщина решітки
- wavelength – довжина хвилі

Для нас з цього набору даних важливими є параметри efficiency, pattern, wavelength та angle. Через те що саме ці параметри є унікалиними для кожного з дизайнів. Саме за допомогою них ми і будемо проводити навчання нашого Random Forest.

Першим кроком у роботи за даними було об’єднання усіх файлів в один для більш зручної робот з ними. Для цього були використані такі пакети мови Python як NumPy та h5py.

Наступним кроком було зчитування цього файлу та виокремлення з відти параметрів для нашої моделі. Для тренування моделі було використано данні с довжиною хвилі $\lambda = 900$ нм та кутом 60 градусів. І зафіксуємо ефективності як $y > 0{,}9$. Усі дизайни та ефективності розділені на тренувальні та тестові. Для цього ми розбили набір даних навпіл. Таким чином отримаємо x_{train}, x_{test}, y_{train}, y_{test}

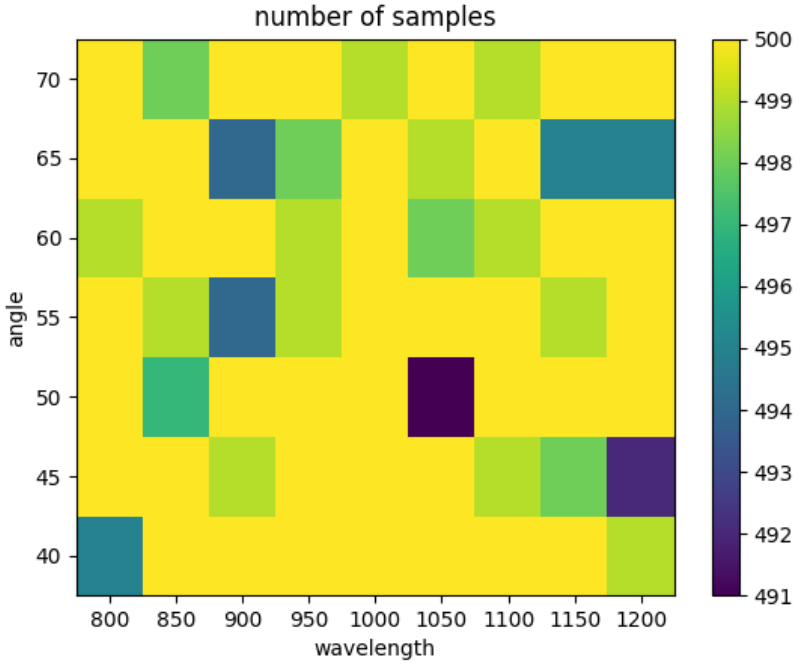


Рис.6 Загальна кількість даних для кожної з комбінації пар кут відхилення до довжини хвилі

