Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
(повна назва інституту/факультету)

Кафедра телекомунікацій
(повна назва кафедри)

«На правах рукопису»
УДК _____

До захисту допущено
В.о. завідувача кафедри

_____ Явіся В.С.___
(підпис) (ініціали, прізвище)

“___”_____2019_р.

Магістерська дисертація

на здобуття освітнього ступеня «магістр»

Спеціальність 172 Телекомунікації та радіотехніка,
(код і назва)

За освітньо-професійною програмою Інженерія та програмування інфокомунікацій.

на тему: Дослідження використання методів машинного навчання в мережах інтернет речей.

Виконав (-ла): студент (-ка) 2 курсу, групи ТЗ-81мп
(шифр групи)

Трофименко Віталій Васильович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник д.т.н., професор Кравчук С.О. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант 1, 2, 3 к.т.н., доцент Міночкін Д.А. _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
(повна назва)

Кафедра телекомунікацій
(повна назва)

Спеціальність 172 Телекомунікації та радіотехніка
(код і назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою Інженерія та програмування інфокомунікацій.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____Явіся В.С.
(підпис) (ініціали, прізвище)
«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Трофименку Віталію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Дослідження використання методів машинного навчання в мережах інтернет речей.

науковий керівник дисертації Кравчук Сергій Олександрович, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_07_» «_11_» 2019р. № _3840-с_

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження мережі Інтернет речей

4. Предмет дослідження система управління на базі машинного навчання

5. Перелік завдань, які потрібно розробити
   - Привести загальний огляд IoT;
   - Проаналізувати основні методи та алгоритми машинного навчання;
   - Розробити модель прогнозування погодних показників на основі даних, отриманих з мереж IoT;

6. Орієнтовний перелік ілюстративного матеріалу

1) Тема, мета та завдання магістерської дисертації
2) Загальні відомості про архітектуру IoT;
3) Аналіз даних та машинне навчання у хмарних платформах;
4) Огляд основних методів та алгоритмів машинного навчання:
5) Проведення розробки моделі:
6) Результати розробки;
7) Висновки;

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Доцент Міночкін Д.А.	26.12.2019	27.03.2019
2	Доцент Міночкін Д.А.	27.03.2019	20.09.2019
3	Доцент Міночкін Д.А.	20.09.2019	20.11.2019

9. Дата видачі завдання 20.09.2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Розробка, оформлення, узгодження та затвердження технічного завдання на дипломну роботу	20.09.2018	Виконано
2	Опрацювання літературних джерел з теми досліджень	26.12.2018	Виконано
3	Аналіз вимог завдання, вибір методів і засобів розв’язання поставленої задачі	18.01.2019	Виконано
4	Дослідження відомостей про мережі IoT та їх архітектура	27.03.2019	Виконано
5	Дослідження даних та машинне навчання у хмарних платформах	18.06.2019	Виконано
6	Аналіз основних методів та алгоритмів машинного навчання	20.09.2019	Виконано
7	Отримання вихідних даних для проведення розробки моделі	11.11.2019	Виконано
8	Розробка моделі та аналіз результатів	20.11.2019	Виконано
9	Оформлення пояснювальної записки	07.12.2019	Виконано

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Робота містить 93 сторінки, 35 рисунків, 8 таблиць. Було використано 6 джерел інформації.

Метою роботи є дослідження можливості використання методів машинного навчання для створення адаптивної системи управління в мережах Інтернет речей.

В роботі коротко розглянуті мережі Інтернет речей та їх архітектура. Основну частину роботи займає дослідження основних методів та алгоритмів машинного навчання. В результаті роботи було проведено розробку моделі з використанням машинного навчання, а саме розробка моделі прогнозування погодних показників на базі отриманих даних з мереж IoT.

Ключові слова: мережі інтернет речей, машинне навчання, Python, TensorFlow.

ABSTRACT

The work contains 93 pages, 35 illustrations, 8 tables, 6 sources of information were used.

The purpose of this work is to investigate the possibility of using machine learning methods to create an adaptive control system in the Internet of Things.

The work briefly discusses the Internet of Things and their architecture. The main part of the work is the study of basic methods and algorithms of machine learning. As a result, a model was developed using machine learning, namely, the development of a weather forecasting model based on data obtained from IoT networks.

Keywords: internet of things network, machine learning, Python, TensorFlow.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API (application programming interfaces) – прикладний програмний інтерфейс,

HTTP (hyper text transfer protocol) – протокол передачі гіпертексту,

IoT (Internet of Things) – Інтернет речей,

IP (Internet Protocol) – протокол мережевого рівня стеку TCP / IP,

МН (machine learning) – машинне навчання,

SaaS (software as a service) – програмне забезпечення як послуга,

MQTT (message queuing telemetry transport) – спрощений мережевий протокол поверх TCP / IP,

TCP (Transmission Control Protocol) – протокол керування передачею

ВСТУП

Актуальність роботи. Інтернет речей повільно проникає в кожен аспект нашого життя. Передбачається, що до 2020 року загальна кількість підключених до інтернету пристроїв буде складати від 25 до 50 мільярдів. По мірі того як дані цифри ростуть, а технології стають більш зрілими, об'єм публікованих даних буде збільшуватись. На додаток до збільшення об'єму, IoT генерує Big Data, що характеризуються їх швидкістю з точки зору їх залежності від часу та місцерозташування, з множиною різноманітних способів та якістю даних. Без належної стратегії керування даними та їх аналізу, дані технології просто створюють більше шуму і заповнюють більше серверів, фактично не використовуючи їх потенціал. Інтелектуальна обробка та аналіз цих даних є ключем до розробки інтелектуальних додатків IoT. Використання методів машинного навчання дає можливість переходу від серверу, де дані збираються та зберігаються, до місця, де вони інтерпретуються і розуміються.

