Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: **Розроблення інформаційної системи аналізу передвісників землетрусів**

Виконав: студент VI курсу, групи КН-61м
спеціальності 122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)
Вороновський Д. Ю.
(прізвище та ініціали)

Керівник Пірко І. Б.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Львів – 2024 р.

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Борецька І. Б.
"____" _____2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Вороновський Данило Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи **Розроблення інформаційної системи аналізу передвісників землетрусів**

керівник роботи Пірко І. Б., канд. фіз.-мат. наук, доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 13.02. 2023 року № С-49

2. Термін подання студентом роботи 05. 01. 2024 р.

3. Вихідні дані до роботи:

- вивчити предметну область, проаналізувати існуючі фактори та методи моделювання, а також відповідні програмні продукти;
- розглянути і використати алгоритми, які лежать в основі математичної моделі аналізу передвісників землетрусів;
- спроектувати інформаційну систему з допомогою мови програмування Python та відповідних бібліотек для побудови інтерфейсу та візуалізації результатів.
- протестувати роботу програмного продукту.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Розділ 1. Стан проблемної області

Розділ 2. Інформаційне забезпечення

Розділ 3. Математичне забезпечення

Розділ 4. Програмне забезпечення

Розділ 5. Стартап проекту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Додаток А.

6. Дата видачі завдання 15 лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних даних та інших джерел згідно досліджуваної теми	15.02-31.03.2023	виконано
2	Аналіз досліджуваної теми та вибір відповідних варіантів її розробки	01.04-30.04. 2023	виконано
3	Постановка задачі та її формалізація	01.05-31.05. 2023	виконано
4	Вибір та обґрунтування методів і засобів проведення дослідження	01.06-30.06. 2023	виконано
5	Розроблення концептуальної схеми реалізації завдання	01.07-31.07. 2023	виконано
6	Програмна реалізація завдання	01.08-30.09. 2023	виконано
7	Тестування програмного продукту та отриманих результатів	01.10-30.10. 2023	виконано
8	Розробка пояснювальної записки магістерської роботи	01.11-30.11. 2023	виконано
9	Корегування пояснювальної записки згідно вимог, розроблення презентації	01.12-04.01. 2024	виконано

Студент _____ Вороновський Д. Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пірко І. Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 85 сторінок пояснювальної записки, 16 рисунків, 4 таблиці, 10 джерел, 1 додаток.

В дипломній роботі засобами Python та його бібліотек візуалізації результатів досліджень отримано інформаційну систему, з допомогою якої можна моделювати та досліджувати вплив передвісників землетрусу в досліджуваних точках на місцевості. Отримано показники для оцінки адекватності отриманої моделі, оцінки на їх основі точності прогнозів ймовірності виникнення землетрусів.

Ключові слова: передвісники землетрусів, Python, Scikit-Learn, Matplotlib.

ABSTRACT

The thesis contains 85 pages of explanatory note, 16 figures, 4 tables, 10 sources, 1 appendice.

In the diploma thesis, using Python and its libraries for visualization of research results, an information system was obtained, with the help of which it is possible to simulate the impact of investigating the impact of earthquake harbingers at the studied points on the terrain. Indicators were obtained for assessing the adequacy of the obtained model, assessing the accuracy of forecasts of the probability of earthquakes based on them.

Keywords: earthquake predictors, Python, Scikit-Learn, Matplotlib.

.

ТЕХНІЧНЕ ЗАВДАННЯ

В дипломній роботі потрібно вирішити такі завдання.

Розробити інформаційну систему для аналізу передвісників землетрусів:

- провести огляд літератури по даній тематиці;
- проаналізувати існуючі системи прогнозування землетрусів;
- розробити математичну модель системи та алгоритм функціонування інформаційної системи;
- реалізувати програмну інформаційної системи;
- провести просторовий аналіз даних та кореляцію отриманих результатів.

ПЕРЕЛІК СКОРОЧЕНЬ І СПЕЦІАЛЬНИХ ТЕРМІНІВ

DAS (distributed acoustic sensing) – розподілене акустичне зондування;

DT (Decision Tree) – класифікатор дерева рішень;

EDA (Exploratory Data Analysis) – попередній аналіз даних;

EEW (earthquake early warning) – системи раннього попередження землетрусів;

IDE (Integrated Development Environment) – інтегроване середовище розробки;

KNN (K-Nearest Neighbors) – метод k-найближчих сусідів;

ML (Machine Learning) – машинне навчання;

RF (Random Forest) – класифікатор випадкового лісу;

SVM (Support Vector Machines) – метод опорних векторів;

USGS (United States Geological Survey) – геологічна служба США;

ШІ – штучний інтелект.

ЗМІСТ

ВСТУП

Актуальність дипломної роботи

Сейсмологічні події протягом усього часу існування людства завдавали людям шкоди. Останнім часом ситуація загострюється промисловим освоєнням сейсмоактивних регіонів та розробкою родовищ корисних копалин, які глибоко залягають під землею. Тому для прийняття рішень, які будуть спрямовані на безпеку населення, зниження збитків та проведення превентивних заходів, потрібно здійснювати прогноз сейсмічності. Вихідною інформацією для прогнозу є сейсмічні каталоги, що містять відомості про слабкі поштовхи, їх місце, час і силу, що передують сильним. При цьому в завдання прогнозу входять визначення сили очікуваної сейсмічної події, місця і часу його виникнення.

Аналітичні системи на основі аналізу даних та машинного навчання застосовуються в сейсмології багато років. Вони дозволяють отримувати із сейсмологічних показників приховані закономірності, на основі яких можна прогнозувати землетруси. Виконання такої роботи людиною трудоємне і вимагає залучення фахівців високої кваліфікації і не завжди можливе.

Предметом дослідження є розробка інформаційної системи для аналізу передвісників землетрусів.

Об’єктом дослідження є передвісники землетрусів.

Мета роботи – розроблення інформаційної системи для аналізу передвісників землетрусів.

Відповідно до мети дослідження поставлено наступні **завдання**:

- провести огляд літератури по даній тематиці;
- проаналізувати існуючі системи прогнозування сейсмічних явищ;
- розробити математичну модель системи та алгоритм функціонування інформаційної системи;

- реалізувати програмну модель інформаційної системи;
- представити результати роботи інформаційної системи.

Наукова новизна одержаних результатів

Наукова обгрунтованість положень та висновків підтверджується використанням великого об’єму даних спостережень. Дослідження в області аналізу передвісників землетрусів активно використовують техніки глибокого машинного навчання для обробки та інтерпретації великого об’єму геологічних даних. З допомогою даної інформаційної системи Моделі можна виявляти зв’язки між різними параметрами та попереджати про можливі землетруси. Аналіз змін у поверхневій температурі, вологості ґрунту та інших параметрів може слугувати індикаторами можливих сейсмічних подій. Це дозволяє краще розуміти процеси, які передують землетрусам, та розробляти ефективніші методи передбачення. Враховуються не тільки класичні параметри, але й інші важливі чинники, такі як зміни гравітаційного поля, електромагнітні сигнали чи газовий викид. Це розширює спектр інформації, яку можна використовувати для передбачення землетрусів.

Практичне значення одержаних результатів

Результати аналізу передвісників землетрусів дозволяють розробляти системи попередження, що значно знижують ризики людських втрат та матеріальних збитків. Знання про можливі місця та часи землетрусів є ключовим фактором при проектуванні нових місць проживання, інфраструктури та будівель, сприяючи створенню більш стійких до сейсмічних впливів об’єктів. Результати досліджень передвісників землетрусів відкривають нові можливості для розвитку геологічних наук, сприяючи глибшому розумінню процесів, які призводять до землетрусів, та розробці більш точних моделей передбачення.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Прогнозування землетрусів

Активні тектонічні розломи призводять до землетрусів як результат викиду енергії під час швидкого ковзання вздовж лінії розлому. Але точно передбачити, коли відбудуться наступні поштовхи і наскільки сильними вони будуть, сейсмологи поки не здатні.

Проте в останні роки вчені досягли неабиякого прогресу в розробці систем раннього оповіщення про землетруси, в яких сейсмометри виявляють епіцентр і відправляють оповіщення прямо на телефони людей. Але, на жаль, поки що такі системи здатні сповіщати людей не за дні чи години, а лише за секунди до передбачуваного поштовху. Сейсмічні удари планети занадто раптові, але навіть кілька секунд фори можуть врятувати тисячі людей, надавши додатковий час для підготовки до землетрусу, що наближається. Йдуть роботи з покращення цих систем раннього оповіщення, але результат все одно буде близько кількох секунд, залежно від того, наскільки близько до епіцентру землетрусу буде знаходитися людина.

Новий метод називається розподіленим акустичним зондуванням чи distributed acoustic sensing (DAS). Хоча ця технологія ще знаходиться в стані розробки, DAS може підключитися до оптоволоконних кабелів для виявлення сейсмічних хвиль. Ці кабелі використовуються для телекомунікацій, але їх можна перепрофілювати для виявлення землетрусів, тому що рух землі трохи порушує рух світла, що проходить по кабелю, створюючи виразний сигнал.

Системи раннього оповіщення

Коли відбудуться землетруси, передбачити в принципі не можна. Будь-яка система, чи то сейсмометр чи оптоволоконний кабель, не може виявити речі до того, як вони відбудуться. Але їх можна виявити на ранній стадії та встигнути підготуватися. А для цього потрібно мати датчик,

розташований якомога ближче до епіцентру. Але оптоволоконних кабелів багато, і вони проходять майже скрізь. Отже, якщо можна було б використовувати всі оптоволоконні кабелі для виявлення землетрусів, то можна було б отримувати інформацію про потенційні епіцентри.

Первинні або P-хвилі рухаються зі швидкістю 6 км/с. Вони не завдають великої шкоди будинкам та іншій інфраструктурі. Вторинні хвилі, або S-хвилі, набагато небезпечніші, вони поширюються зі швидкістю 4 км/с. Ще більш руйнівними є поверхневі хвилі, які рухаються приблизно з тією самою швидкістю, як і S-хвилі. Вони поширюються поверхнею землі, що призводить до різкої деформації земної кори. Вони особливо руйнівні, тому що їх енергія концентрується на відносно плоскій поверхні вздовж поверхні, тоді як P-хвилі та S-хвилі поширюються більш тривимірно в товщі землі, розподіляючи і розсіюючи свою енергію.

Існуючі системи раннього попередження землетрусів (Earthquake early warning, EEW), такі як ShakeAlert Геологічної служби США, використовують сейсмометри для використання різних швидкостей сейсмічних хвиль. ShakeAlert включає близько 1400 сейсмічних станцій у Каліфорнії, Орегоні та Вашингтоні, і планує додати ще близько 300. Вони відстежують рухомі P-хвилі, які попереджають про більш небезпечні S-хвилі та поверхневі хвилі. Якщо відбувається землетрус і принаймні чотири окремі станції виявляють подію, цей сигнал відправляється до центру обробки даних. Якщо алгоритми системи визначать, що підземні поштовхи перевищують 4,5 бали, то буде зроблено екстрене сповіщення, яке буде відправлено на мобільні телефони місцевих жителів.

DAS працює за тим же принципом, що і ShakeAlert, тільки замість сейсмометрів, що відстежують P-хвилі, використовує великі ділянки оптоволоконних кабелів. Передача даних через сучасне телекомунікаційне обладнання відбувається зі швидкістю світла, що очевидно набагато швидше за сейсмічні хвилі. Але те, наскільки раніше отримають жителі попередження, залежить від того, наскільки вони далеко від епіцентру.

Якщо вони знаходяться зовсім близько, у них просто не буде достатньо часу, щоб отримати попередження раніше від землетрусу.

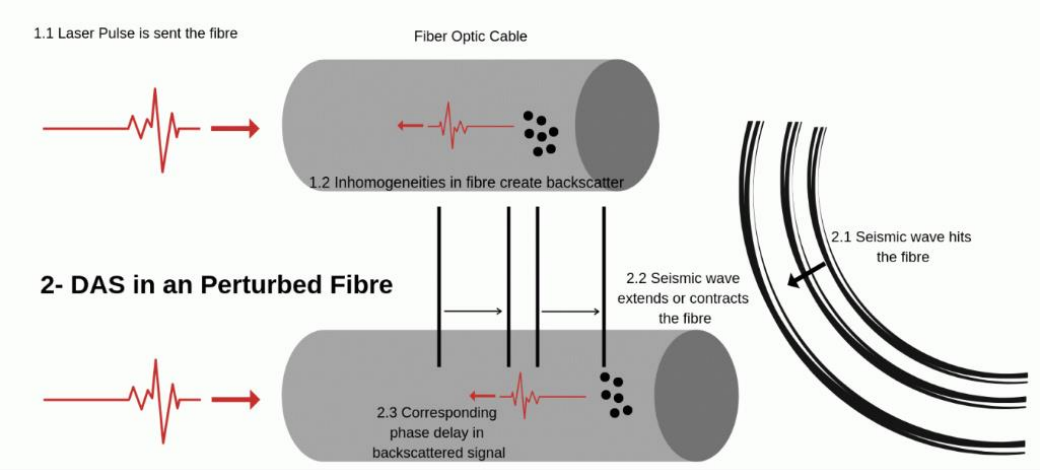

Рис. 1.1. Системи попередження про можливість землетрусу.

Телекомунікаційні компанії часто закладають більше кабелів, ніж їм необхідно. Вчені можуть отримати дозвіл на підключення пристрою опитувальника до кабелів, що не використовуються. Цей пристрій запускає лазерні імпульси по проводі та аналізує частину світла, яке відбивається, коли на волокно впливає деформація ґрунту. Оскільки вчені знають швидкість світла, вони можуть точно визначати збурення, ґрунтуючись на часі, який знадобився сигналу, щоб повернутися до опитувальника. Коли сейсмічні хвилі впливають на кабель, неоднорідності вздовж волокна змінюють своє положення. Цей метод дозволяє перетворити будь-яке оптичне волокно на масив сейсмоакустичних датчиків, проводячи вимірювання з просторовою і часовою роздільною здатністю.

Замість проведення сейсмічних вимірювань в одній точці, як це робить сейсмометр, DAS більше схожий на струну довжиною в кілька кілометрів, яка утворює один гігантський датчик землетрусів. Однією з великих переваг DAS є те, що багато таких кабелів вже є, тому вони легко доступні. DAS також може збирати дані там, де немає відповідних сейсмічних станцій, наприклад у сільській місцевості, де під ними протягнуті оптоволоконні кабелі. Оскільки ці кабелі також перебувають

під водою, вони проходять вздовж берегової лінії і з'єднують континенти через океани, вони можуть вловлювати землетруси.

У 2022 році вчені використовували кабель, що простягся від Великобританії до Канади, для виявлення землетрусів у Перу. Цей метод був настільки чутливим, що кабель вловлював навіть рух припливів, а це означало, що його потенційно можна використовувати для виявлення цунамі, спричинених підводними землетрусами.

Оскільки традиційний сейсмометр вимірює в одній точці, він може спотворюватися локальним шумом даних, наприклад, викликаним великими транспортними засобами, що проїжджають повз. Якщо є волокна, можна досить легко відрізнити землетрус від шуму, оскільки землетрус майже миттєво реєструється на відстані сотень метрів. Якщо це якесь локальне джерело шуму, наприклад, автомобіль, поїзд або ще щось, це видно лише на декількох десятках метрів.

DAS значно підвищує точність сейсмічних даних. Це не означає, що він замінить ці сейсмометри та інші прилади, швидше, доповнить їх. Загальна ідея полягає в тому, щоб просто розмістити більше сейсмічних детекторів ближче до епіцентрів землетрусів, покращивши охоплення.

Дослідження DAS мають кілька проблем, з якими потрібно боротися, зокрема, з тим, що оптоволоконні кабелі не були призначені для виявлення сейсмічної активності, вони були розроблені для передачі інформації. Оптоволоконні лінії можуть бути прокладені рандомно в трубопроводі, в той час як сейсмометр точно налаштований і розташований для виявлення землетрусів. Вчені досліджують, як збирання даних по кабелю може змінюватися в залежності від того, як він прокладений під землею.

Google розробила систему, яка буде сповіщати користувачів Android про землетрус, що починається. Для виявлення землетрусів система використовує вбудований смартфон акселерометр. Землетруси відбуваються по всьому світу щодня. Своєчасне попередження про стихійне лихо може допомогти людям підготуватися. Однак розгортання

повноцінної інфраструктури для сповіщень може бути дуже витратним для багатьох регіонів. Компанія планує використовувати смартфони на Android, якими сьогодні користуються мільйони людей, як сейсмодатчики. Більше того, смартфони можуть моніторити сейсмічну активність у регіоні, що перетворить їх на найбільшу у світі мережу виявлення землетрусів.

В усіх смартфонах є акселерометри. Їх можна використовувати для виявлення ознак землетрусу. Дані, які отримує датчик акселерометра, можна відправити на сервер виявлення землетрусів разом із приблизними координатами, де сталися поштовхи. Сервер, обробляючи дані, які отримані з різних смартфонів, визначить, чи справді в тій чи іншій області відбувається землетрус. Попередження навіть за кілька секунд може мати значення. Людина може встигнути знайти укриття або хоча б впасти і вхопитися за щось до того, як почнеться землетрус. Дані про сейсмічну активність згодом відображатимуться в пошукових системах Google.

1.2. Передбачення землетрусів з допомогою методів штучного інтелекту

Для передбачення виникнення землетрусів було застосовано алгоритм машинного навчання до передвісників землетрусів. Озброївшись даними для навчання, використали алгоритм машинного навчання під назвою random forest для систематичного пошуку комбінацій ознак, явно пов’язаних з кількістю часу, що залишився до поштовхів. Вивчивши експериментальні дані, алгоритм міг розпочати прогнозувати час поштовхів з урахуванням акустичних ознак.