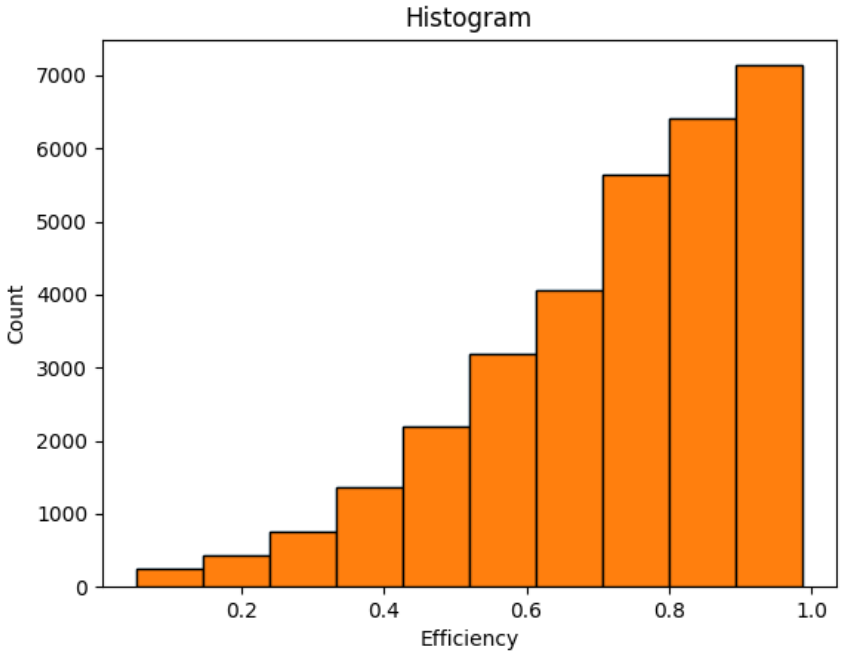


Рис.7. Гістограма кількості даних в залежності від ефективності

В тестуванні був використаний метод DecisionTreeClassifier из пакету scikit-learn

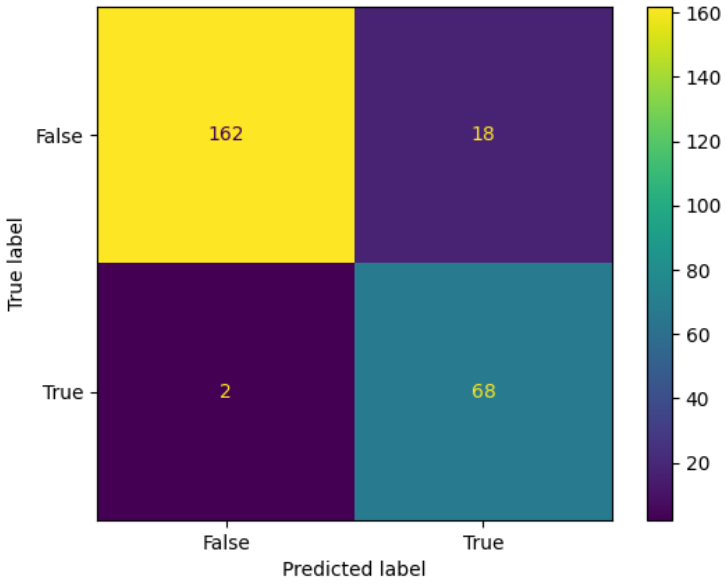


Рис. 8. True Positive (TP) =68, False Negative(FN) = 2, False Positive(FP) = 18, True Negativ(TN) = 162, $\lambda = 900$, $y > 0{,}9$, $angle = 60$

Виходячи з результату перевірки (рис. 8) у тестових даних було 70 позитивних набори і модель змогла спрогнозувати правильно 68 з них і 180 негативних набори даних з них модель змогла правильно спрогнозувати 80.

$$\text{precision}_{\text{score}} = 0.79$$

$$\text{recall}_{\text{score}} = 0.97$$

Якщо ми подивимось на параметр pattern з набору даних то побачимо що він виглядає як матриця розмірністю (1, 256) із набору 1 та
-1 де 1 відповідає за наявність матеріалу 1, а -1 за його відсутність. Зрозуміло що всі дизайни у наборі даних мають різний вигляд і для того щоб поліпшити нашу

модель ми створили функцію align яка приведе всі дизайни до умовно однакового вигляду.

Наша функція align знаходить усі -1 на початку дизайну до першої 1 та переставляє їх у кінець масиву. Тобто таким чином ми говоримо що всі наші дизайни починаються з 1 и закінчуються на -1. Це також у подальшому дасть змогу визначити ширини матеріалу у дизайнах і допоможе привести дані до одного виду, завдяки цьому для моделі такий вид поставленої задачі буде більш коректним. Після цього перетворення було також проведене тренування моделі.