Метою роботи є дослідження можливості використання методів машинного навчання для створення адаптивної системи управління в мережах Інтернет речей.

Для досягнення поставленої мети було визначено наступні **основні завдання:**

1) Привести загальний огляд IoT;
2) Аналіз основних методів та алгоритмів машинного навчання;
3) Розробка моделі прогнозування погодних показників на основі даних, отриманих з мереж IoT;

Об'єкт дослідження – мережі Інтернет речей;

Предмет дослідження – система управління на базі машинного навчання;

1. ЗАГАЛЬНИЙ ОГЛЯД IoT

1.1 Архітектура та ключові модулі IoT

Архітектура інтернету речей починається з найпростіших датчиків, розташованих у найвіддаленіших куточках світу, що перетворюють аналоговий фізичний вплив в цифрові сигнали (мова інтернету). Потім дані роблять складну подорож по провідним і бездротовим сигналам, різним протоколам, в результаті чого потрапляють в інтернет. Звідти пакети даних по різних каналах передаються в хмару або в великий центр обробки даних. Сильна сторона інтернету речей полягає в тому, що він являє собою не просто одиничний сигнал від одного датчика, а суму всіх сигналів з сотень, тисяч, можливо, мільйонів датчиків, точок і пристроїв.

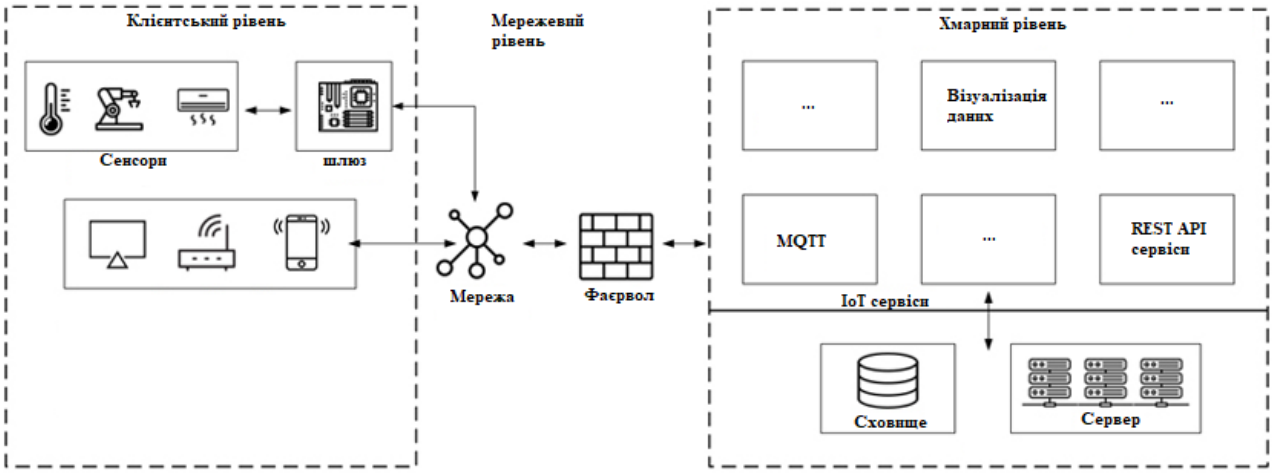

Рис.1.1. Базова архітектура IoT

Ці сфери активно користуються тою безліччю пристроїв, програмного забезпечення і сервісів, які пропонує інтернет речей. Практично кожна велика технологічна компанія вкладає або вкладала гроші в інтернет речей. Вже встигли сформуватися нові ринки і технології (а деякі з них встигли зазнати фіаско або були перепродані). Практично у кожному сегменті інформаційних технологій є своя роль для інтернет речей:

- Датчики: вбудовані системи, операційні системи реального часу, джерела безперебійного живлення, мікроелектромеханічні системи;
- Системи зв'язку між датчиками: зона охоплення бездротових персональних мереж становить від 0 см до 100 м. Для обміну даними між датчиками застосовуються низькошвидкісні малопотужні інформаційні канали, які часто побудовані не на протоколі IP;
- Локальні обчислювальні мережі: зазвичай це системи обміну даними на основі протоколу IP, наприклад, 802.11 Wi-Fi для швидкого радіозв'язку, часто це зіркоподібні мережі;
- Агрегатори, маршрутизатори, шлюзи: постачальники вбудованих систем, самі бюджетні складові (процесори, динамічна оперативна пам'ять і система зберігання даних), виробники модулів, виробники пасивних компонентів, виробники тонких клієнтів, виробники стільникових і бездротових радіосистем, постачальники межплатформного програмного забезпечення, розробники інфраструктури туманних обчислень, інструментарій для граничної аналітики, безпеку граничних пристроїв, системи управління сертифікатами;
- Глобальна обчислювальна мережа: оператори стільникового зв'язку, оператори супутникового зв'язку, оператори малопотужних(Low-Power Wide-Area Network, LPWAN). Зазвичай застосовуються транспортні протоколи інтернету для IoT і мережевих пристроїв (MQTT, CoAP та навіть HTTP);
- Хмарні технології: інфраструктура в якості постачальника послуг, платформа в якості постачальника послуг, розробники баз даних, постачальники послуг потокової і пакетної обробки даних, інструменти для аналізу даних, програмне забезпечення в якості постачальника послуг, постачальники озер даних, оператори програмно-визначених мереж;