Алгоритм випадковий ліс вибрали для передбачення часу, що залишився до нових поштовхів, зокрема тому, що (в порівнянні з нейромережами та іншими популярними алгоритмами машинного навчання) випадковий ліс відносно легко інтерпретувати. Працює алгоритм, по суті, як дерево рішень, у якому кожна гілка поділяє набір

даних з урахуванням якоїсь статистичної ознаки. Тому в дереві зберігаються записи того, які ознаки алгоритм використовував для передбачень – і відносна важливість кожної з ознак, які допомагали алгоритму дійти певного передбачення.

Десятиліттями люди намагалися передбачати землетруси, ґрунтуючись на попередніх поштовхах та інших ізольованих сейсмічних подіях. Цей результат передбачає, що всі вони шукали не там – і що ключем до передбачень була менш явна інформація, яку можна зібрати під час відносно спокійних періодів між великими сейсмічними подіями. Було створено сейсмічний каталог: розбили сейсмічні дані на невеликі сегменти і описали кожен із них набором статистичних ознак. Потім вони надали ці дані та інформацію про те, коли відбувалися попередні повільні землетруси, алгоритму машинного навчання.

Потренувавшись на даних, алгоритм зміг успішно передбачати повільні землетруси на основі даних, записаних за кілька місяців до кожної події. Ключовим чинником виступала сейсмічна енергія – величина, що близько пов’язана з дисперсією акустичного сигналу. Як і дисперсія, сейсмічна енергія характерно зростала напередодні кожного землетрусу.

Метою передбачення землетрусів ніколи не було передбачення повільних землетрусів. Усім треба передбачати раптові та катастрофічні поштовхи, що загрожують життю та здоров’ю. Для машинного навчання це, здавалося, є парадоксом: найбільші землетруси, які сейсмологам найбільше хотілося б передбачати, трапляються найрідше.

Сейсмічні закономірності, що передують невеликим землетрусам, статистично схожі на ті, що передують великим, а в одному розломі будь-якого дня можуть статися десятки дрібних землетрусів. Навчавшись на тисячах цих маленьких поштовхів, комп’ютер, можливо, зможе передбачати й великі. Також алгоритми машинного навчання зможуть навчатися на комп’ютерних симуляціях швидких землетрусів, які зможуть стати заміною на реальні дані.

Вчені стикаються з тим, що хоча фізичні процеси, що призводять до розлому на межі землетрусу, і можуть стати передбачуваними, саме виникнення землетрусу – зростання невеликих сейсмічних збурень, що призводять до повномасштабного розриву розлому, містить елемент випадковості. Якщо це так, то, незалежно від якості навчання машин, вони, можливо, ніколи не зможуть передбачати землетруси так, як вчені змогли передбачати інші природні катастрофи.

У кращому разі прогнози великих землетрусів видаватимуть часові інтервали тривалістю тижні, місяці чи роки. Такі прогнози не можна буде використовувати, наприклад, для організації масової евакуації міст напередодні поштовхів. Але вони можуть покращити підготовку до цієї події, допомогти сконцентрувати зусилля на зміцненні небезпечних будівель та іншим чином зменшити небезпеку землетрусу.

1.3. Система ShakeAlert раннього попередження про можливість землетрусу

Геологічна служба США (United States Geological Survey, USGS) розробила систему раннього попередження про землетруси (Earthquake warning system, EEW) під назвою ShakeAlert для зон підвищеного ризику. Мета системи полягає у зменшенні впливу землетрусів та порятунку людей та власності, роблячи масові оповіщення.

Використовуючи мережу датчиків руху ґрунту та складні комп’ютерні алгоритми, ShakeAlert може виявити землетрус, обчисливши його епіцентр та інтенсивність поштовхів, що дозволяє вжити відповідних захисних заходів. Завдання ShakeAlert - швидкий та надійний збір та доставка даних із датчиків до центрів обробки для подальшого оповіщення населення, щоб зменшити збитки та врятувати життя людей.

Подібно до інших систем раннього попередження про землетруси, ShakeAlert не передбачає землетрусу, а намагається швидко ідентифікувати сейсмічну подію і видавати попередження до того, як

відчується поштовхи. Вона робить це, виявляючи P-хвилі, що швидко рухаються, потім обчислює місце розташування і оцінює магнітуду землетрусу, після чого видає попередження. Залежно від відстані людини від епіцентру землетрусу, попередження може досягти його раніше, ніж це зроблять повільніші S-хвилі.

Системи EEW працюють не лише у США. Громадське оповіщення зроблено по всій країні в Японії та на Тайвані, а також у деяких частинах Мексики, Китаю та Кореї. Туреччина, Італія та Румунія роблять більш обмежене сповіщення для захисту інфраструктури. Декілька країн активно працюють над такими системами, такі як Індія, Ізраїль та Чилі.

Задіяно загалом 1675 високоякісних сейсмічних станцій ANSS (Advanced National Seismic System) з можливістю EEW у реальному часі: 1115 з них у Каліфорнії та 560 на північному заході Тихого океану. Відстань між станціями становить 10 кілометрів у міських районах, 20 км у районах із сейсмічними джерелами, що становлять небезпеку для населених пунктів, та 40 км в інших районах.

ShakeAlert являє собою набір географічно розподілених, але взаємопов’язаних компонентів для виявлення землетрусу настільки швидко, щоб оповіщення були надіслані людям та центрам обробки, що дозволяють вживати захисних заходів перед сильними поштовхами. ShakeAlert надає інформацію в реальному часі і представляє в декількох варіантах для задоволення різних користувачів. Вихідне рішення та оцінка переглядаються в режимі реального часу в міру зростання розлому і в міру того, як дані про рух землі стають доступними зі станцій, які розташовані поруч із розривом, і в міру отримання додаткових даних від віддалених станцій.

Алгоритми системи можуть видавати два типи рішення для джерел землетрусів. Перше – це точкове рішення (earthquake point-source integrated code, EPIC), що описує осередок землетрусу та магнітуду. Другий – рішення з лінійним вихідним кодом (finite-fault detector, FinDer), яке

зображує розрив розлому у вигляді лінії на земній поверхні разом з його величиною.

1.4. Прогнозування виникнення землетрусу по електромагнітних коливаннях в іоносфері Землі

У періоди від кількох хвилин за кілька днів до землетрусу як в земній корі, так і в атмосфері відбуваються електромагнітні флуктуації. Їх виявлення та обробка можуть дозволити сейсмологам ефективно передбачати природні катастрофи. Чим сильніший землетрус, тим раніше з'являються ці атмосферні аномалії. Залишається відкритим питання, як рух земної кори призводять до появи електромагнітного випромінювання. Вважають, що шари землі, що нагріваються від тиску, здатні породжувати позитивні заряди.

Ці ж ефекти пояснюють свідчення про незвичайні явища, що передують землетрусам, таких, як свічення, що вириває́ться з-під землі, та неполадки магнітного компасу в зоні землетрусу. У зв'язку з цим сейсмологи поєдналися з проектом Європейського космічного агентства під назвою SWARM, що вивчає магнітне поле Землі. Об'єднані дані від SWARM, а також дані з GPS-супутників та сейсмометрів можуть допомогти у розробці технології передбачень майбутніх землетрусів.

Сейсмологи вже давно намагаються знайти зв'язок між іоносферою та землетрусами. Вони вважають, що навіть якщо передбачити землетрус, це зовсім не означає, що можна буде передбачити його силу і тривалість. Цілком можливо, що сильні землетруси починаються так само, як і слабкі. Поки що, незважаючи на значні зусилля сейсмологів у дослідженнях, неможливо дати такий прогноз землетрусів із точністю до дня чи навіть місяця. Вчені досі не знають всіх деталей фізичних процесів, які пов'язані із землетрусами, та методи, якими їх можна точно передбачати.

На думку Сейсмологічного співтовариства Америки, метод прогнозу, який можна було б назвати вірним, повинен описати очікувану магнітуду з

певним допустимим відхиленням, добре визначену зону епіцентру, діапазон часу, в який відбудеться ця подія, і ймовірність того, що вона дійсно відбудеться. Дані, на яких базується прогноз, повинні піддаватися перевірці.

Іоносфера – це шар атмосфери, який сильно іонізований внаслідок опромінення космічними променями. На нашій планеті це верхня частина атмосфери, що складається з суміші газу нейтральних атомів, молекул та плазми. Коли відбувається розлом, він породжує цілу послідовність різних сейсмічних хвиль. Довгі хвилі низької частоти можуть поширюватися на велику відстань від джерела і гойдати високі будови. Високочастотні хвилі розгойдують будинки та мости, а іноді й повністю їх руйнують.

Згідно з традиційними моделями землетрусів, коли блок земної кори починає ковзати і тертися об інший, тертя між ними породжує сейсмічні хвилі. Сейсмологи визнають спрощеність цих моделей у порівнянні з реальними процесами, що відбуваються в районі лінії розлому. Однак вони точно описують низькочастотний компонент хвильового набору землетрусу – критично важливий ранній індикатор магнітуди землетрусу та життєво необхідну інформацію.

Однак традиційні моделі не в змозі пояснити велику кількість високочастотних хвиль, що породжуються землетрусом. Це стає проблемою, коли намагаються розібратися, чому певні тріщини виявляються руйнівнішими. У традиційних моделях високочастотні хвилі пов’язують із коливаннями розлому – непередбачуваними рухами тріщини, які то виникають, то загасають.

Геологічна служба США (USGS) запропонувала створити мережу попередження про землетруси на межі двох технологій – краудсорсингу та смартфонів. Додаток, що відстежує вібрації смартфонів тисяч учасників і будує карту поштовхів, зможе, в перспективі, попереджати про катаклізм, що наближається. Служба, яка має кілька станцій спостереження, пропонує об’єднати зусилля для попередження про землетруси, яке

допоможе зберегти життя людей, які перебувають у небезпечних районах. Кожен пристрій міг би аналізувати дані, що надходять, визначати, чи вказують вони на можливість землетрусу. Якщо досить велика кількість телефонів одночасно надішле на сервер схожі повідомлення, це стане приводом для сповіщення про тривогу.

Смартфони вже почали поширюватися у багатьох небагатих регіонах планети – там, де дороге сейсмологічне обладнання поки що просто відсутнє. Така попереджувальна система може бути дуже доречною. Ідея використовувати смартфони як сейсмічні датчики вже спадала на думку винахідникам. Існує проект iShake з додатком для iPhone, який розробляє таку саму систему.

На жаль, сучасна наука не в змозі прогнозувати землетруси. Відсутні достовірні фізико-математичні моделі для таких розрахунків. А навіть якби точність вимірювань і неіснуюча поки модель сейсмічного процесу дали можливість з достатньою точністю визначити місце і час початку руйнування ділянки земної кори, магнітуда майбутнього землетрусу все одно залишилася б невідомою.

ВИСНОВКИ ДО РОЗДІЛУ 1

В першому розділі проаналізовано предметну область, засоби та технології проектування інформаційної системи оцінки передвісників землетрусу. Розглянуто існуючі системи, з допомогою яких можна оцінити ймовірність виникнення землетрусів. Дані сучасні системи раннього оповіщення є перспективними шляхами забезпечення безпеки. Ці технології використовують оптоволоконні кабелі для виявлення сейсмічних хвиль та дозволяють відправляти оповіщення заздалегідь. Незважаючи на складнощі точного прогнозування, постійний розвиток технологій сприяє зменшенню ризиків та захисту населення.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Методи інтелектуального аналізу даних

Метою DataMining є знаходження таких моделей, які не можуть бути знайдені звичайними методами. Існує два види моделей: передбачувальні та описові.

Передбачувальні моделі: позиціонуються на наборі даних з відомими результатами. Вони використовуються для прогнозування результатів на підставі інших наборів даних. Це моделі класифікації (описують правила, за якими можна віднести опис об’єкта до одного з класів) і моделі послідовностей (вони описують функції, якими можна прогнозувати зміну неперервних числових параметрів).

Описові моделі: вони приділяють особливу увагу суті залежностей набору даних, взаємному впливу різних факторів, побудові емпіричних моделей. Відповідно до класифікації по стратегіях, завдання Data Mining поділяються на такі групи:

- навчання з учителем;
- навчання без вчителя;
- інші.

Категорія навчання з учителем представлена такими завданнями Data Mining: класифікація, оцінка, прогнозування. Категорія навчання без учителя представлена завданням кластеризації. До категорії інші входять завдання, не включені до попередніх двох стратегій. Базовими методами, які може знайти технологія DataMining, є:

1. Асоціація – застосовується, коли кілька подій пов’язані між собою.

2. Класифікація – виявлення рис, які характеризуватимуть групу, до якої належить об’єкт, на основі навчання на вже класифікованих об’єктах. Для вирішення задач класифікації можуть використовуватися такі методи: найближчого сусіда (Nearest Neighbor), k-найближчого сусіда (k-Nearest

Neighbor), байєсівські мережі (Bayesian Networks), нейронні мережі (Neural Networks).

3. Кластеризація – відрізняється від класифікації тим, що групи заздалегідь невідомі і Data Mining самостійно виявляють різні однорідні групи даних.

4. Послідовність – застосовується при існуванні ланцюга подій, які пов’язані між собою в часі.

5. Прогнозування – створення чи знаходження шаблонів, які будуть показувати тенденцію поведінки необхідних показників у часових рядах. З них можна передбачити поведінку системи у майбутньому. Для вирішення таких завдань широко застосовуються методи математичної статистики, нейронні мережі та інші.

Часто параметри за замовчуванням виявляються для алгоритмів, реалізованих у сучасних потужних бібліотеках машинного навчання, достатніми для досягнення результату, що підтверджує можливість їх використання різними фахівцями, які не мають глибоких знань в алгоритмах машинного навчання. Але таким фахівцям потрібно надати доступ до інструментів, для цього призначена аналітична система, що буде розроблятися. Різні інструменти аналізу даних представлені в різних програмних бібліотеках Python.

Система повинна мати у своєму складі інструменти інтелектуального аналізу даних. При цьому слід розділити частину системи, яка відповідає за введення даних та аналіз даних. Для навчання та перевірки вибірку необхідно розділити у співвідношенні 70/30 на навчальну та тестову відповідно та виконати класифікацію з використанням класифікаторів Naïve Bayes, Decision Tree, Random Forest, Neural Network, KNN. Моніторинг інформаційної системи здійснюється за такими параметрами: правильність (accuracy), точність (precision), повнота (recall), f-міра (f-score).

f-міра є основним параметром порівняння. Інші величини взяті для побудови повної картини.

2.2. Моніторинг сейсмічних передвісників для прогнозу землетрусів

Поняття передвісника використовується в різних областях знань і означає явище, якек передує деякій події і обов'язково пов'язане з цією подією. Підготовка тектонічного землетрусу викликає збурення, які виділяються на фоні випадкових варіацій, різноманітних геофізичних полів, які трактуються як передвісники землетрусу. Передвісниками змелетрусу називають зміни геофізичнх полів, які спричинені процесом підготовки землетрусу.

Інформаційно-аналітична система прогнозу землетрусів розробляється з метою узагальнити дані по передвісниках майбутніх поштовхів землі та скоректувати регіональні коефіцієнти закономірностей прояву передвісників. Землетруси являються одним з найбільш небезпечних природних явищ, в зв'язку з цим теорії виникнення передвісників, які використовуються для прогнозу землетрусів, надають великого значення. Землетруси повторюються в одних і тих же сейсмоактивних районах, але, очевидно, не підлягають якійсь закономірності. Накопичення енергії та напруженого стану земної кори, які фіксуються вздовж активних розломів, дають основу прогнозувати в цьому місці землетрус. Однак для того, щоб передбачити час поштовхів та епіцентр землетрусу, потрібно обробити велику кількість незалежних даних.

Намагаючись отримати відомості для надійного прогнозування землетрусів, використовують такі прилади: деформографи, гравіметри, лазерні дальноміри, магнітометри та інші. Серед об'єктів досліджень – регіональні зміни магнітного поля Землі, електропровідність, часові зміни швидкості сейсмічних хвиль, геодезичні дані, флуктуації рівня води у

свердловинах, вміст радону в грунтових водах, аномальна поведінка тварин, а також тривала сейсмічна історія.

Основою теорії прогнозу землетрусів являються кореляційні залежності між часом прояву передвісників та енергією землетрусу. Передбачається, що передвісники виникають одночасно на різних відстанях від епіцентру майбутнього землетрусу. Але процес підготовки землетрусу відбувається не тільки в часі, але й в просторі, тому тривалість передвісників залежить від відстані пункта спостереження до епіцентру землетрусу.

Всі короткотривалі передвісники – це свідчення певного фізичного процесу в зоні можливого землетрусу, незалежно від методу спостережень вони повинні проявлятися в один і той же момент часу. Тривалість короткотермінових передвісників складає декілька годин. Тривалість процесу накопичення енергії, що являється причиною появи передвісників, збільшується з ростом магнітуди землетрусу.