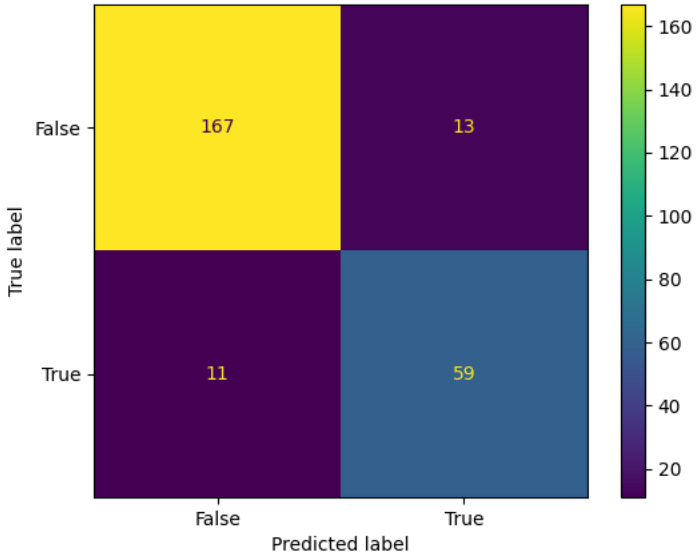


Рис. 9. True Positive(TP) =59, False Negative(FN) = 11, False Positive(FP) = 13, True Negativ(TN) = 167, $\lambda = 900$, $y > 0{,}9$, $angle = 60$

$$\text{precision}_{\text{score}} = 0.819$$

$$\text{recall}_{\text{score}} = 0.842$$

Була створена функція яка порахує ширини матеріалу у наших дизайнах. Вона рахує послідовні кількості 1 та -1 і ділить результат на довжину масиву.

Таким чином ми отримаємо нові масиви даних в яких вже будуть вказані саме ширини. За таких умов класифікаторам зможуть швидше обробляти дані відповідно швидше тренуватись і як результат якісніше розділяти дизайні на позитивні та негативні випадки. Виходячи з того що наш набір даних має багато дизайнів стало зрозумілим що і кількість ширин у них є різною. Тому для подальших тестів це число буде зафіксована як

$$N_{width} = 10$$

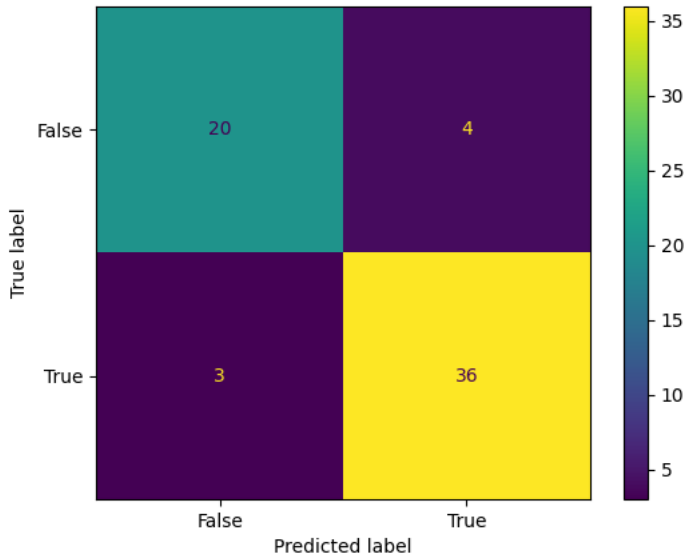


Рис. 10. True Positive(TP) =36, False Negative(FN) = 3, False Positive(FP) =4, True Negativ(TN) =20, $\lambda = 900$, $N_{width} = 10$, $y > 0{,}9$, $angle = 60$

Це тренування було проведено вже після використання нашої функції для обчислення ширин і тут ми можемо бачити такі значення метрик, $\text{precision}_{\text{score}} = 0.9$ та $\text{recall}_{\text{score}} = 0.923$