- Аналіз даних: величезні масиви інформації передаються в хмару. Робота з великими обсягами даних і отримання з них користі - це завдання, що вимагає комплексної обробки подій, аналітики і прийомів машинного навчання;
- Безпека: при зведенні всіх елементів архітектури воєдино постають питання безпеки. Безпека стосується кожного компонента: від датчиків фізичних величин до ЦПУ і цифрового апаратного забезпечення, систем радіозв'язку і самих протоколів передачі даних. На кожному рівні необхідно забезпечити безпеку, достовірність і цілісність. У цьому ланцюзі не повинно бути слабких ланок, оскільки інтернет речей стане головною мішенню для атак хакерів в світі.

IoT-архітектура охоплює безліч технологій. Кожен архітектор повинен розуміти, який вплив обране проектне рішення буде надавати на всю систему в цілому і кожну з її частин окремо. Складнощі і багатогранність інтернету речей пов'язані з тим, що ця технологія набагато більш комплексна, ніж традиційні технології: її відрізняє не тільки великий розмах, але і поєднання різних, часто не пов'язаних між собою типів архітектури. Кількість можливих проектних рішень вражає уяву. Наприклад, на даний момент в світі існує більше 700 IoT - провайдерів, що пропонують хмарні сховища, SaaS-компоненти, системи управління IoT, системи безпеки IoT і будь-які види аналізу даних. Додайте сюди величезну кількість різних протоколів персональних, локальних і глобальних мереж, які постійно змінюються і коригуються в залежності від регіону. Вибір невідповідного протоколу персональної мережі може призвести до проблем з обміном даними і помітно низької якості сигналу, і ситуацію можна виправити, тільки додавши більше вузлів в мережу. Архітектор повинен враховувати інтерференцію в локальних і глобальних мережах: яким чином дані знімаються з граничних пристроїв і передаються в інтернет? Архітектор повинен оцінити відмовостійкість системи і вартість можливої втрати даних. Який рівень повинен відповідати за відмовостійкість системи - нижні рівні або рівень протоколу?

Архітектор також повинен вибрати інтернет-протоколи: MQTT или CoAP и AMQP, - а також необхідно продумати, як все це буде працювати в разі переходу на інший хмарний сервіс. Також потрібно вирішити, в якій точці буде здійснюватися обробка даних. На цьому етапі можна розглянути туманні обчислення як спосіб обробки даних поруч з джерелом, що вирішує проблему запізнювання і, що важливіше, дозволяє знизити завантаження мережі і витрати при передачі даних з глобальних мереж і хмарних сервісів. Далі ми розглядаємо всі варіанти аналізу отриманих даних. Невідповідний інструмент аналітики може стати причиною захаращення системи надлишковими даними або змусить вас користуватися алгоритмами, які вимагають занадто великих обчислювальних ресурсів для роботи на граничних вузлах. А як запити, що направляються від хмари до датчика, вплинуть на роботу батарейки самого датчика? Крім усього цього широкого діапазону можливих варіантів, ми не повинні забувати про систему безпеки, оскільки створена нами IoT- система стає найбільшою мішенню для атак в місті. Як ви бачите, вибір величезний і кожне рішення впливає на інші. На даний момент нам доступні більше 1,5 млн різних комбінацій архітектурних рішень (рис. 1.2).