Розвиток комп’ютерних технологій надає можливість їх застосування в різних областях науки та техніки. Прогноз землетрусів не є винятком. У зв’язку з цим на даний момент існують системи, які дозволяють оперативно в режимі реального часу проводити обробку геофізичних даних, що призводить до більш якісного прогнозу надзвичайних ситуацій. Також спостерігається тенденція переходу запису сигналів від аналогового виду в цифровий. Такі тенденції зближують дві області: комп’ютерну та геофізичну.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі приведено відомості про особливості проектування інформаційної системи засобами Python, обробки даних, візуалізації результатів. Data Mining полягає у виявленні моделей, недосяжних звичайними методами. Передбачувальні моделі служать для прогнозування результатів на основі відомих даних, в той час як описові моделі акцентують увагу на залежностях та взаємодіях факторів. Розглянено методи класифікації завдань Data Mining за стратегіями: навчання з учителем, навчання без учителя та інші. Основні методи включають асоціацію, класифікацію, кластеризацію, послідовність та прогнозування. Завдання Data Mining допомагають аналізувати дані та робити прогнози на основі навчальних алгоритмів та статистичних методів.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Математичне моделювання землетрусів

При прогнозуванні землетрусів розглядають питання визначення абсолютних значень елементів руху точок, які тим чи іншим чином закріплені на поверхні землі. Ці значення мають деякий довірчий інтервал, тобто мають ймовірнісний характер. Опорні пункти, по відношенню до яких визначають зміщення точок, не являються абсолютно стабільними з часом. Спостереження намагаються організувати так, щоб отримані відносні зміщення найкращим чином відображали абсолютні величини зміщень. При прогнозуванні розглядають питання про види рухів, які виникають при порушенні стійкості точок. Для цього виділяють на поверхні елементарну площу ΔS і розміщують на ній початок системи нерухомих координат XYZ.

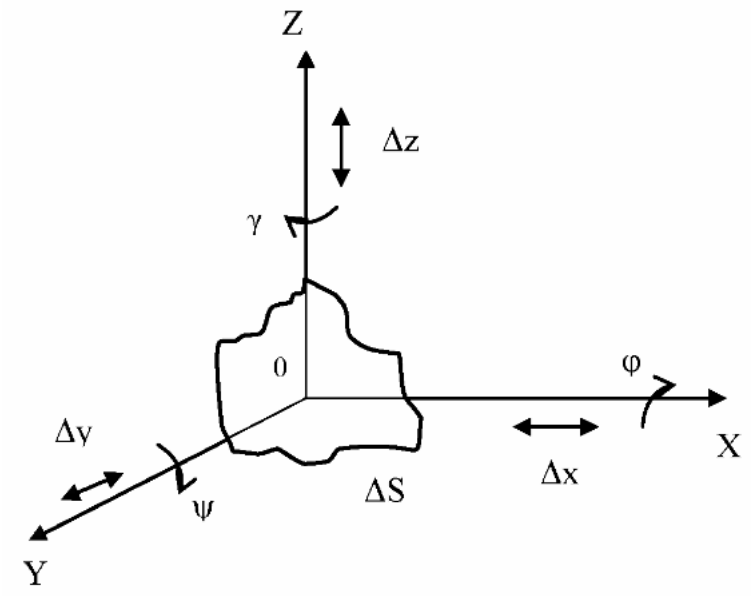

Рис. 3.1. Елементи руху точки земної поверхні.

Обертовий рух площадки ΔS біля осей OX та OY викликає її нахили по відношенню до осі OZ, а обертання біля осі OZ – нерівномірні зміщення її точок в горизонтальній площині. В процесі деформацій окремі точки деякого блоку мають ще й взаємні переміщення одне відносно одного, що

в свою чергу порушує цілісність цього блоку. Такі зміщення викликають зміну напружено-деформованого стану землі.

Перейшовши до координат якої-небудь точки, шлях, який пройдений нею за проміжок часу $\Delta t = t_2 - t_1$, отримують за формулою:

$$I = \sqrt{\left[x(t_2) - x(t_1)\right]^2 + \left[y(t_2) - y(t_1)\right]^2 + \left[z(t_2) - z(t_1)\right]^2}\,, \tag{3.1}$$

де $x(t_1)$, $y(t_1)$, $z(t_1)$ – координати деякої точки в момент часу; $x(t_2)$, $y(t_2)$, $z(t_2)$ – те саме в момент часу t_2.

Якщо в момент часу t_1 відстань між двома точками була $I(t_1)$, а через деякий проміжок Δt в момент часу t_2 виявилася рівною $I(t_2)$, причому $I(t_2) \neq I(t_1)$, то взаємне переміщення за цей же проміжок часу Δt буде $\Delta I = I(t_2) - I(t_1)$.

Для прогнозування землетрусу проводять спостереження за деформаціями землі для визначення швидкості руху точок земної поверхні:

$$V = \frac{\Delta S}{\Delta t}. \tag{3.2}$$

Використавши методи теорії похибок вимірювання, після перетворень середнє квадратичне відхилення швидкості зміщень можна представити у виді:

$$\sigma(v) = \frac{1}{\Delta t}\sqrt{\sigma^2(\Delta S)v^2\sigma^2(\Delta t)}\,. \tag{3.3}$$

$\sigma(\Delta S)$ – середнє квадратичне відхилення досліджуваної точки; $\sigma(\Delta t)$ – середнє квадратичне відхилення часу між циклами спостережень. Тоді відносна похибка визначення швидкості зміщень буде:

$$\frac{\sigma(v)}{v} = \sqrt{\frac{\sigma^2(\Delta S)}{\Delta S^2} + \frac{\sigma^2(\Delta t)}{\Delta t^2}}\,. \tag{3.4}$$

При проектуванні спостережень за зміщеннями виникають задачі визначення потрібної точності спостережень зміщень ΔS періоду спостережень Δt та оцінки точності його $\sigma(\Delta t)$ по заданих значеннях. З виразів видно, що число рівнянь, які пов’язують шукані та задані

величини, виявляється недостатнім для однозначного визначення вказаних параметрів.

Нехай значення відносної похибки швидкості задовільняє заданій умові:

$$\sigma(v)/v \le 1/K. \tag{3.5}$$

Тоді з врахуванням виразу можна записати:

$$\frac{\sigma^2(\Delta S)}{\Delta S^2} + \frac{\sigma^2(\Delta t)}{\Delta t^2} \le \frac{1}{K^2}. \tag{3.6}$$

Скориставшись принципом рівного впливу джерел похибок, для визначення вказаних вище параметрів отримують такі вирази:

$$\begin{aligned} &\Delta t \le K\sigma(\Delta t)\sqrt{2}; \\ &\sigma(\Delta t) \le \frac{\Delta t}{K\sqrt{2}}; \quad . \\ &\sigma(\Delta S) \le \frac{v\Delta t}{K\sqrt{2}} \end{aligned} \tag{3.7}$$

Таким чином, задаючись деяким приємливим для конкретних умов спостережень середнім квадратичним відхиленням, можна отримати необхідний період спостережень за зміщеннями. Якщо є хоча би наближена уява про хід зміщення з часом, то визначають період часу, протягом якого зміщення змінюються рівномірно. Зміщення точок земної поверхні мають характер нерівномірного руху. В початковий період розвитку процесу швидкість зміщень поступово наростає і за якийсь проміжок часу вступає в катастрофічну фазу. Потім швидкість зменшується і переходить в стадію тимчасової стабілізації. Тому середнє квадратичне відхилення $\sigma(\Delta S)$ потрібно визначати для деякої заданої ступені ΔS. Потрібно враховувати і той факт, що швидкість зміщень нерівномірна.

Якщо ввести умову:

$$\sigma(v) \le \sigma_0(v), \tag{3.8}$$

де $\sigma_0(v)$ – граничне значення середнього квадратичного відхилення швидкості, то отримують:

$$\begin{aligned} &\sigma(\Delta t) \le \sigma\left(\Delta S\right)/v \\ &\sigma\Delta S \le \frac{\sigma_0(v)}{\sqrt{2}}\Delta t \end{aligned} \quad . \tag{3.9}$$

Під прогнозом землетрусів розуміють визначення місця, часу та сили (магнітуди) землетрусу. По часу прогноз поділяється на довготерміновий (на десятиріччя вперед), середньотерміновий (на роки вперед), короткотерміновий (на дні-місяці вперед) та оперативний (на хвилини-години вперед). Кожний етап прогнозу базується на певному наборі передвісників землетрусу – геофізичних явищах, які випереджають і попереджають про можливість виникнення землетрусу. Нараховують декілька сотень різних по своїй природі передвісників землетрусів. Їх можна поділити на дві групи.

Перша – це передвісники, які пов’язані із закономірною поведінкою різних геофізичних полів на різних етапах підготовки землетрусу. Передвісники цієї групи покривають практично весь діапазон прогнозу по часу: від довготермінового до оперативного. Друга група – це передвісники, які пов’язані з незвичною поведінкою біологічних об’єктів перед виникненням землетрусу. Ця група передвісників менш вивчена, ніж перша. Їх можна віднести до короткотермінових та оперативних.

В свою чергу геофізичні передвісники поділяються на сейсмічні, гідрогеодинамічні, деформаційні, геохімічні, термічні, гравітаційні, електромагнітні. Не дивлячись на велику кількість передвісників, ні один з них не дає точних даних на час, місце та силу майбутнього землетрусу. В різних сейсмоактивних районах різні передвісники працюють по-різному, даючи великий розкид в оцінках місця, часу та сили майбутнього землетрусу.

Тому прогноз землетрусів по своїй природі має ймовірнісний характер. Повідомлення про передвісників землетрусу являються

одиничними і по них важко, а інколи неможливо оцінити їх статистичні характеристики: ймовірність правильного прогнозу, середнього часу очікування землетрусу після виникнення передвісника.

Аналіз багаторічних даних по ряду геофізичних пеердвісників показав, що ймовірність успішного прогнозу по кожному з них не перевищує 0,5. Одним з можливих виходів з цієї ситуації являється спільне використання декількох прогностичних ознак. Комплексне їх використання дозволить підвищити надійність та ефективність прогнозних оцінок.

Швидкості вертикальних рухів земної кори (мм/рік) визначаються за співвідношеннями:

$$\begin{aligned} \Delta V &= \left(h_2 - h_1\right) / \Delta T \\ \Delta T &= T_2 - T_1 \end{aligned} \quad . \tag{3.10}$$

де h_2 та h_1 – перевищення між суміжними знаками (мм/ рік), які отримані в T_2 та T_1 (роки); ΔT – інтервал часу в роках між повторними вимірами.

Великі простори земної кори, в яких відбуваються розриви і виникають тектонічні деформації, породжують сильні землетруси: чим менший об’єм епіцентру, тим слабші сейсмічні поштовхи. Гіпоцентром чи фокусом землетрусу називають умовний центр на глибині, а епіцентром – проекцію цього гіпоцентра на поверхню Землі.

Осередок землетрусу характеризується інтенсивністю сейсмічного ефекту, який виражається в балах та магнітуді. Використовують 12-бальну шкалу інтенсивності землетрусу. Згідно цієї шкали прийнята така градація інтенсивності землетрусів: I-III бали – слабі; IV-V – відчутні; VI-VII – сильні (пошкоджуються старі будови); VIII – руйнівні; IX – спустошливі; X – нищівні; XI – катастрофічні; XII – згубні катастрофи.

Самі сильні із зареєстрованих землетрусів досягають величини 8,5-9 по шкалі Ріхтера. На даний час оцінка землетрусів в магнітудах застосовують частіше, ніж в балах.

Енергія, яка виділяється при землетрусах:

$$E = p^2 rV(a/T), \quad (3.11)$$

де V – швидкість поширення сейсмічних хвиль, r – густина верхніх шарів землі, a – амплітуда зміщення, T – період коливань. Нижче представлено співвідношення для зв’язку між енергією землетрусу та його магнітною складовою по шкалі Ріхтера:

$$\log E = 9{,}9 + 1{,}9M - 0{,}024M^2. \quad (3.12)$$

Дана формула показує дуже велике зростання енергії при збільшенні магнітуди землетрусу.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі приведено дані про принципи побудови алгоритму математичної моделі та підходів до аналізу передвісників землетрусів. На основі приведеної математичної моделі спроектовано інформаційно-аналітичну систему. Проведені дослідження та моделювання підтверджують достовірність розробленої математичної моделі інформаційної системи.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Розроблення інформаційно-аналітичної системи передвісників землетрусу

Встановлюють необхідні бібліотеки. Це бібліотеки Numpy, Pandas, Matpotlib та Seaborn – для графічного представлення та аналізу наявних даних. Команда %matplotlib inline дозволяє відображати графіки безпосередньо в Jupyter Notebook або веб-інтерфейсі Jupyter.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Наступні два рядки імпортують необхідні підмодулі з бібліотеки scikit-learn для розбиття даних на навчальний та тестовий набори, а також для стандартизації ознак:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Така стандартизація допомагає моделі працювати більш ефективно, особливо для алгоритмів, які чутливі до масштабування ознак. Ці імпорти є дуже важливими для підготовки даних перед навчанням моделі. Вони допоможуть правильно розділити дані та забезпечити оптимальні умови для роботи з алгоритмами машинного навчання.

Далі імпортують різні класифікатори (моделі для задач класифікації) з бібліотеки Scikit-Learn. Ці класифікатори представляють різні підходи до вирішення задач класифікації. Вибір конкретного класифікатора залежить від характеру даних та завдання, над яким працюють.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier,
```

Також імпортують функції для обчислення різних метрик для оцінки якості моделі класифікації. Ці метрики дозволяють оцінити якість прогнозів, зроблених моделлю класифікації. Вони корисні для розуміння,

наскільки добре модель працює на різних аспектах класифікації, таких як точність, повнота і т.д.

```
from sklearn.metrics import f1_score,accuracy_score, recall_score,
precision_score
```

Наступний фрагмент коду використовує бібліотеку Pandas для завантаження даних з CSV-файлу, виведення перших п’яти рядків та створення копії датафрейму. Після виконання цих кроків будемо мати завантажені дані у df, виведені перші п’ять рядків для огляду та створену копію df1, з якою далі можна буде працювати без впливу на вихідний набір даних.

```
df = pd.read_csv('/content/earthquake_data.csv')
df.head().T
df1=df.copy()
```

З допомогою команди `alerts = df["alert"].unique()` отримують унікальні значення зі стовпця alert в df. alert може бути стовпцем, який містить різні види попереджень або сигналів, які пов’язані із землетрусами, такі як green, yellow, red і т.д.

З допомогою функції df.head() можна вивести перші п’ять рядків з датафрейму df. Після цього видно структуру датафрейму: кількість рядків, стовпців, які дані в ньому містяться. Кожен рядок представляє собою окремий землетрус і містить значення різних ознак у стовпцях.

	title	magnitude	date_time	cdi	mmi	alert	Earthquake	sig	net	nst	dmin	gap	magType	depth
0	M 7.0 - 18 km SW of Malango, Solomon Islands	7.0	22-11-2022 02:03	8	7	green	1	768	us	117	0.509	17.0	mww	14.000
1	M 6.9 - 204 km SW of Bengkulu, Indonesia	6.9	18-11-2022 13:37	4	4	green	0	735	us	99	2.229	34.0	mww	25.000
2	M 7.0 -	7.0	12-11-2022 07:09	3	3	green	1	755	us	147	3.125	18.0	mww	579.000
	M 7.3 - 205 km		11 11											

Рис. 4.1. Фрагмент набору даних.

З допомогою функції df.columns отримують заголовки стовпців у датафреймі:

```
Index(['title', 'magnitude', 'date_time', 'cdi', 'mmi', 'alert', 'Earthquake',
       'sig', 'net', 'nst', 'dmin', 'gap', 'magType', 'depth', 'latitude',
       'longitude', 'location', 'continent', 'country'],
      dtype='object')
```

Функція df.info() виводить інформацію датафрейм, включаючи загальну кількість записів, типи даних для кожного стовпця та кількість ненульових значень. Це допомагає отримати загальне уявлення про набір даних та їх структуру:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 782 entries, 0 to 781
Data columns (total 19 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   title       782 non-null    object
 1   magnitude   782 non-null    float64
 2   date_time   782 non-null    object
 3   cdi         782 non-null    int64
 4   mmi         782 non-null    int64
 5   alert       415 non-null    object
 6   Earthquake  782 non-null    int64
 7   sig         782 non-null    int64
 8   net         782 non-null    object
 9   nst         782 non-null    int64
 10  dmin        782 non-null    float64
 11  gap         782 non-null    float64
 12  magType     782 non-null    object
 13  depth       782 non-null    float64
 14  latitude    782 non-null    float64
 14  latitude    782 non-null    float64
 15  longitude   782 non-null    float64
 16  location    777 non-null    object
 17  continent   206 non-null    object
 18  country     484 non-null    object
dtypes: float64(6), int64(5), object(8)
memory usage: 116.2+ KB
```

Ця інформація може допомогти зрозуміти, які дані присутні в датафреймі, чи потребує виправлення недостачі або невідповідності даних та визначити, які колонки можуть бути корисними для подальшого аналізу.