3.2 Оцінка різних моделей для передбачення ефективності

Проведемо тренування класифікаторів и порівняємо результати

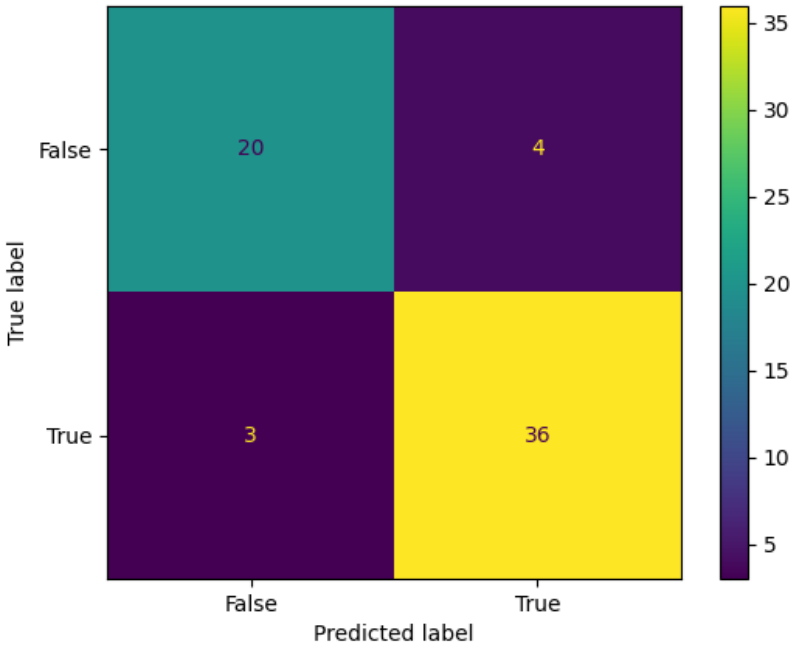


Рис.11. Таблиця результату для DecisionTreeClassifier. True Positive(TP) = 36, False Negative(FN) = 3, False Positive(FP) = 4, True Negativ(TN) = 20, $\lambda = 900$, $N_{width} = 10$, $y > 0{,}9$, $angle = 60$

Також для тренувальних даних цього класифікатора було побудовано діаграму (рис. 11)

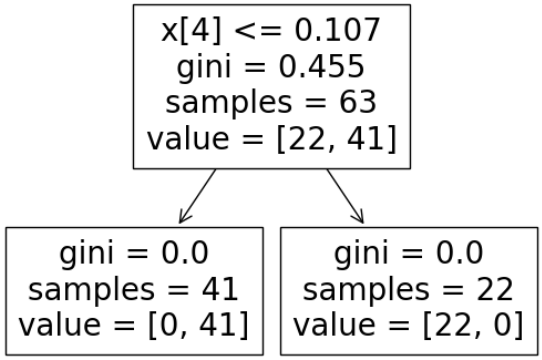


Рис.12 Діаграма роботи DecisionTreeClassifier

Ця діаграма наочно демонструє, що принци роботи дерева рішень повністю відповідає своїй назві. Кожен з вузлів має корисні характеристики. Samples говорить скільки екземплярів з тренувальної вибірки відповідає цій умові, value розбиває цю кількість по цільовим параметрам (класам), gini дає індикс домішок Джині, який оцінює кількість екземплярів, шо була класифікована некоректно. $G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$ де i - індекс вузла, $p_{i,k}$ - співвідношення кількості екземплярів классу k та всіх тренувальних (вхідних) екземплярів для цього вузла: якщо gini = 0 то це відповідє ідеальному випадку, коли всі приклади були класифіковані вірно. Як показано на діаграмі у першому вузлі загальна кількість екземплярів $samples = 63$. Вона розбиті по цільовим параметрам $value = [22, 41]$. Далі йдуть два вузли в яких $samples = 41$ та $samples = 22$. В них $value$ розбиває кількість на цільві параметри як $value = [0, 41]$ та $value = [22, 0]$ відповідно. Вони є кінцевими вузлами у побудові дерева рішень