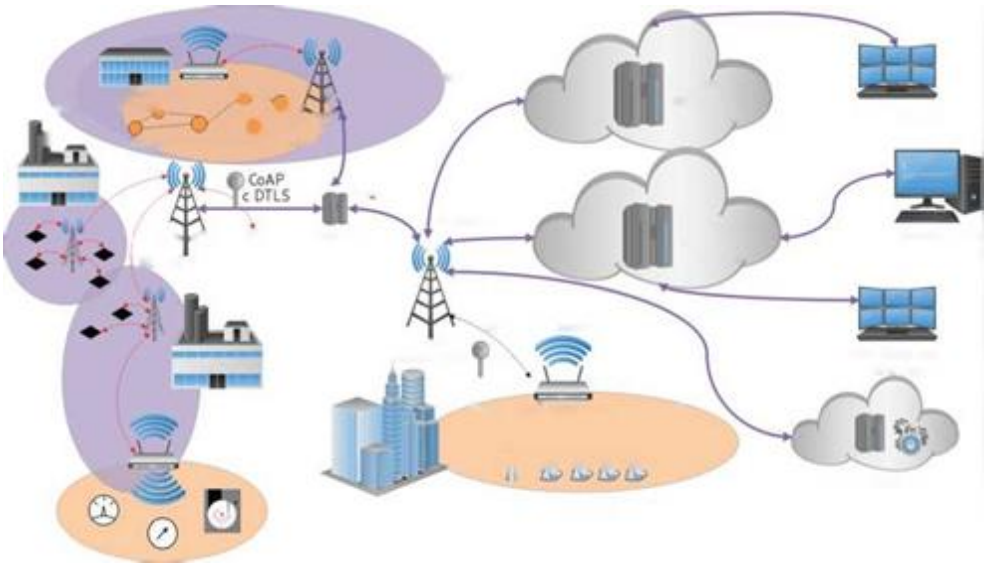

рис. 1.2. Варіанти IoT-рішень. Повний спектр різних варіантів на всіх рівнях IoT-архітектури: від датчика до хмари, і навпаки

1.2 Аналіз даних та машинне навчання у хмарних платформах

Основна цінність IoT-систем полягає не в архівації мільйонів подій, згенерованих датчиками, а в їх інтерпретації та прийнятті відповідних рішень. Світ, в якому мільярди сутностей з'єднуються і взаємодіють один з одним та з хмарними платформами, захоплює дух, але нас в першу чергу цікавлять дані, їх вміст, те, що в них потрапило, і закономірності, які з них можна вивести. Це та область інтернету речей, що відноситься до науки про даних та їх аналізу, є, напевно, найбільш цінною для клієнтів.

В інтернеті речей аналізується така інформація:

- структуровані дані (SQL-сховище) передбачуваного формату;
- неструктуровані дані (необроблене відео або сигнали) високого ступеня випадковості і варіативності;
- частково структуровані дані (Twitter-потоки) з деяким ступенем випадковості варіативності.

Іноді дані необхідно інтерпретувати і аналізувати в режимі реального часу у вигляді потоку, а іноді вони архівуються з подальшим поглибленим аналізом в хмарі. Це стадія споживання даних. У деяких випадках інформацію доводиться зіставляти з іншими джерелами на льоту, а іноді вона просто записується і скидається в так зване озеро даних, на зразок Hadoop.

Далі йде етап переміщення. Такі системи обміну повідомленнями, як Kafka, направляють дані в поточний або пакетний процесор (або в обидва відразу). Дані обробляються у вигляді безперервного потоку. Цей процес зазвичай має певні обмеження і високу продуктивність, тому що інформація обробляється в пам'яті. У зв'язку з цим обробка даних повинна відбуватися не повільніше, ніж їх надходження. І якщо в хмарі можна розраховувати на роботу в режимі, близькому до реального часу, то промислові пристрої або безпілотні автомобілі не дають жорстких гарантій щодо продуктивності.

З іншого боку, пакетна обробка добре підходить для великих обсягів даних, особливо при зіставленні показників датчиків з раніше збереженою інформацією.

Наступний етап може включати в себе прогнозування і повернення відповіді. Дані можуть бути виведені на панель управління, записані в журнал або повернуті назад прикордонному пристрою, яке може вжити певних заходів для вирішення деяких проблем.

1.2.1 Простий аналіз даних в IoT

Аналіз даних займається пошуком подій, як правило, в потоках інформації. Існує кілька видів подій і ролей, які має підтримувати пристрій потокового аналізу в режимі реального часу. Нижче наведені загальні категорії аналітичних функцій:

- попередня обробка - відкидання малоцікавих подій, денатурація, витяг властивостей, сегментація, переклад даних в більш відповідний формат (хоча озера даних воліють уникати негайного перетворення), додавання таких атрибутів, як ярлики (озерам даних ярлики не потрібні);
- оповіщення - перевірка даних; якщо вони не відповідають якимось граничним умовам, відправляється попередження. Елементарний приклад: температура перевищує певний ліміт, встановлений в датчику;
- вікна подій - створюється рухливий діапазон (вікно), в якому діють свої власні правила. Вікно може бути обмежена за часом (наприклад, протягом однієї години) або довжині (наприклад, в рамках 2000 показань датчика). Вікна можуть бути рухомими (наприклад, перевірка останніх 10 подій датчика і повернення результатів при кожному новому подію) і пакетними (коли подія генерується тільки по завершенні вікна). Ця функція добре підходить для створення правил і подій, що займаються підрахунком. Наприклад, ви можете дізнатися

кількість стрибків температури за останню годину і зробити висновок про те, що один з пристроїв має дефект;