Команда df.isna().sum() обчислює кількість пропущених значень для кожного стовпця. Пропущені значення - це значення, які відсутні в даних для певних рядків чи стовпців:

```
title          0
magnitude      0
date_time      0
cdi            0
mmi            0
alert        367
Earthquake     0
sig            0
net            0
nst            0
dmin           0
gap            0
magType        0
depth          0
latitude       0
longitude      0
location       5
continent    576
country      298
dtype: int64
```

Наступний код проходить по кожному стовпцю в датафреймі df і перевіряє, скільки унікальних значень міститься в кожному стовпці. Якщо кількість унікальних значень менше 100, код виводить назву стовпця та список унікальних значень для подальшого аналізу.

```
for val in df.columns:
    if len(pd.unique(df[val]))<100:
        print('*'*10,val,'*'*10)
        print(pd.unique(df[val]))
```

```
********** magnitude **********
[7.   6.9  7.3  6.6  6.8  6.7  7.6  6.5  7.2  7.5  7.1  8.1  8.2  7.4
 7.7  7.8  8.   7.9  8.3  8.6  9.1  8.8  8.4  8.16]
********** cdi **********
[8 4 3 5 0 1 7 9 2 6]
********** mmi **********
[7 4 3 5 2 6 8 9 1]
********** alert **********
['green' 'yellow' 'orange' 'red' nan]
********** Earthquake **********
[1 0]
********** net **********
['us' 'at' 'pt' 'ak' 'nn' 'ci' 'hv' 'nc' 'official' 'duputel' 'uw']
```

```
********** magType **********
['mww' 'mwb' 'Mi' 'ml' 'mw' 'mwc' 'ms' 'mb' 'md']
********** continent **********
['Oceania' nan 'North America' 'Asia' 'South America' 'Europe' 'Africa']
********** country **********
['Solomon Islands' nan 'Fiji' 'Panama' 'Mexico' 'Taiwan'
 'Papua New Guinea' "People's Republic of China" 'Philippines' 'Brazil'
 'Peru' 'Argentina' 'Indonesia' 'United States of America' 'Antarctica'
 'Vanuatu' 'Haiti' 'Japan' 'Mongolia' 'Greece' 'Chile' 'Russia' 'Turkey'
 'United Kingdom of Great Britain and Northern Ireland (the)' 'Ecuador'
 'South Georgia and the South Sandwich Islands' 'Venezuela' 'Bolivia'
 'Costa Rica' 'Iran' 'Guatemala' 'Botswana' 'New Zealand' 'Italy'
 'Myanmar' 'Afghanistan' 'India' 'Tajikistan' 'Nepal' 'Nicaragua'
 'Pakistan' 'Colombia' 'Canada' 'Tonga' 'Kyrgyzstan' 'Martinique'
 'Mozambique' 'Tanzania' 'Algeria' 'El Salvador']
```

Це може допомогти швидко оглянути і аналізувати невеликі стовпці з обмеженою кількістю унікальних значень. На наступному етапі роботи переходять до очистки даних. За це відповідає функція nullValues(df):

```
def nullValues(df):
    total=df.isnull().sum()
    percent=df.isnull().sum()/df.isnull().count()*100
    null_df=pd.concat([total,percent],axis=1,keys=["Total","Percent"])
    null_df=null_df[null_df["Percent"]>0]
    null_df=null_df.sort_values(by="Percent",ascending=False)
    print(pd.DataFrame(null_df))
    plt.figure(figsize=(16,10))
    sns.barplot(x=null_df.index,y=null_df["Percent"],color="g")
    plt.xticks(rotation=90)
    plt.xlabel("Null_value_Column")
    plt.ylabel("Percent")
nullValues(df)
```

4.2. Визначення стовпців наору даних

CDI - базується на повідомленнях людей, які відчули землетрус, і враховувати фізичні характеристики землетрусу.

MMI (змінена інтенсивність Меркаллі) також базується на звітах осіб, які відчули землетрус, і враховує низку факторів, таких як сила струсу, пошкодження будівель та інфраструктури тощо.

sig: скорочення від magnitude significance, це міра відносної сили землетрусу. Розраховується на основі магнітуди землетрусу, кількості сейсмічних станцій, які повідомляють про подію, і відстань між подією та цими станціями.

net: скорочення від seismic network, це відноситься до мережі сейсмометрів та інших сейсмічних датчиків, що використовуються для виявлення та реєстрації сейсмічних подій. Кожна сейсмічна мережа має свої специфічні характеристики та можливості, які можуть вплинути на точність і повноту виявлення землетрусу та місцезнаходження.

nst: скорочення від number of seismic stations, це кількість сейсмічних станцій, які виявили та зафіксували землетрус. Чим більше сейсмічних станцій виявляють подію, тим точніше її місцезнаходження та інше можна визначити його атрибути.

dmin: скорочення від minimum distance, це найкоротша відстань між землетрусом і найближчою сейсмічною станцією, яка це виявила. Ця відстань може вплинути на точність визначення місця землетрусу та інші атрибути, а також силу сейсмічних хвиль, виявлених станцією.

розрив відноситься до азимутального розриву між сейсмічними станціями, які фіксують землетрус. Азимутальний розрив - кут між двома найбільш віддаленими сейсмічними станціями, які фіксують землетрус. Великий розрив може вказувати на відсутність важливих сейсмічних даних, що може ускладнити точне визначення місце землетрусу, магнітуди та інші характеристики.

magType відноситься до типу обчислення магнітуди, який використовується для визначення розміру землетрусу. Є кілька різних типів розрахунків величини, включаючи величину Ріхтера, величину моменту та поверхню величини хвилі. Кожен тип обчислення магнітуди враховує різні аспекти землетрусу, наприклад як амплітуду сейсмічних хвиль або енергію, яка вивільнена під час землетрусу. Вибір типу величини може впливати на зареєстроване значення магнітуди, а також можуть впливати на оцінку небезпеки землетрусу та планування реагування на надзвичайні ситуації.

Наступний фрагмент коду вибирає певні ознаки та цільову змінну з датафрейму для подальшого аналізу та моделювання:

```
features = ["longitude","latitude", "depth", "cdi", "mmi",
"sig","dmin","nst","gap" ]
target = "alert"
df = df[features + [target]]
```

features - назви стовпців, які були обрані як ознаки для аналізу та моделювання. Ознаки, здебільшого числові, включають longitude, latitude, depth, cdi, mmi, sig, dmin, nst і gap. Ці ознаки використовуватимуться для навчання моделі. target - це змінна, яка була визначена як цільова. df[features + [target]]: цей рядок коду вибирає колонки зі списку ознак features та додає до них цільову змінну target. Отриманий датафрейм буде містити тільки обрані ознаки та цільову змінну для подальшого використання.

4.3. Попередня обробка даних

Цей фрагмент коду виводить кількість унікальних значень для кожного стовпця в датафреймі df. Він допомагає зрозуміти, які значення та їх кількість містяться в кожному стовпці.

```
for val in df.columns:
    print('*'*10,val,'*'*10)
    print(df[val].value_counts())
```

```
********** longitude **********
 159.5960    1
 167.3030    1
 138.5280    1
 158.0530    1
 157.8770    1
            ..
 126.7580    1
 166.8750    1
-71.3815     1
 42.3568     1
-115.2950    1
Name: longitude, Length: 415, dtype: int64
********** latitude **********
-9.7963      1
-14.8595     1
-2.6286      1
-9.3438      1
-9.3293      1
            ..
 2.2580      1
-13.3360     1
-30.0404     1
-43.1219     1
 32.2862     1
```

```
********** depth **********
10.000     54
20.000     17
12.000     10
35.000     10
24.000     10
           ..
540.000     1
461.000     1
550.000     1
576.000     1
9.987       1
Name: depth, Length: 213, dtype: int64
********** cdi **********
0    62
8    59
9    55
5    54
7    51
6    47
4    33
3    28
1    14
2    12
Name: cdi, dtype: int64
********** mmi **********
6    99
7    92
5    68
4    65
8    40
3    34
9    12
2     4
1     1
Name: mmi, dtype: int64
********** sig **********
650     22
670     16
691     13
776     11
651     10
        ..
1145     1
909      1
1600     1
803      1
2048     1
Name: sig, Length: 240, dtype: int64
********** dmin **********
0.00000    44
0.28900     2
2.70500     2
1.77800     2
1.50500     2
           ..
1.19300     1
1.40200     1
2.62900     1
0.04616     1
0.51370     1
Name: dmin, Length: 363, dtype: int64
********** nst **********
0      354
117      2
131      2
445      2
531      1
385      1
626      1
264      1
548      1
770      1
506      1
491      1
640      1
317      1
480      1
697      1
515      1
421      1
562      1
```

```
********** gap **********
18.0     23
16.0     21
22.0     21
17.0     18
12.0     18
         ..
62.0      1
123.0     1
8.0       1
51.0      1
239.0     1
Name: gap, Length: 88, dtype: int64
********** alert **********
green     325
yellow     56
orange     22
red        12
Name: alert, dtype: int64
```

У цьому циклі метод df.columns повертає список назв стовпців в датафрейм. Для кожного стовпця код виводить заголовок стовпця за допомогою методу print('*'*10, val, '*'*10), а потім використовують метод .value_counts() для виведення кількості кожного унікального значення в цьому стовпці.

```
df[target].value_counts().plot(kind='bar', title='Count (target)',
color=['green', 'yellow', 'orange', 'red']);
```

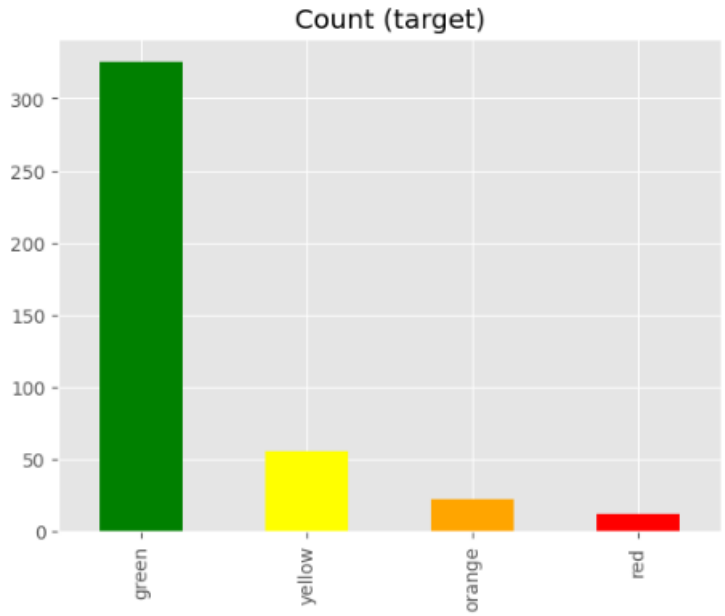

Рис. 4.2. Розподіл кількості значень в цільовій змінній.

Цей фрагмент коду виводить стовпчикову діаграму, яка показує розподіл кількості значень в цільовій змінній alert. Кожен стовпець на діаграмі представляє один з можливих значень цільової змінної, а висота стовпця відображає кількість входжень цього значення в набір даних. Ця діаграма допомагає візуалізувати розподіл різних значень в цільовій

змінній і може бути корисною для зрозуміння балансу класів та змін в розподілі.

```
X = df[features]
y = df[target]
X = X.loc[:,~X.columns.duplicated()]
sm = SMOTE(random_state=42)
X_res, y_res= sm.fit_resample(X, y,)
y_res.value_counts().plot(kind='bar', title='Count (target)',
color=['green', 'orange', 'red', 'yellow']);
```

Цей фрагмент коду стосується підготовки даних для моделювання, використовуючи метод збільшення за допомогою SMOTE та виводить стовпчикову діаграму для показу балансу класів після застосування SMOTE.

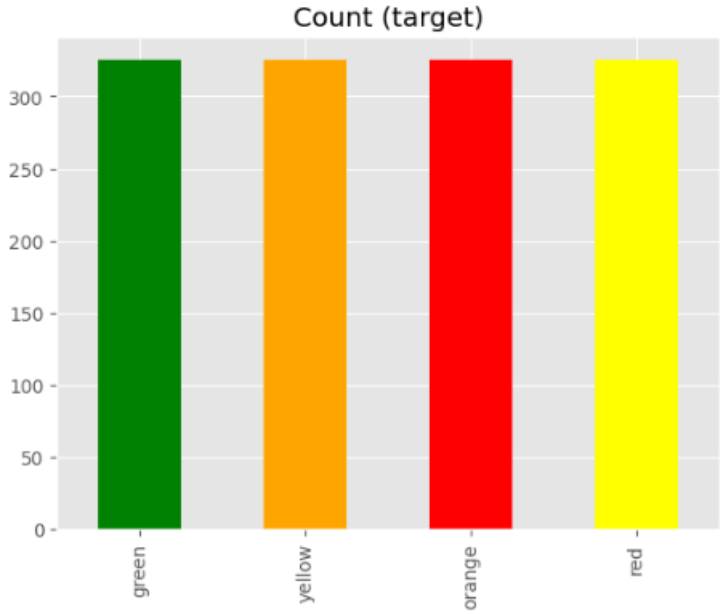

Рис. 4.3. Підготовка даних до моделювання, баланс класів.

4.4. Розбиття та масштабування даних

На наступному етапі проектування інформаційної системи переходять до розбиття та масштабування. Після застосування методу SMOTE для збільшення класів і створення нового набору даних X_res колонки в цьому новому наборі даних залишаться такі ж, як у вихідному наборі даних X. Ця команда допомагає перевірити, чи не втратилися жодні ознаки під час застосування SMOTE:

```
X_res.columns
```

```
Index(['longitude', 'latitude', 'depth', 'cdi', 'mmi', 'sig', 'dmin', 'nst',
       'gap'],
      dtype='object')
```

```
X_res.head(5).T
```

	0	1	2	3	4
longitude	159.5960	100.7380	-178.3460	-172.1290	178.2780
latitude	-9.7963	-4.9559	-20.0508	-19.2918	-25.5948
depth	14.0000	25.0000	579.0000	37.0000	624.4640
cdi	8.0000	4.0000	3.0000	5.0000	0.0000
mmi	7.0000	4.0000	3.0000	5.0000	2.0000
sig	768.0000	735.0000	755.0000	833.0000	670.0000
dmin	0.5090	2.2290	3.1250	1.8650	4.9980
nst	117.0000	99.0000	147.0000	149.0000	131.0000
gap	17.0000	34.0000	18.0000	21.0000	27.0000

Цей фрагмент коду використовує функцію train_test_split для поділу підготовлених даних на навчальний і тестовий набори:

```
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res,
test_size=0.2, random_state=42)
```

Після виконання цієї операції отримають чотири змінні: X_train: Навчальний набір ознак.X_test: Тестовий набір ознак. y_train: Навчальний набір цільових змінних. y_test: Тестовий набір цільових змінних. Ці набори використовуватимуться для навчання та оцінки моделей машинного навчання.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Цей фрагмент коду використовує StandardScaler для масштабування ознак у навчальному та тестовому наборах даних. Масштабування даних допомагає зробити їх більш придатними для моделювання та забезпечити стабільність під час навчання моделі. Ця операція допомагає забезпечити, що ознаки в обох наборах даних мають схожий масштаб та розподіл, що може покращити роботу багатьох алгоритмів машинного навчання.

Наступний код на Python використовує бібліотеку Matplotlib разом із бібліотекою Seaborn для створення графіків гістограм для трьох різних числових змінних у наборі даних df1.

```
fig, axes = plt.subplots(1, 3, figsize=(18, 6), dpi=100)
```

```
sns.histplot(data = df1, x = 'magnitude', ax=axes[0])
axes[0].set_title("Magnitude")
sns.histplot(data = df1, x = 'nst', ax=axes[1])
axes[1].set_title("NST")
sns.histplot(data = df1, x = 'mmi', ax=axes[2])
axes[2].set_title("MMI")
```

В першому рядку створюють фігуру fig та три осі axes, які будуть розташовані в одному рядку та трьох стовпцях. Параметр figsize вказує розмір фігури, а dpi - роздільну здатність (пікселі на дюйм).

Наступна команда використовує бібліотеку Seaborn для створення гістограми для змінної magnitude з набору даних df1 і розміщує її на першій осі. Заголовок графіка встановлюється як Magnitude.

Третя команда створює гістограму для змінної nst і розміщує її на другій осі і встановлює заголовок NST. З допомогою останньої команди створюють гістограму для змінної mmi і розміщують її на третій осі і встановлюють заголовок MMI.

Отже, цей код створює одну фігуру з трьома графіками гістограм, які демонструють розподіл трьох різних числових змінних у наборі даних та надає їм відповідні заголовки. Можна використати цей код для візуалізації розподілу даних та порівняння їх між собою.

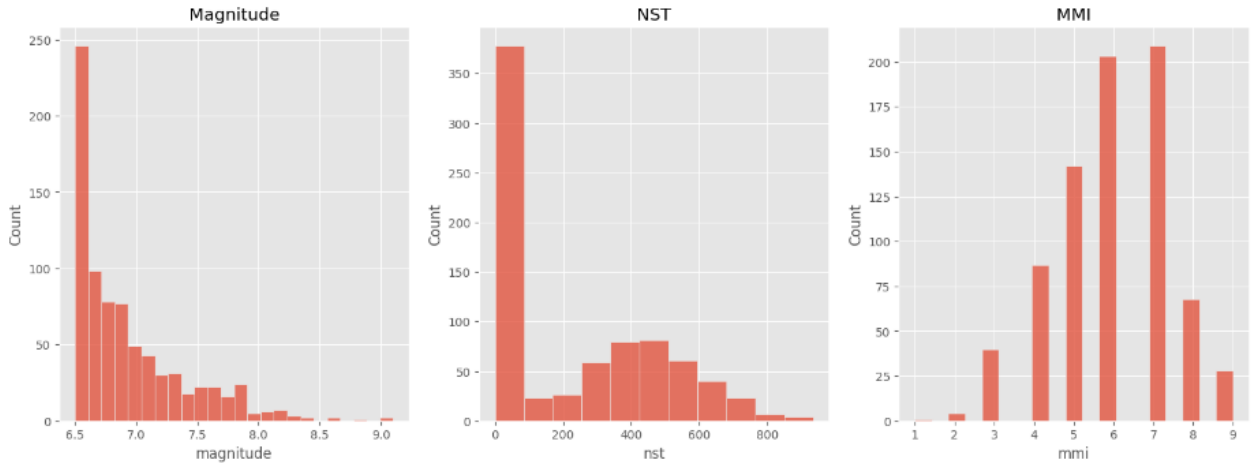

Рис. 4.4. Розподіл трьох різних числових змінних у наборі даних.

```
fig, axes = plt.subplots(1, 2, figsize=(18, 6), dpi=100)
sns.histplot(data = df1, x = 'sig', ax=axes[0])
axes[0].set_title("SIG")
sns.histplot(data = df1, x = 'depth', ax=axes[1])
```

```
axes[1].set_title("Depth")
```

Цей код створює фігуру з двома графіками гістограм, які демонструють розподіл змінних sig та depth у наборі даних df1 та надає їм відповідні заголовки:

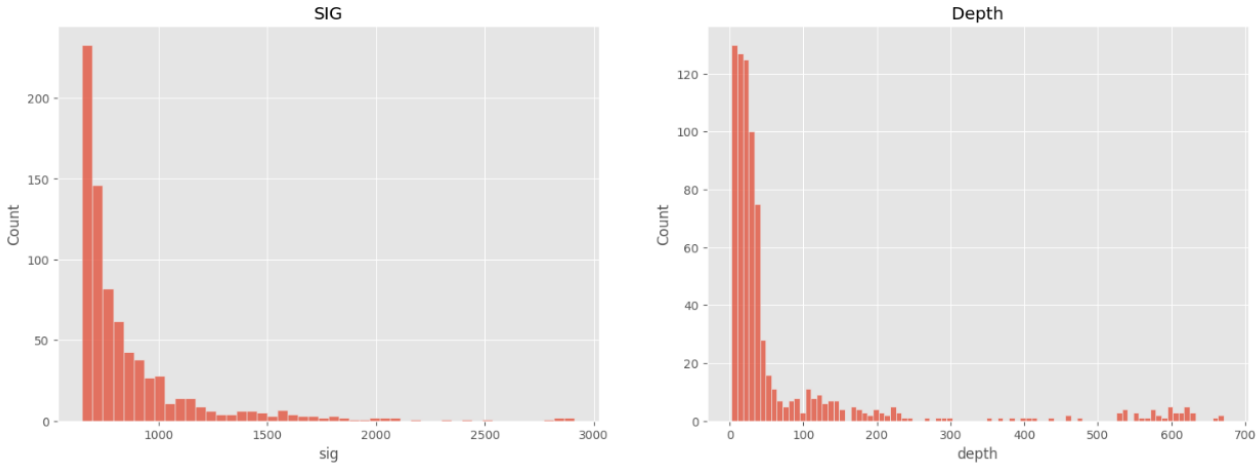

Рис. 4.5. Розподіл змінних sig та depth у наборі даних.

```
plt.plot(X_train[:, 0], X_train[:, 1], 'o', alpha=0.5)
plt.xlabel('magnitude')
plt.ylabel('depth')
plt.title('magnitude vs depth');
plt.show()
```

Використовують бібліотеку Matplotlib для створення графіка розсіювання scatter plot, який відображає взаємозв'язок між двома змінними magnitude і depth з набору даних X_train, де кожна точка на графіку представляє одне спостереження зі значеннями цих двох змінних.