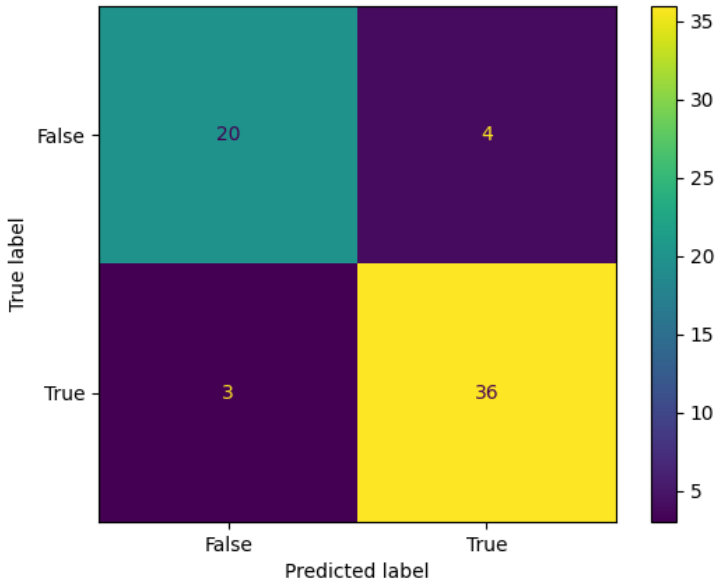


Рис.13. Таблиця з результату для RandomForestClassifier. True Positive(TP) =36, False Negative(FN) = 3, False Positive(FP) = 4, True Negativ(TN) = 20, $\lambda = 900$, $N_{width} = 10$, $y > 0{,}9$

Як можна бачити із таблиць (рис. 12, 13) що DecisionTreeClassifier зміг правильно передбачити $N_{true} = 20 + 36 = 56$ варіанти.

І RandomForestClassifier передбачив $N_{true} = 20 + 36 = 56$.

Тобто класифікатор з таким набором даних впорались однаково. Це також можна побачити поглянувши на метрики для DecisionTreeClassifier $\text{precision}_{\text{score}} = 0.9$, $\text{recall}_{\text{score}} = 0.923$, та для RandomForestClassifier $\text{precision}_{\text{score}} = 0.9$, $\text{recall}_{\text{score}} = 0.923$.

Далі ми змінили параметр наших класифікаторі max_depth вказавши, max_depth = None. Це означає, що глибина наших дерев рішень тепер буде максимальною. Як можна бачити з попереднього порівняння через малу кількість наборів даних з ширино $N_{width} = 10$. Ми маємо однакові результати при тестуванні. Було прийнято рішення зменшити кількість ширин у дизайнах $N_{width} = 8$ таким чином збільшивши кількість наборів даних