- з'єднання - об'єднують кілька потоків даних в один. Можна навести приклад в сфері логістики. Припустимо, компанія розсилає посилки і відстежує їх за допомогою спеціальних маяків, а її численні вантажівки, літаки і об'єкти нерухомості шлють потоки геолокаційних даних. Таким чином, ми маємо два потоки: один для посилки, а інший для заданого вантажівки. Коли вантажівка підбирає посилку, ці два потоки об'єднуються;
- помилки - працюючи з мільйонами датчиків, ви неодмінно зіткнетеся з втратою даних, їх спотворенням або неправильним порядком проходження. Це актуально для інтернету речей з його множинними асинхронними і незалежними потоками. Наприклад, інформація може загубитися в мережі, якщо автомобіль заїде в підземний гараж. Цей аналітичний шаблон зіставляє дані в рамках одного потоку, намагаючись знайти подібні помилки;
- бази даних - аналітичний пакет повинен взаємодіяти з якимось сховищем даних. Наприклад, якщо дані надходять з ряду датчиків, таких як ярлики Bluetooth, які сигналізують про розкрадання або зникнення пристрою, вам потрібно буде зіставляти їх з базою даних ідентифікаторів розшукуваних ярликів;
- тимчасові події та шаблони - ця функція найчастіше застосовується в поєднанні з вікнами подій, згаданими вище. Тут цільовим шаблоном є набори або послідовності подій. Це такий кінцевий автомат. Уявіть, що ми відстежуємо стан пристрою по його температурі, вібраціям і шуму, який воно видає. Послідовність часових подій може виглядати так:

    1) визначити, чи перевищує температура 100 ° C;

    2) визначити, чи перевищує рівень вібрацій 1 м / с;

    3) визначити, видає чи пристрій звук гучністю 110 дБ;

4) відправити оповіщення, якщо всі ці події відбуваються в такий по-послідовності;

- відстеження - подія генерується за фактом наявності або відсутності будь-яких даних в заданому місці або в певний момент часу. Найпростішим прикладом є геолокація для службових автомобілів, коли компанії необхідно знати їх точне місце розташування і коли вони в останній раз відвідували ту чи іншу місце. Цю функцію можна застосовувати в сільському господарстві, комунальному вивезенні сміття, прибирання снігу, для відстеження пасажиропотоків, пацієнтів, цінних активів, багажу і т. Д .;
- тенденції - цей шаблон особливо корисний в прогностичному облуживании. Він має на увазі створення правила для виявлення подій на основі наборів даних, пов'язаних за часом. Тимчасові події мають схожий принцип роботи, але замість часу в них фігурує поняття послідовності. У цій моделі час є одним з вимірів процесу. Актуальна історія подій, пов'язаних з часу, може використовуватися сільському господарстві. Наприклад, до голови корови можна прикріпити датчик, який буде відслідковувати її переміщення і температуру. Щоб подивитися, рухалася чи корова протягом дня, можна скласти послідовність подій. Відсутність руху може вказувати на те, що тварина захворіла або померло;
- пакетні запити - пакетна обробка зазвичай виявляється більш комплексної та глибокої, ніж обчислення в режимі реального часу. Добре спроектована потокова платформа може розділяти аналіз на кілька частин і передавати їх системі пакетної обробки. Ми ще повернемося до цієї теми під час обговорення lambda-функцій;
- глибокий аналіз - при обробці в режимі реального часу рішення про виникнення подій приймаються на льоту. Чи повинно це подія згенерувати повідомлення - це питання, відповідь на який може

зажадати додаткової обробки, яка виконується пізніше. Такий підхід працює, оскільки події подібного роду повинні бути рідкісними; і, поки нові події створюються в режимі реального часу, старі передаються в движок глибокої обробки. Як приклад можна привести систему відеоспостереження. Уявіть, що платформа розумного міста б'є на сполох, якщо зникла дитина, і запускає просту модель розпізнавання та класифікації зображень для поточних движків реального часу. Модель здатна виявити номерний знак автомобіля, в якому перебуває дитина, або навіть логотип на дитячій футболці. Насамперед потрібно буде захопити зображення з номерними знаками і логотипами на одязі перехожих і передати їх в хмару. З мільйонів отриманих зразків система аналізу спробує розпізнати потрібний номер або логотип. Це перший етап. Потім, щоб виключити помилкові спрацьовування, розпізнаний кадр (разом з сусідніми кадрами) буде направлений в більш просунуту систему аналізу, яка використовує більш докладні алгоритми розпізнавання об'єктів (злиття зображень, збільшення дозволу, машинне навчання);

- моделі і навчання - модель першого рівня, описана вище, насправді може бути механізмом логічного висновку для системи машинного навчання. Подібні системи будуються на тих, яких навчають моделях і можуть використовуватися на льоту при аналізі в режимі реального часу;
- обмін сигналами - дії часто доводиться повертати назад прикордонному компоненту або датчику. Типовим прикладом є автоматизація і безпеку фабрики. Якщо температура пристрою перевищить певний ліміт, ця подія потрібно не тільки записати в журнал, а й відправити прикордонному компоненту, щоб уповільнити пристрій. Механізм комунікацій в системі повинен бути дуплексним;

- управління - нарешті, у нас повинна бути можливість управляти всіма цими засобами аналізу. Для роботи з системою повинні бути передбачені такі функції як запуск, зупинка, створення звітів, запис в журнал і налагодження.

1.2.2 Верхній рівень хмарної архітектури

На рис. 1.3 показаний типовий шлях прямування даних від датчика до панелі управління. Дані проходять через кілька проміжних ланок (широкосмуговий канал, хмарне сховище у вигляді озера даних). При виборі архітектури для побудови хмарного аналітичного рішення слід враховувати ефект масштабування. Рішення, прийняті на ранніх етапах проектування, можуть згодитися для десятка IoT-вузлів і одного хмарного кластера, але, коли число кінцевих IoT-пристроїв зросте до тисяч, а система почне охоплювати кілька географічних зон, масштабування може виявитися неефективним.