Створюють графік розсіювання. X_train[:, 0] і X_train[:, 1] відповідають значенням змінних magnitude і depth з набору даних. Параметр o вказує, що маркерами на графіку будуть кола. alpha=0.5 встановлює прозорість маркерів. Встановлюють підпис для осі x, яка відображає дані змінної magnitude, а також підпис для осі y, яка відображає дані змінної depth. Відображають заголовок графіку magnitude vs depth. З допомогою методу plt.show() отримують графік розсіювання на екрані.

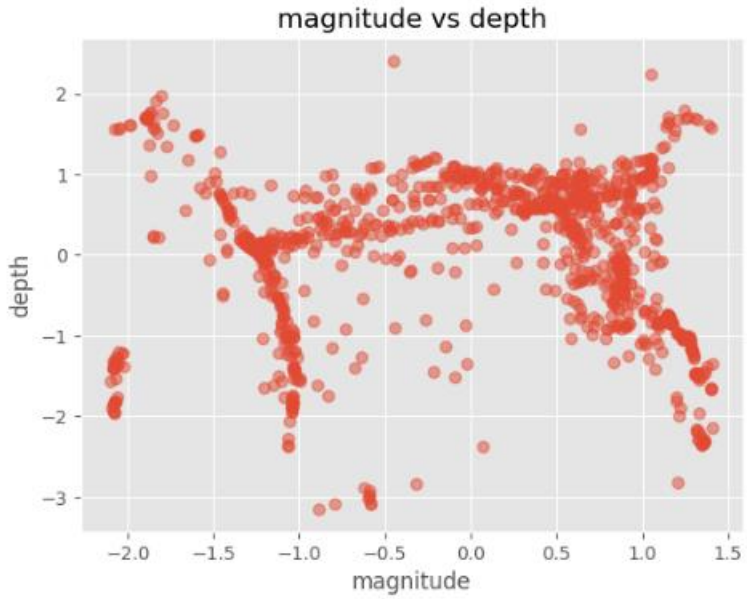

Рис. 4.6. Взаємозв’язок між двома змінними magnitude і depth з набору даних X_train

```
sns.heatmap(df.corr(), annot=True, fmt=".2f")
```

Створюють теплову карту, яка графічно відображає кореляцію між числовими змінними. З неї можна зробити висновок про те, які змінні корелюють між собою та в якому напрямку (позитивна або негативна кореляція). Матриця кореляції показує ступінь взаємозв’язку між парами змінних. Значення кореляції від -1 до 1, де -1 вказує на повну негативну кореляцію, 1 - на повну позитивну кореляцію, а 0 - на відсутність кореляції. З допомогою параметра annot=True вказуть, щоб значення кореляції були виведені на самій тепловій карті. Ці значення будуть розміщені в кожній клітинці теплової карти, що відповідає парі змінних. Параметр fmt=.2f вказує формат для виведення значень кореляції. .2f означає, що значення будуть виведені з двома десятковими знаками після коми.

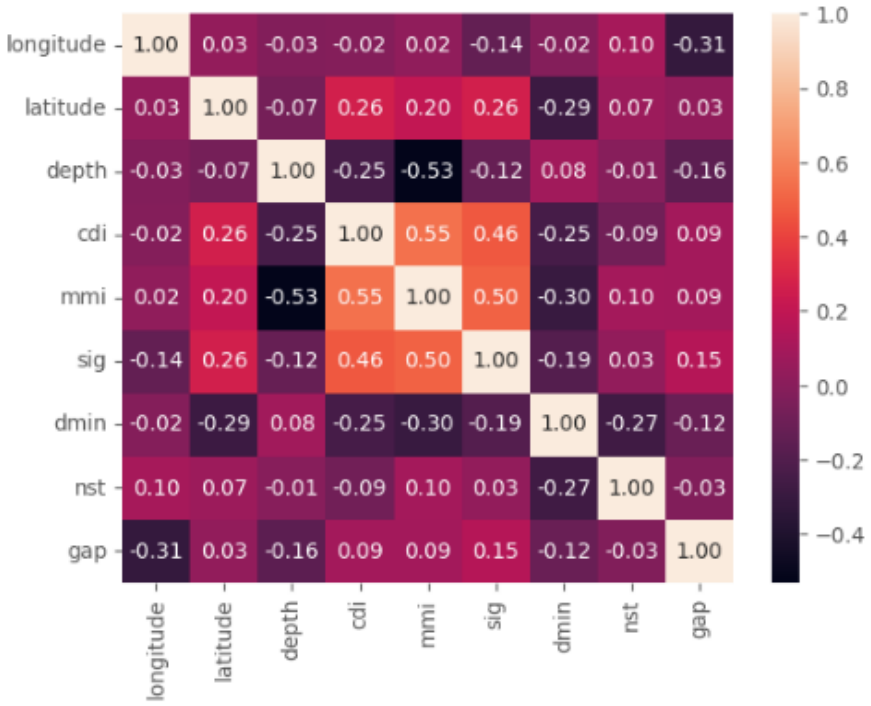

Рис. 4.7. Теплова карта кореляції між числовими змінними.

4.5. Проведення моделювання

```
models = []
```

Створюють порожній список з іменем models. Список models служить для зберігання і організації моделей чи інших об’єктів у коді.

```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
models.append(dt)
```

Створюють модель класифікації рішень Decision Tree Classifier з використанням бібліотеки Scikit-Learn, навчають її на навчальних даних та додають цю навчену модель до списку models.

Створюють об’єкт моделі класифікації рішень Decision Tree Classifier з параметром random_state=42. random_state використовується для задання початкового стану генератора випадкових чисел, щоб зробити результати відтворюваними.

За допомогою команди dt.fit(X_train, y_train) навчають модель dt на навчальних даних X_train та y_train. X_train містить ознаки, які використовуються для навчання моделі, а y_train - відповіді чи мітки класів, які модель намагається передбачити. З допомогою методу

models.append(dt) додають навчену модель dt до списку models. Після цього список models буде містити модель DecisionTreeClassifier, яку навчили на навчальних даних.

knn = KNeighborsClassifier()

knn.fit(X_train, y_train)

models.append(knn)

Тут було створено модель класифікації методом найближчих сусідів K-Nearest Neighbors Classifier за допомогою бібліотеки Scikit-Learn, навчають її на навчальних даних і додають цю навчену модель до списку models. Ця модель буде використовувати значення k=5, тобто вона буде враховувати п’ять найближчих сусідів при класифікації.

Список models буде містити обидві моделі, які навчили: модель класифікації рішень і модель K-Nearest Neighbors.

```
svm = SVC(random_state=42)
svm.fit(X_train, y_train)
models.append(svm)
```

Тут було створено модель опорних векторів Support Vector Machine, навчено її на навчальних даних і додано її до списку models. Параметр random_state=42 використовується для задання початкового стану генератора випадкових чисел. Тепер список models міститиме три моделі: модель класифікації рішень, модель K-Nearest Neighbors і модель SVM.

rf = RandomForestClassifier(random_state=42)

rf.fit(X_train, y_train)

models.append(rf)

Створюється модель випадкового лісу Random Forest Classifier, навчено її на навчальних даних і додано цю модель до списку models. Параметр random_state=42 використовується для задання початкового стану генератора випадкових чисел.

Список models тепер вже буде містити чотири моделі: модель класифікації рішень, модель K-Nearest Neighbors, модель SVM і модель Random Forest.

```
print(models)
```

Якщо виконати дану команду після додавання всіх моделей до списку models, то отримаємо виведення списку з усіма моделями, які було створено і навчено. Буде показано список моделей, вони будуть відображатися як об’єкти в квадратних дужках. Кожен об’єкт представляє відповідну навчену модель.

X_train.shape

Форма shape об’єкта X_train вказує на розмір та розмірність навчальної вибірки (набору даних) для ознак, які використовуються для навчання моделей машинного навчання. X_train - це двовимірний масив (матриця), де кількість рядків відповідає кількості прикладів у навчальній вибірці, а кількість стовпців відповідає кількості ознак. Отримаємо результат:

```
(1040, 9)
```

Це означає, що є 1040 прикладів у навчальній вибірці і кожен приклад має 9 ознак.

Ця інформація є важливою для правильного налаштування та навчання моделей машинного навчання, оскільки вони очікують, що навчальна вибірка матиме вірну форму даних.

4.6. Оцінка моделей

```
Name=[type(model).__name__ for model in models]
accuracy = [accuracy_score(y_test, model.predict(X_test)) for model in
models]
precision = [precision_score(y_test, model.predict(X_test),
average='weighted') for model in models]
recall = [recall_score(y_test, model.predict(X_test), average='weighted')
for model in models]
f1_scores = [f1_score(y_test, model.predict(X_test), average='weighted')
for model in models]
```

Вище представлено, як буде визначено різні метрики ефективності для кожної моделі, яка знаходиться в списку models. Створюється список Name, в якому будуть зберігатися імена моделей. Використовується

генератор списку, щоб витягнути імена моделей за допомогою функції type(), параметр .__name__ витягує ім’я класу моделі. Тобто цей список буде містити імена моделей, такі як DecisionTreeClassifier, KneighborsClassifier та інші.

Також створюється список accuracy, в якому зберігаються значення точності для кожної моделі. Використано функцію accuracy_score з бібліотеки Scikit-Learn для обчислення точності передбачень моделі на тестових даних (X_test та y_test) для кожної моделі.

Створюється третій список precision, в якому зберігаються значення точності (precision) для кожної моделі. Для цього буде використано функцію precision_score з параметром average='weighted' для обчислення точності для кожної моделі.

Створюється список recall, в якому зберігаються значення чутливості (recall) для кожної моделі. За це буде відповідати функція recall_score з параметром average='weighted для обчислення чутливості для кожної моделі.

В списку f1_scores будуть зберігатися значення F1-оцінки для кожної моделі. Використано функцію f1_score з параметром average=weighted для обчислення F1-оцінки для кожної моделі. Буде отримано доступ до цих метрик ефективності для кожної моделі, вони будуть збережені в списках accuracy, precision, recall, f1_scores, які відображають важливі параметри ефективності для набору моделей.

```
Dict =
{'Name':Name,'Accuracy':accuracy,'Precision_score':precision,'Recall_score'
:recall,'F1_score':f1_scores}
model_df = pd.DataFrame(Dict)
model_df
```

Створюють DataFrame (таблицю даних) за допомогою бібліотеки Pandas для зберігання та відображення метрик ефективності (імена моделей, точність, точність, чутливість і F1-оцінка) для різних моделей.

Створюють словник Dict, в якому ключами є назви метрик, а значеннями є відповідні списки Name, accuracy, precision, recall, f1_scores, які містять інформацію про моделі та їх метрики ефективності.

Після цього створюють датафрейм зі словника Dict. Кожен ключ словника буде назвою стовпця у таблиці, а відповідні значення зі списків стануть значеннями цих стовпців. Отримають таблицю зі стовпцями: Name (назви моделей), Accuracy (точність), Precision_score (точність), Recall_score (чутливість), F1_score (F1-оцінка).

З допомогою команди model_df виводять отриманий датафрейм, можна побачити таблицю з метриками ефективності для кожної моделі, що були обчислені раніше. Ця таблиця є корисною для порівняння різних моделей та визначення того, яка з них працює найкраще на наборі даних. Результат виконання:

	Name	Accuracy	Precision_score	Recall_score	F1_score
0	DecisionTreeClassifier	0.942308	0.941710	0.942308	0.941875
1	KNeighborsClassifier	0.923077	0.934314	0.923077	0.923851
2	SVC	0.930769	0.945339	0.930769	0.932321
3	RandomForestClassifier	0.976923	0.977345	0.976923	0.976927

```
import numpy as np
plt.figure(figsize=(20, 10))
company=Name
xpos =np.arange(len(coampany))
```

Цей фрагмент коду використовує бібліотеку Matplotlib для створення графіку, на якому будуть відображені імена моделей на осі x. Імпортують бібліотеку NumPy, яка використовується для роботи з масивами та чисельними обчисленнями. Команда plt.figure(figsize=(20, 10)) створює нову фігуру (графік) з заданим розміром (20x10 одиниць). Параметр figsize вказує ширину та висоту графіка в дюймах. Створюють список company, який містить імена моделей, цей список використовується для підпису осі x на графіку.

Команда xpos = np.arange(len(company)) створює масив xpos, який містить послідовні числа від 0 до (len(company) - 1). Параметр

len(company) визначає кількість моделей у списку company. Це допоможе розташувати імена моделей на осі x безпосередньо під відповідними стовпцями або позначками на графіку.

На графіку будуть відображені імена моделей. Для цього потрібно використати функцію plt.bar(), plt.xticks() та інші функції бібліотеки Matplotlib для налаштування графіка. Таким чином буде створено стовпчикову діаграму, де на осі x будуть імена моделей, а на осі y - точність кожної моделі.

```
plt.bar(xpos+0., accuracy, width=0.2, label="Accuracy")
plt.bar(xpos+0.2, precision, width=0.2, label="precision")
plt.bar(xpos+0.4, recall, width=0.2, label="Recall")
plt.bar(xpos+0.6, f1_scores, width=0.2, label="F1_score")
```

Цей фрагмент коду використовує бібліотеку Matplotlib для створення стовпчикової діаграми, на якій відображені метрики ефективності (точність, точність, чутливість та F1-оцінка) для кожної моделі. Ці метрики представлені на одному графіку з різними кольорами і підписами. Цей код додає підписи осей, заголовок, легенду і виводить графік з відображенням метрик ефективності для кожної моделі.

Команда plt.bar(xpos+0., accuracy, width=0.2, label="Accuracy") створює стовпчики для метрики точності (accuracy). Параметр xpos+0.0 визначає позицію стовпчика на осі x для кожної моделі, accuracy - значення точності, width=0.2 встановлює ширину стовпців, label="Accuracy" додає підпис до цього стовпця.

Команда plt.bar(xpos+0.2, precision, width=0.2, label="precision") створює стовпчики для метрики точності (precision). Команда plt.bar(xpos+0.4, recall, width=0.2, label="Recall") створює стовпчики для метрики чутливості (recall). Команда plt.bar(xpos+0.6, f1_scores, width=0.2, label="F1_score") створює стовпчики для метрики F1-оцінки (F1_score).

```
plt.xticks(xpos, rotation=80)
plt.xlabel("Algorithms")
plt.axhline(max(precision), color='r', linestyle='--')
plt.legend()
plt.show()
```

У цьому фрагменті коду налаштовують графік і додають деякі додаткові елементи до нього. Після виконання цього коду отримають графік зі стовпчиками, де імена моделей розташовані на осі x, а на графіку також буде виділена горизонтальна лінія, що показує максимальне значення точності. Графік має підписи осей, легенду і налаштування для кращого відображення метрик ефективності.

Команда plt.xticks(xpos, rotation=80) встановлює підписи на осі x на позиціях, які були обчислені в масиві xpos. Параметр rotation=80 встановлює кут повороту підписів на осі x.

Команда plt.xlabel("Algorithms") додає підпис до осі x Algorithms.