Виходячи с цих змін було проведено тестування двох класифікаторі

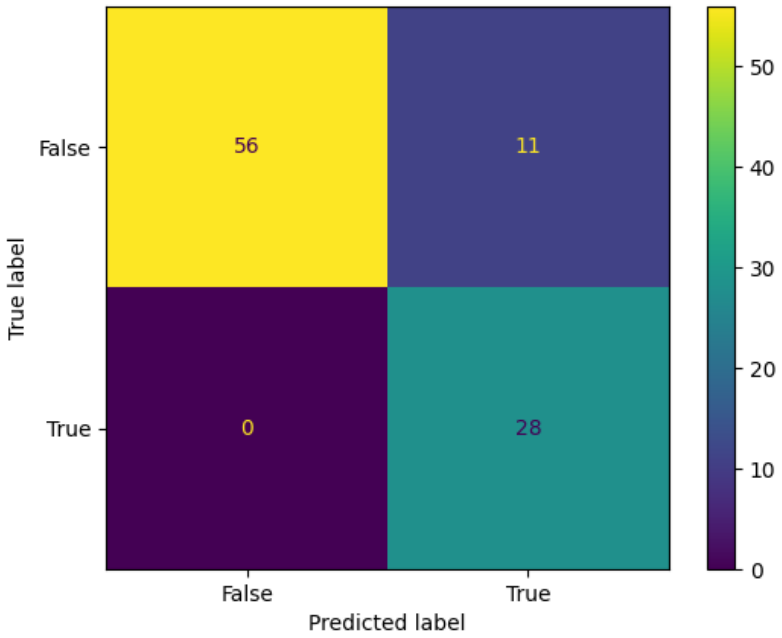


Рис.14. Таблиця результату для DecisionTreeClassifier. True Positive(TP) = 28, False Negative(FN) = 0, False Positive(FP) = 11, True Negativ(TN) = 56, $\lambda = 900$, $N_{width} = None$, $y > 0{,}9$

В цьому випадку кількість правильних варіантів $N_{true} = 28 + 56 = 84$

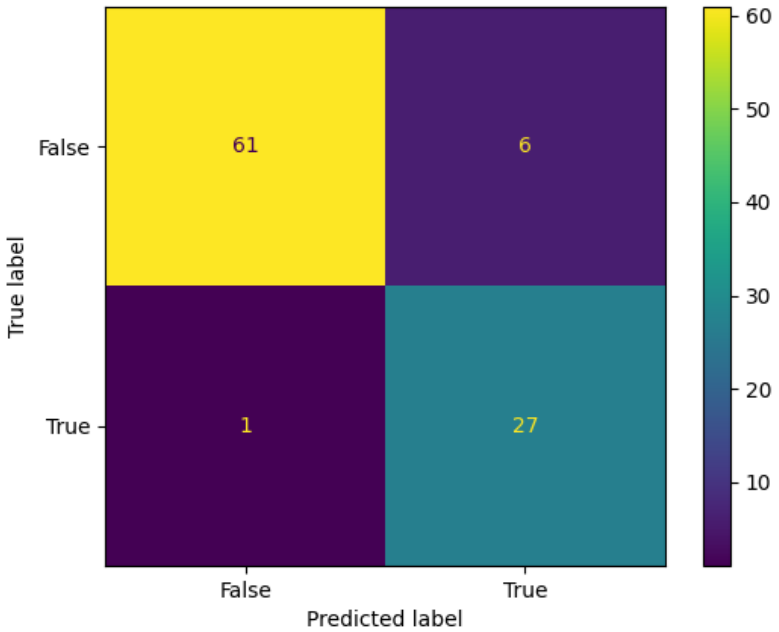


Рис.15. Таблиця з результату для RandomForestClassifier. True Positive(TP) = 27, False Negative(FN) = 1, False Positive(FP) = 6, True Negativ(TN) = 61, $\lambda = 900$, $N_{width} = None$, $y > 0{,}9$

В даній таблиці (рис.15) наведений результат тестування RandomForestClassifier і тут можна побачити, що $N_{true} = 27 + 61 = 88$. Тобто при умові, що глибина дерев максимальна класифікатор RandomForestClassifier впорався краще.

Також розглянемо ще одну метрику score.

$$dt = 0.884$$

$$rf = 0.926$$

Де dt – це score для DecisionTreeClassifier, а rf – для RandomForestClassifier

Можемо зробити висновок, що у випадку коли для тренування і тестування використовується більше даних класифікатор RandomForestClassifier показує себе краще и набирає більший score