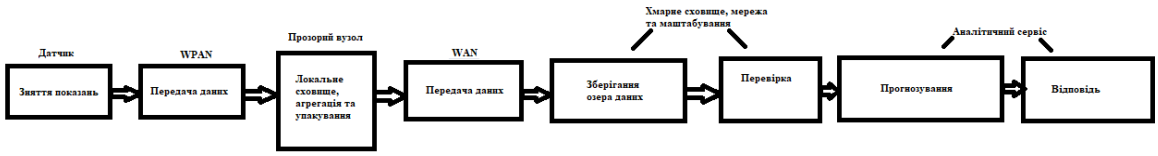

Рис. 1.3. Типовий шлях прямування даних від датчика до панелі управління

Аналітичні функції хмарної системи (прогнозування-відповідь) можуть приймати кілька форм:

- система правил - визначає дію і повертає результат;
- потокова обробка - тут події, такі як показники датчиків, передаються поточному процесору. Шлях обробки представляє собою граф з операторами в якості вузлів; при цьому дані переходять від одного оператора до іншого. Кожен вузол містить код для певного етапу обробки та посилання на наступний вузол в графі.

Такий граф можна реплікувати і виконувати паралельно в кластері, тому його можна масштабувати на сотні комп'ютерів;

- обробка складних подій - ця частина системи заснована на мові запитів високого рівня і займається обробкою подій. Вона заточена під низькі затримки;
- Lambda-архітектура - ця модель намагається збалансувати пропускну здатність системи і рівень затримок, виконуючи пакетну обробку і паралельні обчислення з величезними наборами даних.

Причина, по якій слід обговорювати аналіз в режимі реального часу, пов'язана з тим, що дані одночасно надходять з мільйонів вузлів; це відбувається безперервно і асинхронно, з різними помилками, проблемами форматування і розбіжностями по часу. У Нью-Йорку встановлено 250 000 вуличних ліхтарів. Уявіть, що кожен ліхтар є розумним - тобто, він регулює свою яскравість в залежності від руху навколо себе, і, якщо поблизу нікого немає, він залишається затемненим і економить електроенергію (на це йде 2 байта). Кожен ліхтар може відстежувати ліхтарі, які вимагають ремонту (1 байт). Крім того, він вимірює температуру (1 байт) і вологість (1 байт), допомагаючи генерувати локальний прогноз погоди. Нарешті, дані, з якими він працює, містять його ідентифікатор і тимчасову мітку (8 байт). У штатному режимі ліхтарі генерують, в цілому, 250 000 повідомлень в секунду, хоча в години пік і на свята (а також через велике скупчення людей і завдяки туристичних пам'яток) цей показник може досягати 325 000. Припустимо, наш хмарний сервіс здатний обробляти 250 000 повідомлень в секунду - тобто, відставання може досягати 75 000 подій в секунду. Якщо годину пік дійсно затягнеться рівно на 1 годину, за цей час у нас накопичиться 270 000 000 необроблених подій. Щоб дати системі шанс надолужити згаяне, нам доведеться збільшити обчислювальну потужність кластера або зменшити вхідний потік. Якщо кожну секунду в періоди низької активності прибирати з потоку 200 000 повідомлень, хмарному кластеру

знадобиться 1,1 години і 585 Мб пам'яті (270 мільйонів подій за 13 байт кожне), щоб нівелювати відставання.

1.2.3 Система правил

Система правил - це програмний компонент, який виконує якісь дії у відповідь на події. Наприклад, якщо вологість в кімнаті перевищить 50%, власнику відправляється SMS. Цей механізм також називають системою управління бізнес-правилами.

Деякі системи правил володіють статками. Це означає, що вони можуть зберігати історію подій і виконувати різні дії в залежності від їх порядку, кількості або характеру їх надходження. Системи, що не мають стану, перевіряють тільки поточну подію (рис. 1.4).