З допомогою команди plt.axhline(max(precision), color='r', linestyle='--') додають горизонтальну лінію на графіку. Використовують як параметр max(precision) для визначення максимального значення точності і розташовують лінію на відповідному рівні точності. З допомогою параметра color='r' встановлюють червоний колір для лінії, параметр linestyle='--' встановлює пунктирний стиль для лінії. Команда plt.legend() додає легенду до графіку, що показує, яка лінія відповідає якій метриці (точність, точність, чутливість, F1-оцінка). З допомогою команди plt.show()відображають графік з усіма налаштуваннями. Результат виконання даного коду:

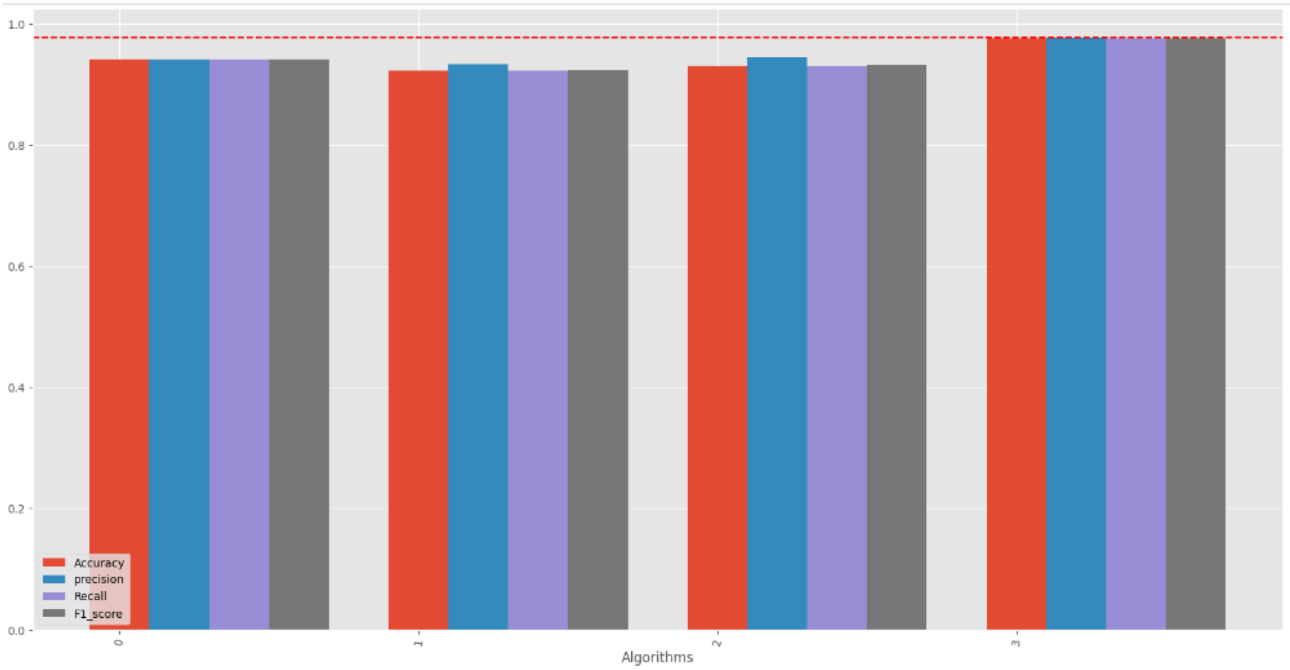

Рис. 4.8. Метрики ефективності accuracy, precision, recall, F1_score для кожної моделі.

```
import numpy as np
import matplotlib.pyplot as plt
algorithm_names = ['DecisionTreeClassifier', 'KNeighborsClassifier', 'SVM',
'RandomForestClassifier']
xpos = np.arange(len(algorithm_names))
```

Створюють список algorithm_names, який містить імена алгоритмів (моделей), потім створюють масив xpos, який містить позиції на осі x для кожного алгоритму.

Створюють список algorithm_names, в якому містяться імена алгоритмів чи моделей, які потрібно відобразити на графіку. Є чотири алгоритми: DecisionTreeClassifier, KNeighborsClassifier, SVM, RandomForestClassifier.

Коли є список імен алгоритмів та масив xpos з позиціями на графіку, можна використовувати ці дані для налаштування графіка або додавання алгоритмів на нього.

```
colors = ['#5cb85c', '#5bc0de', '#f0ad4e', '#d9534f']
fig, ax = plt.subplots(figsize=(12, 6))
ax.bar(xpos, accuracy, width=0.2, label="Accuracy", color=colors[0])
ax.bar(xpos + 0.2, precision, width=0.2, label="Precision",
color=colors[1])
ax.bar(xpos + 0.4, recall, width=0.2, label="Recall", color=colors[2])
```

```
ax.bar(xpos + 0.6, f1_scores, width=0.2, label="F1 Score", color=colors[3])
```

Цей фрагмент коду створює стовпчикову діаграму, на якій відображені метрики ефективності (точність, точність, чутливість та F1-оцінка) для кожного алгоритму. Кожен алгоритм представлений на графіку власним кольором. Після виконання цього коду отримають стовпчикову діаграму, на якій будуть представлені різні метрики ефективності для кожного алгоритму з відповідними кольорами і підписами для легенди. Графік також має встановлений розмір фігури для зручного відображення.

Створюють список colors, в якому вказані чотири кольори для стовпчиків графіку.

Команда fig, ax = plt.subplots(figsize=(12, 6) створює фігуру (графік) та вісь, на якому будуть відображені стовпчики. Встановлено розмір фігури на 12x6 одиниць.

За допомогою команд ax.bar(xpos, accuracy, width=0.2, label="Accuracy", color=colors[0]) створюють стовпці для метрики точності (accuracy) за допомогою функції ax.bar(). xpos - це позиції на осі x, accuracy - значення точності, width=0.2 - ширина стовпців, label="Accuracy" - підпис для легенди, color=colors[0] - встановлює колір стовпця згідно із списком кольорів. Аналогічно створюють стовпці для інших метрик (точність, точність, чутливість та F1-оцінка) і встановлюють відповідні кольори для кожного стовпця.

```
ax.set_xticks(xpos)
ax.set_xticklabels(algorithm_names, rotation=45)
ax.set_xlabel("Algorithms")
ax.set_ylabel("Scores")
ax.set_ylim([0, 1])
ax.axhline(max(precision), color='r', linestyle='--')
ax.legend()
plt.show()
```

Отримують графік з відображенням метрик ефективності для кожного алгоритму, з підписами осей, легендою та іншими налаштуваннями для кращого відображення даних.

Команда ax.set_xticks(xpos) встановлює позиції міток на осі x. Використовують масив xpos, щоб вказати, де будуть розташовані мітки. Команда ax.set_xticklabels(algorithm_names, rotation=45) встановлює тексти міток на осі x. Використовують масив algorithm_names як тексти міток і параметр rotation=45, щоб обернути текст міток на 45 градусів для кращої читабельності. Команда ax.set_xlabel("Algorithms") встановлює підпис Algorithms для осі x, що позначає алгоритми чи моделі. Команда ax.set_ylabel("Scores") встановлює підпис Scores для осі y, що позначає метрики ефективності.

З допомогою команди ax.set_ylim([0, 1]) встановлюють межі для осі y, де [0, 1] вказує, що вісь y буде відображати значення в діапазоні від 0 до 1. Це допомагає відобразити метрики ефективності, оскільки вони часто лежать в цьому діапазоні. Команда ax.axhline(max(precision), color='r', linestyle='--') додає горизонтальну лінію на графіку, яка відображає максимальне значення точності. max(precision) визначає максимальне значення точності, ця лінія буде відображена червоним кольором з пунктирним стилем. Наступні команди ax.legend() додають легенду до графіку, що показує, яка лінія відповідає якій метриці (точність, точність, чутливість, F1-оцінка); plt.show() відображає графік з усіма налаштуваннями. Результат виконання даного фрагмента коду:

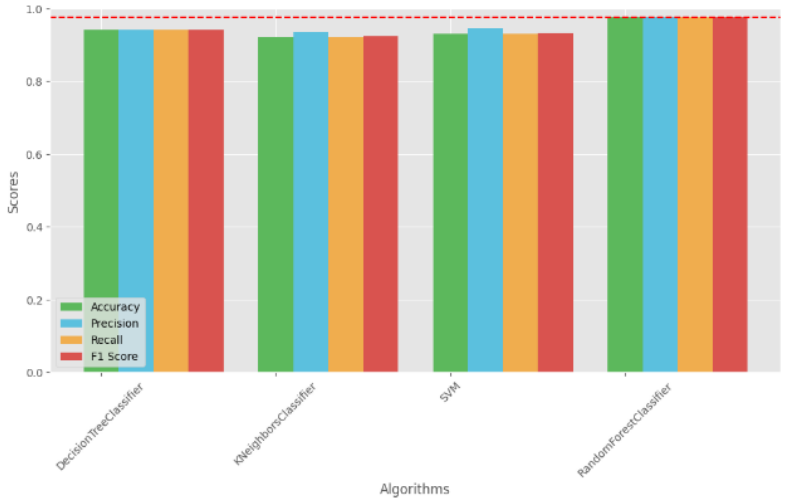

Рис. 4.9. Метрики ефективності accuracy, precision, recall, F1 score для кожної моделі.

```
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
from sklearn.metrics import classification_report,confusion_matrix
from  mlxtend.plotting import plot_confusion_matrix
from matplotlib.colors import ListedColormap
from sklearn import metrics
```

Імпортують різні бібліотеки та функції для роботи з матрицею помилок (confusion matrix), візуалізацією цієї матриці та для обчислення інших метрик класифікації. Імпортують функцію confusion_matrix і клас ConfusionMatrixDisplay з бібліотеки scikit-learn. Ці інструменти використовуються для створення та візуалізації матриці помилок. Функція metrics import classification_report дозволяє генерувати звіт про класифікацію, який включає різні метрики класифікації, такі як точність, чутливість, F1-оцінка. Функція mlxtend.plotting import plot_confusion_matrix імпортується з бібліотеки MLxtend і використовується для візуалізації матриці помилок у вигляді графіка. Клас ListedColormap з бібліотеки Matplotlib може використовуватися для зміни кольорів на графіках, включаючи ті, які використовуються для відображення матриці помилок. Імпорт metrics дозволяє використовувати різні метрики, такі як accuracy_score, precision_score, recall_score, f1_score для оцінки результатів класифікації.

За допомогою цих імпортів можна обчислювати та візуалізувати матрицю помилок, а також отримувати звіти про класифікацію та інші метрики ефективності для класифікаційних моделей.

```
dty_pred = dt.predict(X_test)
cm = metrics.confusion_matrix(y_test, dty_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Цей фрагмент коду використовує бібліотеку Matplotlib та Seaborn для візуалізації матриці confusion matrix для моделі DecisionTreeClassifier. Після виконання цього коду можна побачити теплокарту з матрицею

помилок, яка допоможе оцінити результати класифікації для моделі DecisionTreeClassifier.

Команда dty_pred = dt.predict(X_test) використовує навчену модель DecisionTreeClassifier (dt), щоб зробити передбачення на тестовому наборі даних X_test. Результати передбачень зберігаються у змінній dty_pred.

Команда cm = metrics.confusion_matrix(y_test, dty_pred) обчислює матрицю помилок, яка порівнює реальні значення класів (які збережені у y_test) з передбаченими значеннями класів (dty_pred). Результат матриці помилок зберігається у змінній cm.

Команда sns.heatmap(cm, annot=True, cmap='Blues', fmt='g') створює теплокарту (heatmap) на основі матриці помилок cm за допомогою бібліотеки Seaborn. Параметр annot=True вказує на відображення числових значень в кожній клітинці теплокарти, cmap=Blues` встановлює колірну палітру (синій колір), параметр fmt='g' вказує на відображення значень у форматі змінних чисел.

Наступні дві команди plt.xlabel('Predicted') і plt.ylabel('True') додають підписи до осей x і y на графіку, що позначають Predicted (передбачене) і True (реальне). Команда plt.show() відображає графік з візуалізованою матрицею помилок. Результат виконання:

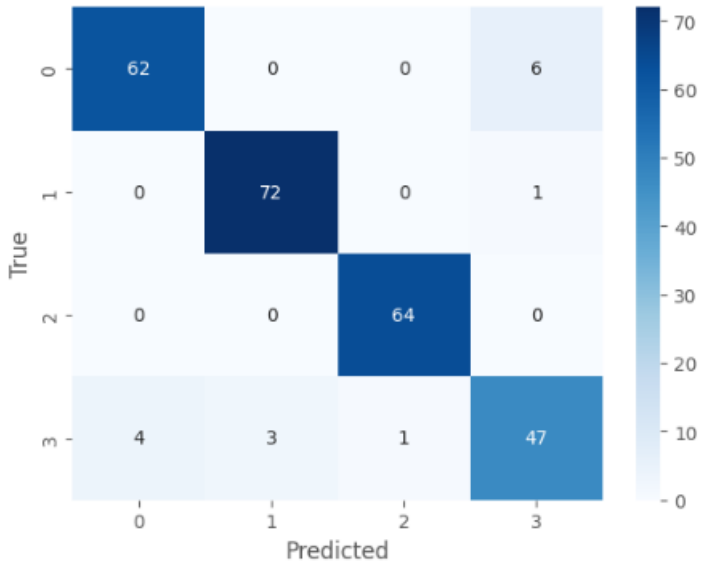

Рис. 4.10. Візуалізації матриці confusion matrix для моделі DecisionTreeClassifier.

```
report = classification_report(y_test, dty_pred)
print(report)
```

Цей фрагмент коду використовує функцію classification_report для обчислення та виведення звіту про класифікацію на основі реальних значень тестового набору y_test та передбачених значень dty_pred для моделі DecisionTreeClassifier. Звіт про класифікацію містить різні метрики, такі як точність, чутливість, F1-оцінка та інші. З допомогою команди print(report) виводять звіт для подальшого аналізу результатів класифікації. Звіт буде містити інформацію про кожен клас та середні метрики для всіх класів, які допоможуть зрозуміти ефективність моделі. Результат виконання:

	precision	recall	f1-score	support
green	0.94	0.91	0.93	68
orange	0.96	0.99	0.97	73
red	0.98	1.00	0.99	64
yellow	0.87	0.85	0.86	55
accuracy			0.94	260
macro avg	0.94	0.94	0.94	260
weighted avg	0.94	0.94	0.94	260

```
kny_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, kny_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Цей фрагмент коду використовує той самий підхід, що і для моделі DecisionTreeClassifier, але тепер для моделі KNeighborsClassifier. Після виконання цього коду отримають теплокарту з матрицею помилок для моделі KNeighborsClassifier, що допоможе оцінити результати класифікації цієї моделі.

kny_pred = knn.predict(X_test) Ви використовуєте навчену модель KNeighborsClassifier (knn), щоб зробити передбачення на тестовому наборі даних X_test. Результати передбачень зберігаються у змінній kny_pred.

cm = confusion_matrix(y_test, kny_pred) Ця команда обчислює матрицю помилок для передбачених результатів (kny_pred) порівняно з реальними результатами (y_test) з використанням функції confusion_matrix. Результат матриці помилок зберігається у змінній cm.

sns.heatmap(cm, annot=True, cmap='Blues', fmt='g') Ця команда створює теплокарту (heatmap) на основі матриці помилок cm з використанням бібліотеки Seaborn. Параметри annot=True вказують на відображення числових значень в кожній клітинці теплокарти, cmap='Blues' встановлюють колірну палітру (синій колір), fmt='g' вказують на відображення значень у форматі змінних чисел.

plt.xlabel('Predicted') і plt.ylabel('True') Ці команди додають підписи до вісей x і y на графіку, що позначають Predicted (передбачене) і True (реальне). plt.show() Ця команда відображає графік з візуалізованою матрицею помилок для моделі KNeighborsClassifier. Результат виконання коду:

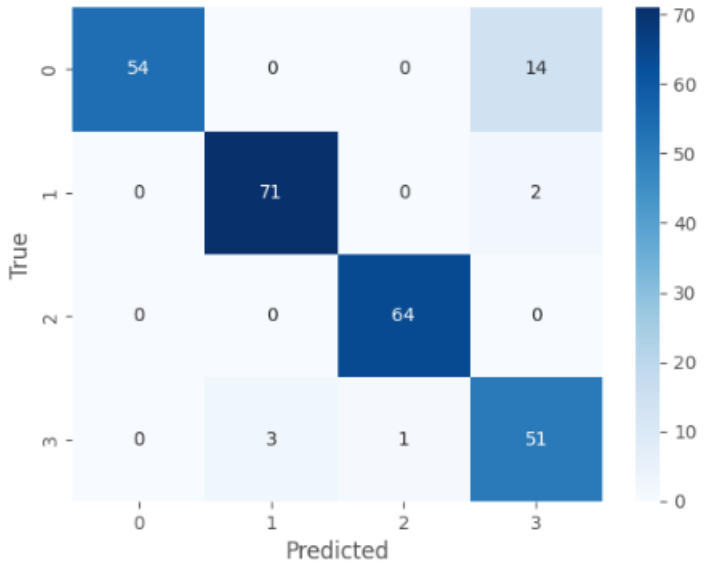

Рис. 4.11. Візуалізації матриці confusion matrix для моделі KNeighborsClassifier.

```
report = classification_report(y_test, kny_pred)
print(report)
```

Тут використовується функція classification_report для обчислення та виведення звіту про класифікацію на основі реальних значень тестового

набору y_test та передбачених значень kny_pred для моделі KNeighborsClassifier. Звіт про класифікацію містить різні метрики, такі як точність, чутливість, F1-оцінка та інші. print(report) виводить звіт для подальшого аналізу результатів класифікації. Звіт буде містити інформацію про кожен клас та середні метрики для всіх класів, які допоможуть зрозуміти ефективність моделі KNeighborsClassifier. Результат виконання:

	precision	recall	f1-score	support
green	1.00	0.79	0.89	68
orange	0.96	0.97	0.97	73
red	0.98	1.00	0.99	64
yellow	0.76	0.93	0.84	55
accuracy			0.92	260
macro avg	0.93	0.92	0.92	260
weighted avg	0.93	0.92	0.92	260

```
svy_pred = svm.predict(X_test)
cm = confusion_matrix(y_test, svy_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Цей фрагмент коду використовується для візуалізації матриці помилок для моделі Support Vector Machine. Після його виконання отримують теплокарту з матрицею помилок для моделі SVM, що допоможе оцінити результати класифікації цієї моделі.

Використовується навчена модель SVM (svm), щоб зробити передбачення на тестовому наборі даних X_test. Результати передбачень зберігаються у змінній svy_pred.

Команда cm = confusion_matrix(y_test, svy_pred) обчислює матрицю помилок для передбачених результатів (svy_pred) порівняно з реальними результатами (y_test) з використанням функції confusion_matrix. Результат матриці помилок зберігається у змінній cm. Команда sns.heatmap(cm, annot=True, cmap='Blues', fmt='g') створює теплокарту на основі матриці помилок cm з використанням бібліотеки Seaborn. Параметри annot=True вказують на відображення числових значень в кожній клітинці теплокарти,

метод cmap='Blues' встановлює колірну палітру (синій колір), fmt='g' вказують на відображення значень у форматі змінних чисел.

plt.xlabel('Predicted') і plt.ylabel('True') Ці команди додають підписи для осей x і y на графіку, що позначають Predicted (передбачене) і True (реальне). Команда plt.show() відображає графік з візуалізованою матрицею помилок для моделі SVM. Результат виконання коду:

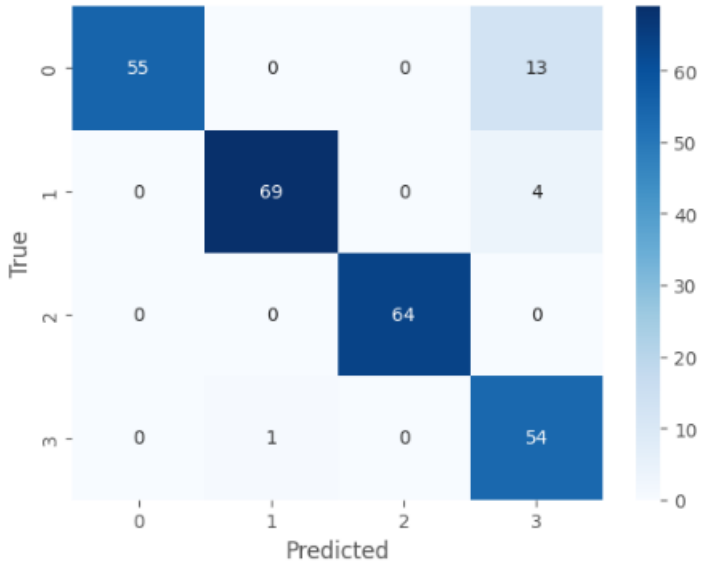

Рис. 4.12. Візуалізації матриці confusion matrix для моделі Support Vector Machine.