Ще декілька порівняльних метрик

$$\text{precision}_{\text{score_}dt} = 0.7179$$

Значення для DecisionTreeClassifier

$$\text{precision}_{\text{score_}rf} = 0.818$$

Значення для RandomForestClassifier

$$recall_{\text{score_}dt} = 1$$

Значення для DecisionTreeClassifier

$$\text{recall}_{\text{score_}rf} = 0.964$$

Значення для RandomForestClassifier

ВИСНОВКИ

У даній роботі було досліджено можливості застосування моделей decision tree і random forest для прямого і оберненого дизайну оптичних метаматеріалів з заданими функціональними властивостями. Продемонстрована ефективність роботи ансамблів дерев рішень для вирішення прямої задачі оцінки ефективності дифракції метагратки. Отримана модель може бути використована як тандемна модель для задач оберненого дизайну, а також сама по собі використана для отримання передбачень структури за методом RIGID. Завдяки використанню випадкових лісів, метод дозволяє уникнути перенавчання та зменшити витрати на навчання і налаштування гіперпараметрів, що особливо важливо при роботі з невеликими наборами даних. Інтерпретованість методу RIGID забезпечується явним оцінюванням ймовірності задоволення цілі дизайну, що дозволяє проводити аналіз отриманих результатів та здійснювати баланс між експлуатацією та дослідженням. Метод демонструє здатність генерувати бажану кількість альтернативних рішень, що дозволяє досліджувати можливі дизайни, які можуть мати додаткові бажані властивості або функціональність. На відміну від більшості методів на основі глибоких нейронних мереж, RIGID не потребує великого обсягу даних для навчання, що значно знижує вимоги до ресурсів. Метод RIGID показав вищу стійкість та надійність у порівнянні з іншими підходами, що робить його перспективним для застосування у широкому спектрі задач оберненого дизайну. Результати цієї дипломної роботи можуть бути основою для подальших досліджень у галузі оберненого дизайну метаматеріалів, сприяючи розвитку нових технологій та інновацій у цій сфері. Впровадження RIDIG та інших статистичних методів інтерполяції на основі random forest у практичні застосування може значно покращити процес розробки нових матеріалів, зменшити витрати та час на розробку, а також сприяти створенню нових продуктів з унікальними властивостями.

Отже, результати даної роботи демонструють потенціал застосування методів дерев рішень і random forest для оптимізації одновимірних оптичних дифракційних мета-граток.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Khalil M. et al. Machine Learning, Deep Learning and Statistical Analysis for forecasting building energy consumption—A systematic review //Engineering Applications of Artificial Intelligence. – 2022. – Т. 115. – С. 105287.

[2] Zhong S. et al. Machine learning: new ideas and tools in environmental science and engineering //Environmental Science & Technology. – 2021. – Т. 55. – №. 19. – С. 12741-12754.

[3] Deng Y. et al. Deep inverse photonic design: a tutorial //Photonics and Nanostructures-Fundamentals and Applications. – 2022. – Т. 52. – С. 101070.

[4] Chen W. et al. Generative Inverse Design of Metamaterials with Functional Responses by Interpretable Learning //arXiv preprint arXiv:2401.00003. – 2023.

[5] Jiang J. et al. Free-form diffractive metagrating design based on generative adversarial networks //ACS nano. – 2019. – Т. 13. – №. 8. – С. 8872-8878.

[6] So S., Rho J. Designing nanophotonic structures using conditional deep convolutional generative adversarial networks //Nanophotonics. – 2019. – Т. 8. – №. 7. – С. 1255-1261.

[7] Ra'di Y., Sounas D. L., Alù A. Metagratings: Beyond the limits of graded metasurfaces for wave front control //Physical review letters. – 2017. – Т. 119. – №. 6. – С. 067404.

[8] Popov V., Boust F., Burokur S. N. Controlling diffraction patterns with metagratings //Physical Review Applied. – 2018. – Т. 10. – №. 1. – С. 011002.

[9] Jiang J. et al. Free-form diffractive metagrating design based on generative adversarial networks //ACS nano. – 2019. – Т. 13. – №. 8. – С. 8872-8878.