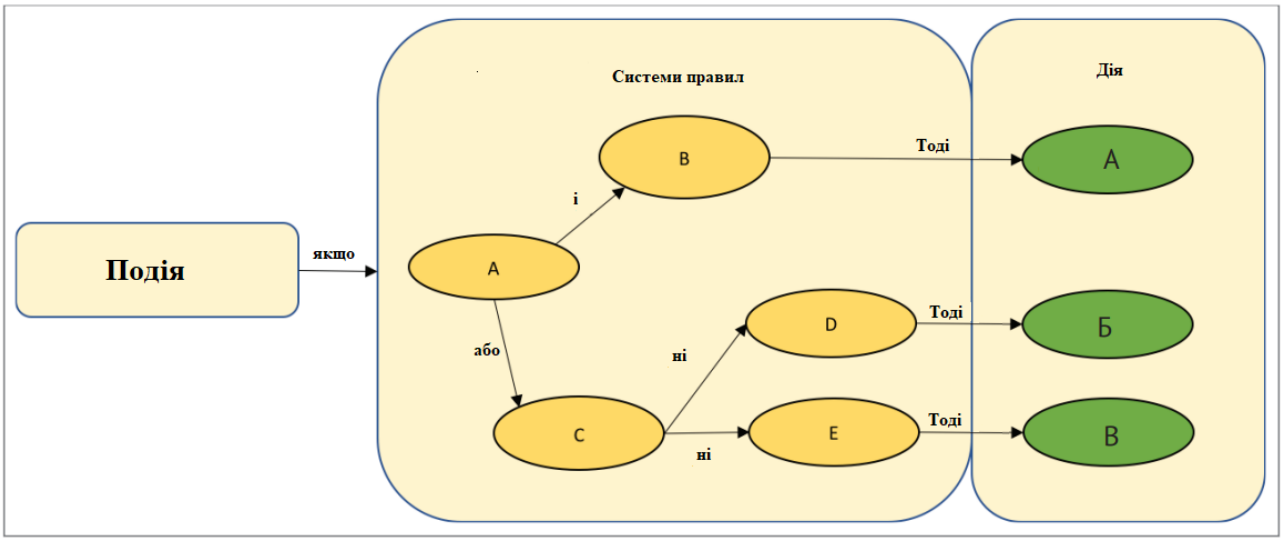

Рис.1.4. Приклад простої системи правил

1.2.4 Споживання інформації: потоки, обробка та Big Data

IoT-пристрій зазвичай пов'язаний з якимось датчиком або іншим пристроєм, який вимірює або відстежує умови в реальному часі. Це робиться асинхронно по відношенню до решти технологій інтернету речей. Тобто, датчик

намагається транслювати дані незалежно від того, чи слухає його хмарний або туманний вузол. Це важливо, оскільки в багатьох організаціях дані представляють найбільшу цінність. Навіть якщо більша частина інформації виявляється надлишкової, в якийсь момент може статися важлива подія. У зв'язку з цим дані передаються у вигляді потоків.

Потік інформації, що передається з датчика в хмару, сприймається як:

- Постійний і нескінченний;
- Асинхронний;
- Неструктурований або структурований;
- Максимально близький до режиму реального часу.

Слід відзначити, що тут мають місце тимчасові затримки, властиві хмарним технологіям. Туманні обчислення допомагають вирішити цю проблему. Але і без туманних вузлів робляться зусилля для оптимізації хмарної архітектури, щоб забезпечити підтримку режиму реального часу в інтернеті речей. Для цього хмара має вміти працювати з безперервним потоком даних. Альтернативою є пакетна обробка. Більшість апаратних платформ використовують одні і ті ж методи роботи з потоками, переміщаючи дані з одного блоку в інший; при цьому на кожному етапі їх надходження активізує наступну функцію. Важливим елементом зниження загальної латентності є ретельне використання сховища даних і доступ до файлової системи.

У зв'язку з цим більшість поточних фреймворків виконують свої операції в пам'яті, намагаючись повністю виключити тимчасове зберігання інформації в файлової системі. Цьому шаблону сприяє добре спроектована черга повідомлень. Побудова успішної хмарної архітектури, здатної маштабуватись на сотні і мільйони вузлів, вимагає ретельного планування.

До того ж, потоки даних рідко бувають ідеальними. У ситуації, коли сотні тисяч датчиків асинхронно передають свої показання, деякі дані будуть загублені (при втраті зв'язку з датчиком), мати некоректний формат (помилка при

передачі) або надходити не в тому порядку (якщо хмара приймає інформацію з кількох джерел). Поточна система повинна як мінімум:

- Маштабуватись при рості кількості подій;
- Давати можливість публікації/підписки у своєму API-інтерфейсі;
- Прикладати зусилля на мінімізацію затримок;
- Забезпечувати маштабування правил обробки;
- Підтримувати озера і сховища даних.

Якщо наша IoT-система складається з тисяч або мільйонів датчиків і кінцевих вузлів, в хмарному середовищі має сенс використовувати озеро даних. Озеро Даних - це, по суті, величезне сховище, в яке стікається необроблена і невідфільтрованого інформація з багатьох джерел. На відміну від звичайних файлових систем воно абсолютно плоске і не підтримує ієрархії, тому, директорії, файли і папки. Для структурування вмісту до кожного запису прикріплюється елемент метаданих (ярлик). Класичним прикладом озера даних є Apache Hadoop, хоча практично всі хмарні провайдери використовують цю модель в тому чи іншому вигляді.

Озеро даних відмінно підходить для Інтернету речей, дозволяючи зберігати інформацію незалежно від того, структурована вона чи ні. Всі дані в ньому вважаються цінними і зберігаються на постійній основі. Таке скупчення інформації є оптимальним для систем аналізу даних. Якість роботи алгоритмів залежить від обсягу даних, які вони споживають або використовують для навчання своїх моделей.

На рис. 1.5 представлена концептуальна архітектура на основі традиційної пакетної і потокової обробки. Хмара Kafka надає пакетний інтерфейс до Spark і передає дані в сховище.

Компоненти використовують стандартні з'єднання, тому топологію даної архітектури можна модифікувати декількома способами.