```
report = classification_report(y_test, svy_pred)
print(report)
```

Використано функцію classification_report для обчислення та виведення звіту про класифікацію на основі реальних значень тестового набору y_test та передбачених значень svy_pred для моделі SVM (Support Vector Machine). Звіт про класифікацію містить різні метрики, такі як точність, чутливість, F1-оцінка. print(report) виводить звіт для подальшого аналізу результатів класифікації. Звіт буде містити інформацію про кожний клас та середні метрики для всіх класів, які допоможуть зрозуміти ефективність моделі SVM. Результат виконання коду:

	precision	recall	f1-score	support
green	1.00	0.81	0.89	68
orange	0.99	0.95	0.97	73
red	1.00	1.00	1.00	64
yellow	0.76	0.98	0.86	55
accuracy			0.93	260
macro avg	0.94	0.93	0.93	260
weighted avg	0.95	0.93	0.93	260

```
rfy_pred = rf.predict(X_test)
cm = confusion_matrix(y_test, rfy_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Цей фрагмент коду використовується для візуалізації матриці помилок для моделі RandomForestClassifier. Використовується навчена модель RandomForestClassifier (rf), щоб зробити передбачення на тестовому наборі даних X_test. Результати передбачень зберігаються у змінній rfy_pred.

Команда cm = confusion_matrix(y_test, rfy_pred) обчислює матрицю помилок для передбачених результатів (rfy_pred) порівняно з реальними результатами (y_test) з використанням функції confusion_matrix. Результат матриці помилок зберігається у змінній cm.

Команда sns.heatmap(cm, annot=True, cmap='Blues', fmt='g') створює теплокарту (heatmap) на основі матриці помилок cm з використанням бібліотеки Seaborn. Параметри annot=True вказують на відображення числових значень в кожній клітинці теплокарти, cmap='Blues' встановлюють колірну палітру (синій колір), fmt='g' вказують на відображення значень у форматі змінних чисел.

plt.xlabel('Predicted') і plt.ylabel('True') Ці команди додають підписи до осей x та y на графіку, що позначають Predicted (передбачене) і True (реальне). Команда plt.show() відображає графік з візуалізованою матрицею помилок для моделі RandomForestClassifier.

Після виконання коду отримують теплокарту з матрицею помилок для моделі RandomForestClassifier, що допоможе оцінити результати класифікації цієї моделі. Результат виконання коду:

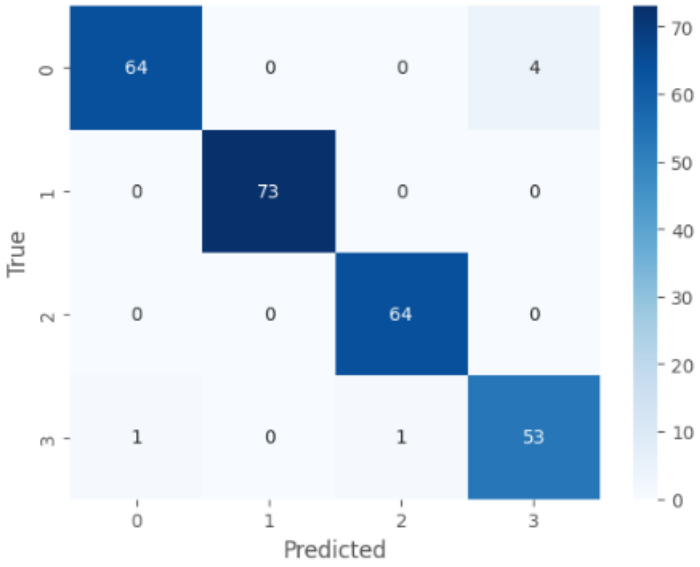

Рис. 4.13. Візуалізації матриці confusion matrix для моделі RandomForestClassifier.

```
report = classification_report(y_test, svy_pred)
print(report)
```

Цей фрагмент коду виглядає так, як його використовували раніше для моделі SVM (Support Vector Machine). Використовують функцію classification_report для обчислення та виведення звіту про класифікацію на основі реальних значень тестового набору y_test та передбачених значень svy_pred для моделі RandomForestClassifier. Звіт про класифікацію містить різні метрики, такі як точність, чутливість, F1-оцінка та інші, він виводиться за допомогою команди print(report). Цей звіт надає інформацію про ефективність моделі RandomForestClassifier на тестовому наборі даних, а саме про якість її класифікації для кожного класу та середні метрики для всіх класів. Результат виконання коду:

```
              precision    recall  f1-score   support

       green       1.00      0.81      0.89        68
      orange       0.99      0.95      0.97        73
         red       1.00      1.00      1.00        64
      yellow       0.76      0.98      0.86        55

    accuracy                           0.93       260
   macro avg       0.94      0.93      0.93       260
weighted avg       0.95      0.93      0.93       260
```

```
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
```

Тут перероблять (перенавчають) модель RandomForestClassifier (rf) знову, використовуючи навчальний набір даних (X_train та y_train). Основна ідея перенавчання моделі полягає в тому, щоб використовувати навчальний набір даних для навчання моделі знову з тими самими параметрами. Це може бути корисно, якщо виявиться, що потрібно покращити модель, або якщо з'явилися нові дані, які потрібно використовувати для навчання. Після виконання цього коду модель RandomForestClassifier буде перенавчена на навчальному наборі даних X_train і y_train, вона буде готова до використання для передбачення на нових даних.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
```

Використовують функцію train_test_split для розділення набору даних на навчальний (X_train, y_train) та тестовий (X_test, y_test) набори даних. Розділення відбувається з використанням параметра test_size=0.2, що означає, що 20 % даних будуть використовуватися для тестування, а 80 % - для навчання. Параметр random_state=42 вказує на використання тієї ж самої початкової генерації випадкових чисел для забезпечення відтворюваності поділу. Після поділу даних створюється модель RandomForestClassifier з параметром random_state=42 для забезпечення відтворюваності навчання моделі. Тепер є два окремих набори даних (X_train, y_train для навчання та X_test, y_test для тестування), і модель RandomForestClassifier, яку можна навчити на навчальних даних і використовувати для передбачення на тестових даних для оцінки її ефективності.

```
train_acc = rf.score(X_train, y_train)
test_acc = rf.score(X_test, y_test)
plt.plot([1, 2], [train_acc, test_acc], marker='o')
plt.xticks([1, 2], ['Train', 'Test'])
plt.ylabel('Accuracy')
plt.title('Random Forest Classifier Accuracy')
```

```
plt.show()
```

Цей фрагмент коду використовує модель RandomForestClassifier, яку було навчено на навчальних даних X_train та y_train, для обчислення точності (accuracy) на навчальному і тестовому наборах даних, і потім візуалізовують ці результати.

Команда train_acc = rf.score(X_train, y_train) використовує навчену модель RandomForestClassifier (rf) для обчислення точності (accuracy) на навчальному наборі даних (X_train і y_train). Результат точності на навчальних даних зберігається у змінній train_acc.

Аналогічно до попереднього кроку, команда test_acc = rf.score(X_test, y_test) обчислює точність на тестовому наборі даних (X_test і y_test) і зберігає результат у змінній test_acc.

Команда plt.plot([1, 2], [train_acc, test_acc], marker='o') створює графік з точністю на навчальних і тестових даних, де 1 і 2 на осі x представляють навчальний і тестовий набори даних відповідно, а точки (marker='o') показують значення точності на цих наборах.

Команда plt.xticks([1, 2], ['Train', 'Test']) задає підписи на осі x, вказуючи, що 1 відповідає навчальному набору даних, 2 - тестовому набору даних. Команди plt.ylabel('Accuracy') і plt.title('Random Forest Classifier Accuracy') додають підписи до осі y (точність) і заголовок графіку відповідно. Команда plt.show() відображає графік з точністю моделі RandomForestClassifier на навчальних і тестових даних. Цей графік допомагає визначити, чи виникає перенавчання або недонавчання в моделі RandomForestClassifier, оцінити її загальну ефективність та порівняти точність на навчальних і тестових даних.

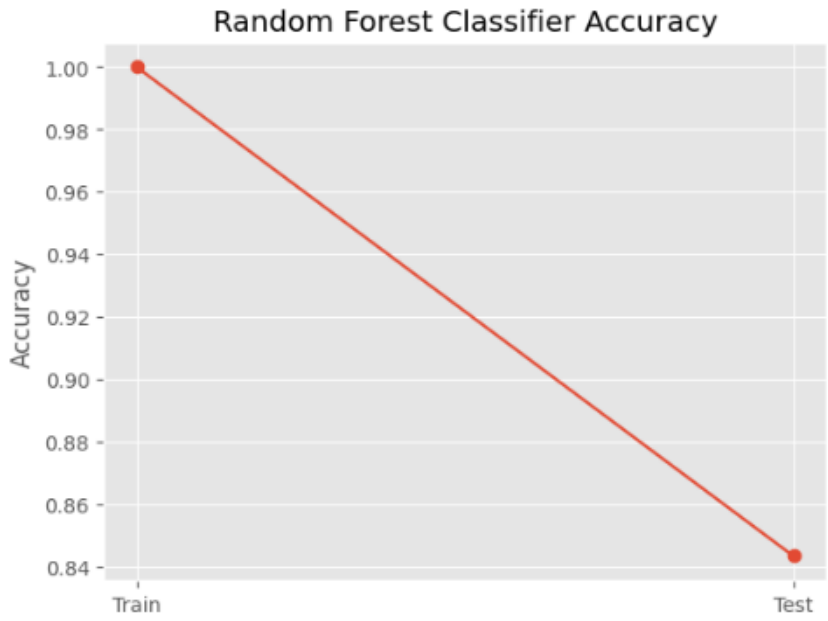

Рис. 4.14. Використання моделі RandomForestClassifier для обчислення точності на навчальному і тестовому наборах даних.

X_res

	longitude	latitude	depth	cdi	mmi	sig	dmin	nst	gap
0	159.596000	-9.796300	14.000000	8	7	768	0.509000	117	17.000000
1	100.738000	-4.955900	25.000000	4	4	735	2.229000	99	34.000000
2	-178.346000	-20.050800	579.000000	3	3	755	3.125000	147	18.000000
3	-172.129000	-19.291800	37.000000	5	5	833	1.865000	149	21.000000
4	178.278000	-25.594800	624.464000	0	2	670	4.998000	131	27.000000
...	...	...	...	...	...	...	...	...	...
1295	-49.945540	-28.300916	26.928511	9	7	1696	0.308319	66	35.999831
1296	153.596611	-4.583648	96.369547	9	7	972	1.490109	0	12.819125
1297	-76.877601	-2.670949	141.767409	7	7	1142	2.018211	0	15.764642
1298	-91.895195	14.317741	55.504201	6	6	956	0.581136	0	41.117683
1299	119.904206	9.031155	14.621107	8	7	720	2.434771	0	34.294980

1300 rows × 9 columns

4.7. Результати роботи інформаційної системи

```
def model_final1(rf):
    longitude = float(input('Enter the longitude: '))
    latitude = float(input('Enter the latitude: '))
    depth = float(input('Enter the depth: '))
    cdi = float(input('Enter the cdi: '))
    mmi = float(input('Enter the mmi: '))
    sig = float(input('Enter the sig: '))
    dmin = float(input('Enter the dmin: '))
    nst = float(input('Enter the nst: '))
```

```
    gap = float(input('Enter the gap: '))
    data = {'longitude': longitude, 'latitude': latitude, 'depth': depth,
'cdi': cdi, 'mmi': mmi, 'sig': sig, 'dmin': dmin, 'nst': nst, 'gap': gap}
    df = pd.DataFrame(data, index=[0])

    val = rf.predict(df)
    if val == 'orange':
        print('High magnitude earthquake')
    elif val == 'green':
        print('Low magnitude earthquake')
    elif val == 'yellow':
        print('Medium magnitude earthquake')
    elif val == 'red':
        print('Very high magnitude earthquake')
```

```
#-178.3460,-20.0508,579,3,3,-13,579,green
```

Функція model_finall використовує навчену модель RandomForestClassifier (rf) для передбачення категорії землетрусу на основі введених ознак (longitude, latitude, depth, cdi, mmi, sig, dmin, nst, gap).

Основні кроки в цій функції такі. Користувач вводить значення деяких ознак землетрусу, такі як довгота (longitude), широта (latitude), глибина (depth) і т. д. Значення цих ознак об'єднуються в словник data. Створюється об'єкт датафрейм df зі словника data, що містить введені користувачем дані. За допомогою навченої моделі rf виконується передбачення на об'єкті df, результат зберігається в змінній val. В залежності від значення val виводиться відповідне повідомлення, що вказує на категорію землетрусу (наприклад, High magnitude earthquake). Ця функція дозволяє ввести характеристики землетрусу та отримати передбачену категорію магнітуди землетрусу з використанням навченої моделі RandomForestClassifier. Важливо мати дані, на яких навчалася модель, які відповідають ознакам, які вводять, для отримання точних передбачень.

```
model_final(rf)
```

Викликається функція model_final1(rf), передавши модель RandomForestClassifier (rf) у якості аргументу. Ця функція використовує цю модель для передбачення категорії землетрусу на основі введених користувачем даних про землетрус. Якщо є конкретні значення для властивостей землетрусу (longitude, latitude, depth, cdi, mmi, sig, dmin, nst, gap), можна ввести їх під час виконання цієї функції, вона повинна повернути передбачену категорію магнітуди землетрусу. Результат виконання:

```
enter the logitude 159.5960
enter the latitude -9.7963
enter depth 14.000
enter cdi 8
enter mmi 7
enter the sig 768
enter sig 0.509
enter nst 117
enter gap 17
['green']
```

```
# the color defines the alerts levels
# green :-  low magnitude earthquakes
# yellow :- medium magnitude earthquakes
# orange :- high magnitude earthquakes
# red :- very high magnitude earthquakes
```

Далі надається пояснення щодо того, які рівні магнітуди землетрусів відповідають кожному з кольорів:

- зелений колір: низька магнітуда землетрусу (low magnitude earthquakes);

- жовтий колір: середня магнітуда землетрусу (medium magnitude earthquakes);

- помаранчевий колір: висока магнітуда землетрусу (high magnitude earthquakes);

- червоний колір: дуже висока магнітуда землетрусу (very high magnitude earthquakes).

Це корисна класифікація рівнів землетрусів за магнітудою, яка може бути використана для оцінки потенційної серйозності землетрусу на основі його магнітуди.