[10] Бригінець В. П. ФІЗИКА. ВЧИМОСЯ РОЗВ'ЯЗУВАТИ ЗАДАЧІ [Електронний ресурс] / В. П. Бригінець, С. О. Подласов, О. В. Матвійчук. – 2020. – Режим доступу до ресурсу: http://physics.zfftt.kpi.ua/mod/page/view.php?id=370.

[11] Genevet P. et al. Recent advances in planar optics: from plasmonic to dielectric metasurfaces //Optica. – 2017. – Т. 4. – №. 1. – С. 139-152.

[12] Omar Khatib, Simiao Ren, Jordan Malof, Willie J. Padilla, Deep learning the electromagnetic properties of metamaterials—a comprehensive review, Adv. Funct. Mater. 31 (31) (2021), 2101748.
[13] Mueller J. L., Siltanen S. (ed.). Linear and nonlinear inverse problems with practical applications. – Society for Industrial and Applied Mathematics, 2012.
[14] Decision Trees and Random Forest [Електронний ресурс] // 2019 – Режим доступу до ресурсу: https://www.kaggle.com/code/emstrakhov/decision-trees-and-random-forest/notebook
[15] Breiman L. Bagging predictors //Machine learning. – 1996. – Т. 24. – С. 123-140.
[16] Altman N., Krzywinski M. Ensemble methods: bagging and random forests //Nature Methods. – 2017. – Т. 14. – №. 10. – С. 933-935.
[17] Powers D. M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation //arXiv preprint arXiv:2010.16061. – 2020.
[18] Piryonesi S. M., El-Diraby T. E. Data analytics in asset management: Cost-effective prediction of the pavement condition index //Journal of Infrastructure Systems. – 2020. – Т. 26. – №. 1. – С. 04019036.
[19] Sammut C., Webb G. I. (ed.). Encyclopedia of machine learning. – Springer Science & Business Media, 2011.
[20] metanet [Електронний ресурс] – Режим доступу до ресурсу: http://metanet.stanford.edu/.
[21] Jiang J. et al. MetaNet: a new paradigm for data sharing in photonics research //Optics express. – 2020. – Т. 28. – №. 9. – С. 13670-13681.
[22] Swain P. H., Hauska H. The decision tree classifier: Design and potential //IEEE Transactions on Geoscience Electronics. – 1977. – Т. 15. – №. 3. – С. 142-147.
[23] Safavian S. R., Landgrebe D. A survey of decision tree classifier methodology //IEEE transactions on systems, man, and cybernetics. – 1991. – Т.

21. – №. 3. – С. 660-674.

[24] Parmar A., Katariya R., Patel V. A review on random forest: An ensemble classifier //International conference on intelligent data communication technologies and internet of things (ICICI) 2018. – Springer International Publishing, 2019. – С. 758-763.

[25] Pal M. Random forest classifier for remote sensing classification //International journal of remote sensing. – 2005. – Т. 26. – №. 1. – С. 217-222.

[26] Belgiu M., Drăguţ L. Random forest in remote sensing: A review of applications and future directions //ISPRS journal of photogrammetry and remote sensing. – 2016. – Т. 114. – С. 24-31.

[27] Elzouka M. et al. Interpretable forward and inverse design of particle spectral emissivity using common machine-learning models //Cell Reports Physical Science. – 2020. – Т. 1. – №. 12.

[28] Lee, Xian Yeow, et al. "Fast inverse design of microstructures via generative invariance networks." *Nature Computational Science* 1.3 (2021): 229-238.

[29] Ren, Zekun, et al. "Inverse design of crystals using generalized invertible crystallographic representation." *arXiv preprint arXiv:2005.07609* 3.6 (2020): 7.

[30] Metropolis, Nicholas, and Stanislaw Ulam. "The monte carlo method." *Journal of the American statistical association* 44.247 (1949): 335-341.

[31] Nummelin, Esa. *General irreducible Markov chains and non-negative operators*. No. 83. Cambridge University Press, 2004.