```
df.head(50)
```

	longitude	latitude	depth	cdi	mmi	sig	dmin	nst	gap	alert
0	159.5960	-9.7963	14.000	8	7	768	0.509	117	17.0	green
1	100.7380	-4.9559	25.000	4	4	735	2.229	99	34.0	green
2	-178.3460	-20.0508	579.000	3	3	755	3.125	147	18.0	green
3	-172.1290	-19.2918	37.000	5	5	833	1.865	149	21.0	green
4	178.2780	-25.5948	624.464	0	2	670	4.998	131	27.0	green
5	178.3810	-26.0442	660.000	4	3	755	4.578	142	26.0	green
6	178.3630	-25.9678	630.379	1	3	711	4.678	136	22.0	green
7	-82.3396	7.6712	20.000	7	6	797	1.151	145	37.0	green
8	-102.9130	18.3300	20.000	8	7	1179	2.137	175	92.0	yellow
9	-103.2520	18.3667	26.943	9	8	1799	1.153	271	69.0	yellow
10	121.3070	23.1444	10.000	9	9	887	0.401	215	34.0	yellow
11	121.3480	23.0290	10.000	7	7	756	0.430	178	54.0	green
12	170.2390	-21.2077	137.000	7	5	761	2.977	192	45.0	green
13	146.4710	-6.2237	116.000	8	8	965	3.158	272	12.0	yellow
14	102.2790	29.7263	12.000	9	8	1043	8.454	141	34.0	orange

ВИСНОВКИ ДО РОЗДІЛУ 4

В 4 розділі розроблено інформаційно-аналітичну систему для аналізу передвісників землетрусів. Однією з важливих частин підготовки даних був вибір конкретних ознак для подальшого аналізу та моделювання. Після цього використано бібліотеку Scikit-Learn для розділення даних на навчальний та тестовий набори. Для забезпечення стабільності та найкращої ефективності моделі використано StandardScaler для масштабування ознак. Особливу увагу було приділено балансу класів, було використали метод Smote для збільшення зразків меншого класу, чим вдалося вирівняти розподіл класів у навчальному наборі даних. Це важливий крок для покращення ефективності моделювання та уникнення перекосу у використанні даних. Було використано бібліотеку Seaborn для побудови графіків та гістограм, зокрема, для візуалізації розподілу деяких ознак у наборі даних, що допомагає краще зрозуміти характеристики даних.

Було підготовлено дані для подальшого моделювання та аналізу. Ці кроки стали основою для розробки та оцінки моделей машинного навчання для передбачення землетрусів та пов'язанихз ними сигналів.

На основі набору даних, який містить інформацію про землетруси, включаючи різні ознаки, такі як довгота, широта, глибина, cdi, mmi, sig, dmin, nst, gap і колір. Було використано бібліотеку Seaborn для візуалізації даних, зокрема для побудови гістограм ознак магнітуди, NST і MMI. Для візуалізації графіків також використовувалася бібліотека Matplotlib, створюючи підграфіки і додавши заголовки та підписи до осей. Було проведено навчаєте кількох класифікаторів машинного навчання, таких як DecisionTreeClassifier, KNeighborsClassifier, SVC і RandomForestClassifier, на вхідних даних. Після навчання моделей їх було збереженоу списку models. Обчислювалися різні метрики, такі як точність (accuracy), точність (precision), чутливість (recall) і F1-оцінка для оцінки ефективності кожної моделі. Було створено датафрейм model_df, щоб зберегти ці метрики разом з іменами моделей. Для порівняння результатів було створено графік, на якому відображені значення цих метрик для різних моделей. Ці моделі було використано для передбачення категорій магнітуди землетрусу на основі введених користувачем даних. Категорії магнітуди землетрусу визначаються за кольорами, де зелений вказує на низьку магнітуду, жовтий - середню, помаранчевий - високу і червоний - дуже високу магнітуду.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

5.1. Опис проекту інформаційної системи

Перш ніж почати просувати свій проект як стартап, потрібно зробити кілька кроків. Перший – створити інформаційну карту проекту. На цій карті показано параметри проекту та орієнтовний бюджет, який потрібно виділити на нього. Сюди входять зарплати розробника, які розраховані на певні цикли розробки, та оренда обладнання. Інформаційна карта наведена в табл. 5.1.

Табл. 5.1. Карта інформаційної системи

Назва номінації	Інформаційно-аналітична система
Назва проекту	Розроблення інформаційної системи аналізу передвісників землетрусів
Назва ВНЗ, факультету, спеціальності	НЛТУ, кафедра комп'ютерних наук, 122 «Комп'ютерні науки»
Прізвище, ім'я, по-батькові	Вороновський Данило Юрійович
Цілі і задачі проекту	Мета проекту – розробити інформаційну систему для аналізу передвісників землетрусів. Задачі проекту: • провести огляд літератури по даній тематиці; • проаналізувати існуючі системи прогнозування землетрусів; • розробити математичну модель системи та алгоритм функціонування інформаційної системи; • реалізувати програмну частину

	інформаційної системи; • провести просторовий аналіз даних та кореляцію отриманих результатів.
Терміни виконання проекту	12 місяців
Бюджет проекту	100 000 грн.

5.2. Стратегія проекту

У загальному випадку успішний стартап зі штучного інтелекту або машинного навчання концентрує зусилля на одному або кількох завданнях зі списку:

- зменшення необхідного обсягу людської праці або звільнення від нього в галузях, які раніше вважалися складними для автоматизації;
- використання «вільного місця», яке утворилося завдяки появі нових можливостей (мова про нові продукти або послуги, раніше надто дорогі або неможливі);
- збільшення цінності традиційних додатків завдяки впровадженню техніки машинного навчання в них.

Нарешті, перед розробкою IT-платформи варто подумати про складність виходу на ринок. Поведінка покупців у різних галузях відрізняється, потрібно враховувати і наявність різних каналів просування. Все це ускладнює роботу з просування комплексного продукту.

Якщо модель машинного навчання можна застосувати в різних індустріях, перед вибором спеціалізації варто врахувати наступні змінні.

- вартість впровадження;
- додаткова цінність.

5.3. Розробка програми стартап проекту

Табл. 5.2. Основні переваги та концепції інформаційної системи

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	аналіз сейсмологічних передвісників	створення власних патернів для аналізу зовнішніх факторів	гнучкість та свобода для кінцевого споживача
2	визначення ймовірності виникнення землетрусу	можливість підключення датчиків	можливість використання даних з вебсайтів та інших джерел
3	привабливий інтерфейс	інтерфейс системи, яка дозволяє проводити аналіз ймовірності виникнення землетрусу в даній точці місцевості	визначення ймовірності виникнення землетрусу

Табл. 5.3. Опис рівнів інформаційної системи

рівні товару	**сутність та її складові**
1. Програмний продукт за задумом	інформаційний аналіз можливості виникнення землетрусу
2. Програмний продукт, який має бути реально виконаний	інформаційно-аналітична система для вивявлення та аналізу сейсмологічної небезпеки
3. Підкріплення	служба для підтримки системи прийняття рішень за допомогою цієї інформаційно-аналітичної системи

Табл. 5.4. Визначення меж для встановлення ціни на інформаційну систему

№ п/п	рівень цін на товари замінники	рівень цін на товари-аналоги	рівень доходів цільової групи споживачів	верхня та нижня межі встановлення ціни на товар/послугу
1	наперед не задано	наперед не задано	800$+	800/400 $

ВИСНОВКИ ДО РОЗДІЛУ 5

В даній роботі розроблено та реалізовано інформаційну систему для аналізу передвізників землетрусів. Ця інформаційна система може бути реалізована на практиці. Її перевагою є те, що альтернативних програмних продуктів немає, а якщо є, то дуже мало. При використанні реклами та оголошень для просування програмного продукту можна досягти успіху та отримати пристойний дохід.

ВИСНОВКИ

В роботі була проведена робота над розробкою математичного та програмного забезпечення сейсмологічної аналітичної системи. В рамках проведеного дослідження:

1. Визначено сейсмологічне поняття інформаційно-аналітичної системи – комплекс апаратних, програмних засобів, інформаційних ресурсів, методик, які використовуються для забезпечення автоматизації аналітичних робіт для вирішення завдань сфери сейсмології. Проведений аналіз аналітичного програмного забезпечення.

2. Проведений аналіз задач і огляд робіт, які присвячені аналізу даних у сфері сейсмології, в ході якого виділено три основні класи завдань:

- задачі сейсмологічної діагностики;
- задачі класифікації та кластеризації;
- задачі передбачення (прогнозування землятрусів).

3. Визначена задача для проведення дослідження, на прикладі рішення якої розроблено проект математичного та програмного забезпечення сейсмологічної аналітичної системи.

4. Проведено дослідження існуючих методів інтелектуального аналізу даних для розробки програмного забезпечення. Проведено аналіз по використанню механізмів машинного навчання в сейсмології та описані приклади їх використання.

5. Розроблено програмне забезпечення сейсмологічної аналітичної системи: система розділена на дві основні частини:

- підсистема вводу, зберігання та управління даними;
- підсистема аналізу даних.

Підсистема аналізу даних реалізується з використанням різних бібліотек для вирішення конкретних завдань у контексті системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Назаревич А.В. Геофізичні провісники деяких відчутних закарпатських землетрусів як відображення процесів формування вогнищевих зон / – Зб. наук. праць “Теоретичні та прикладні аспекти геоінформатики”. – Карпатське відділення Інституту геофізики ім. С.І. Субботіна. – НАН України. – Львів, 2010.

2. Ситник В.Ф., Краснюк М.Т. Інтелектуальний аналіз даних (дейтамайнінг) / – Київ: КНЕУ, 2007. – 376 с.

3. Копей В.Б. Мова програмування Python для інженерів і науковців. Навчальний посібник / – Івано-Франківськ : ІФНТУНГ, 2019. – 272 с.

4. Лубко Д.В., Шаров С.В. Методи та системи штучного інтелекту: навчальний посібник / – Мелітополь: ФОП Однорог Т.В., 2019. – 264 с.

5. Майданська Софія. / – Землетрус.: Родовід, 1998. – 190 с.

6. Петрик М.П. Геофізична екологія. Навчальний посібник для студентів вищих навчальних закладів. / – Луцьк: Видавництво «Волинська обласна друкарня», 2005. – 408 с.

7. Боголюбов В. М., Клименко М. О., Мокін В. Б., Сафранов Т. А., Горова А. І., Прилипко В. А., Адаменко О. М., Полєтаєва Л. М., Картавцев О. М. Моніторинг довкілля. Підручник. / – Вінниця : ВНТУ, 2010. – 232 с.

8. Дегтерева Л.І., Булгакова О.В. Моніторинг довкілля і охорона навколишнього середовища / – Конспект лекцій. – Х.: ХНАМГ, 2011. – 46 с.

9. Максимчук В.І. Інформативність геомагнітного моніторингу в Закарпатській сейсмоактивній зоні / – Геодинаміка. – 2012. – №1 (12). – с. 136-144.

10. Цапка В.Г. Безпека життєдіяльності. Навчальний посібник / – К.: Знання, 2004. – 397 с.

ДОДАТКИ

ДОДАТОК А

1.ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.metrics import f1_score,accuracy_score, recall_score,
precision_score

df = pd.read_csv('/content/earthquake_data.csv')
df.head().T
df1=df.copy()

alerts = df["alert"].unique()
df.head()

df.columns

df.info()


df.isna().sum()
```

```
for val in df.columns:
    if len(pd.unique(df[val]))<100:
        print('*'*10,val,'*'*10)
        print(pd.unique(df[val]))

def nullValues(df):
    total=df.isnull().sum()
    percent=df.isnull().sum()/df.isnull().count()*100
    null_df=pd.concat([total,percent],axis=1,keys=["Total","Percent"])
    null_df=null_df[null_df["Percent"]>0]
    null_df=null_df.sort_values(by="Percent",ascending=False)
    print(pd.DataFrame(null_df))
    plt.figure(figsize=(16,10))
    sns.barplot(x=null_df.index,y=null_df["Percent"],color="g")
    plt.xticks(rotation=90)
    plt.xlabel("Null_value_Column")
    plt.ylabel("Percent")

nullValues(df)

df.columns

features = ["longitude","latitude", "depth", "cdi", "mmi",
"sig","dmin","nst","gap" ]
target = "alert"
df = df[features + [target]]

df.isna().sum()

df.dropna(inplace=True)
df.info()

for val in df.columns:
    print('*'*10,val,'*'*10)
    print(df[val].value_counts())

df[target].value_counts().plot(kind='bar', title='Count (target)',
color=['green', 'yellow', 'orange', 'red']);

X = df[features]
y = df[target]
X = X.loc[:,~X.columns.duplicated()]
```

```
sm = SMOTE(random_state=42)
X_res, y_res= sm.fit_resample(X, y,)
y_res.value_counts().plot(kind='bar', title='Count (target)',
color=['green', 'orange', 'red', 'yellow']);

X_res.columns
X_res.head(5).T

X_train, X_test, y_train, y_test = train_test_split(X_res, y_res,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

fig, axes = plt.subplots(1, 3, figsize=(18, 6), dpi=100)
sns.histplot(data = df1, x = 'magnitude', ax=axes[0])
axes[0].set_title("Magnitude")
sns.histplot(data = df1, x = 'nst', ax=axes[1])
axes[1].set_title("NST")
sns.histplot(data = df1, x = 'mmi', ax=axes[2])
axes[2].set_title("MMI")

fig, axes = plt.subplots(1, 2, figsize=(18, 6), dpi=100)
sns.histplot(data = df1, x = 'sig', ax=axes[0])
axes[0].set_title("SIG")
sns.histplot(data = df1, x = 'depth', ax=axes[1])
axes[1].set_title("Depth")

plt.plot(X_train[:, 0], X_train[:, 1], 'o', alpha=0.5)
plt.xlabel('magnitude')
plt.ylabel('depth')
plt.title('magnitude vs depth');
plt.show()

sns.heatmap(df.corr(), annot=True, fmt=".2f")

models = []

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
models.append(dt)
```

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
models.append(knn)

svm = SVC(random_state=42)
svm.fit(X_train, y_train)
models.append(svm)

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
models.append(rf)

print(models)

X_train.shape

Name=[type(model).__name__ for model in models]
accuracy = [accuracy_score(y_test, model.predict(X_test)) for model in
models]
precision = [precision_score(y_test, model.predict(X_test),
average='weighted') for model in models]
recall = [recall_score(y_test, model.predict(X_test), average='weighted')
for model in models]
f1_scores = [f1_score(y_test, model.predict(X_test), average='weighted')
for model in models]

Dict =
{'Name':Name,'Accuracy':accuracy,'Precision_score':precision,'Recall_score'
:recall,'F1_score':f1_scores}
model_df = pd.DataFrame(Dict)
model_df

import numpy as np
plt.figure(figsize=(20, 10))  # specify figure size
coampany=Name
xpos =np.arange(len(coampany))

plt.bar(xpos+0., accuracy, width=0.2, label="Accuracy")
plt.bar(xpos+0.2, precision, width=0.2, label="precision")
plt.bar(xpos+0.4, recall, width=0.2, label="Recall")
plt.bar(xpos+0.6, f1_scores, width=0.2, label="F1_score")

plt.xticks(xpos, rotation=80)
```

```
plt.xlabel("Algorithms")
plt.axhline(max(precision), color='r', linestyle='--')
plt.legend()
plt.show()

import numpy as np
import matplotlib.pyplot as plt
algorithm_names = ['DecisionTreeClassifier', 'KNeighborsClassifier', 'SVM',
'RandomForestClassifier']
xpos = np.arange(len(algorithm_names))
colors = ['#5cb85c', '#5bc0de', '#f0ad4e', '#d9534f']  # specify colors for
each bar
fig, ax = plt.subplots(figsize=(12, 6))

ax.bar(xpos, accuracy, width=0.2, label="Accuracy", color=colors[0])
ax.bar(xpos + 0.2, precision, width=0.2, label="Precision",
color=colors[1])
ax.bar(xpos + 0.4, recall, width=0.2, label="Recall", color=colors[2])
ax.bar(xpos + 0.6, f1_scores, width=0.2, label="F1 Score", color=colors[3])

ax.set_xticks(xpos)
ax.set_xticklabels(algorithm_names, rotation=45)
ax.set_xlabel("Algorithms")
ax.set_ylabel("Scores")
ax.set_ylim([0, 1])
ax.axhline(max(precision), color='r', linestyle='--')
ax.legend()

plt.show()
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
from sklearn.metrics import classification_report,confusion_matrix
from  mlxtend.plotting import plot_confusion_matrix
from matplotlib.colors import ListedColormap
from sklearn import metrics

dty_pred = dt.predict(X_test)
cm = metrics.confusion_matrix(y_test, dty_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

report = classification_report(y_test, dty_pred)
```

```
print(report)

kny_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, kny_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

report = classification_report(y_test, kny_pred)
print(report)

svy_pred = svm.predict(X_test)
cm = confusion_matrix(y_test, svy_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

report = classification_report(y_test, svy_pred)
print(report)

rfy_pred = rf.predict(X_test)
cm = confusion_matrix(y_test, rfy_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')

plt.show()

report = classification_report(y_test, svy_pred)
print(report)

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

train_acc = rf.score(X_train, y_train)
test_acc = rf.score(X_test, y_test)
```

```
plt.plot([1, 2], [train_acc, test_acc], marker='o')
plt.xticks([1, 2], ['Train', 'Test'])
plt.ylabel('Accuracy')
plt.title('Random Forest Classifier Accuracy')
plt.show()

X_res

def model_final1(rf):
    longitude = float(input('Enter the longitude: '))
    latitude = float(input('Enter the latitude: '))
    depth = float(input('Enter the depth: '))
    cdi = float(input('Enter the cdi: '))
    mmi = float(input('Enter the mmi: '))
    sig = float(input('Enter the sig: '))
    dmin = float(input('Enter the dmin: '))
    nst = float(input('Enter the nst: '))
    gap = float(input('Enter the gap: '))

    data = {'longitude': longitude, 'latitude': latitude, 'depth': depth,
'cdi': cdi, 'mmi': mmi, 'sig': sig,
            'dmin': dmin, 'nst': nst, 'gap': gap}

    df = pd.DataFrame(data, index=[0])

    val = rf.predict(df)

    if val == 'orange':
        print('High magnitude earthquake')
    elif val == 'green':
        print('Low magnitude earthquake')
    elif val == 'yellow':
        print('Medium magnitude earthquake')
    elif val == 'red':
        print('Very high magnitude earthquake')

model_final(rf)
```