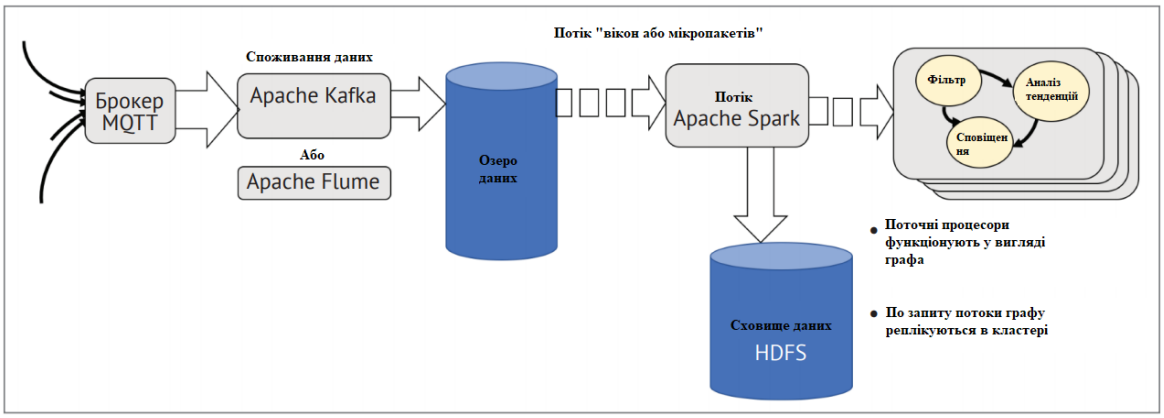

Рис. 1.5. Принцип передачі даних у хмарне сховище.

Висновки до розділу 1

У першому розділі було розглянуто базову архітектуру Інтернет речей та дано описано ролі технології у кожному сегменті інформаційної галузі. Також, було проаналізовано процес аналізу даних та можливість застосування машинного навчання у хмарних платформах.

У наступному розділі буде безпосередньо розглянуто суть, основні методи та алгоритми машинного навчання.

2. АНАЛІЗ МЕТОДІВ ТА АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

2.1 Загальні відомості

З теоретичної точки зору машинне навчання - дисципліна, що знаходиться на перетині математичної статистики, чисельних методів оптимізації, теорії ймовірностей, а також дискретного аналізу. За допомогою її методів відбувається вирішення задачі отримання знань з даних. Згідно [1]: «В теорії штучного інтелекту та експертних систем, знання - це сукупність тверджень про світ, властивості об'єктів, закономірності процесів і явищ, а також правил логічного висновку одних тверджень з інших і правил використання їх для прийняття рішень. Головна відмінність знань від даних полягає в їх структурності і активності: поява в базі знань нових фактів або встановлення нових зв'язків між ними може стати джерелом змін в прийнятті рішень ».

З практичної ж точки зору машинне навчання націлене на створення систем, здатних адаптуватися до вирішення різних завдань без явного створення алгоритму, тобто систем, здатних навчатися.

2.2 Задачі машинного навчання

Як було сказано вище, машинне навчання націлене на створення систем, здатних отримувати знання з даних, а також здатних за допомогою навчання покращувати показники своєї роботи. Таким чином, можна стверджувати, що машинне навчання є однією з областей науки про дані (DataScience). На рис.2.1 представлені її складові і показано місце машинного навчання серед них.

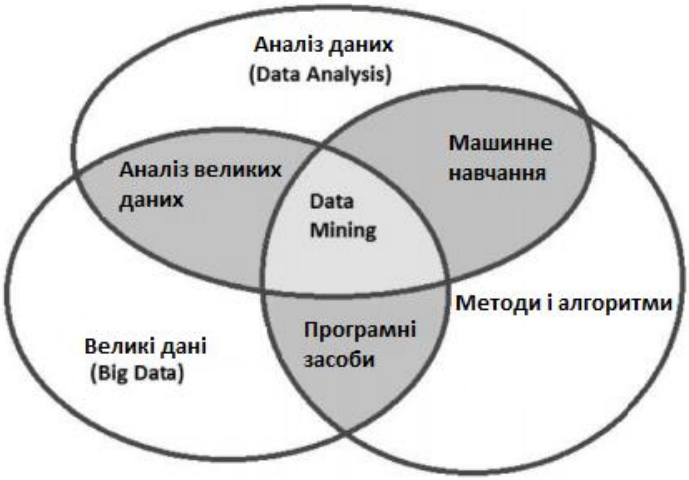

Рис.2.1. Складові частини DataScience

Як видно з рис.2.1, машинне навчання перетинається з аналізом даних, тобто має спільні риси, але також і свою специфіку. Аналіз даних - це величезна і вже відносно не молода область математики та інформатики. Під неї потрапляє будь-яка обробка даних. Наприклад, обробка сигналу методом Фур'є, підрахунок мінімального і максимального значень або відношення якихось величин може бути аналізом даних, якщо ці операції тягнуть за собою важливі висновки, знання про дані і дозволяють вирішити поставлену задачу. Так отримання дисперсії якоїсь величини є аналізом даних, але не є в чистому вигляді машинним навчанням. Хоча важливо відмітити, що методи статистики мають велике значення і в машинному навчанні. Відмінною особливістю методів машинного навчання є те, що вони здатні вирішувати широкий спектр завдань без явного задання та опису алгоритму рішення. Наприклад, метод регресії входить в машинне навчання. Регресія давно застосовується в економіці для прогнозування і оцінки різноманітних параметрів, тому що дана математична модель в деякому роді універсальна. Однак варто відмітити, що регресія з'явилася в математиці набагато раніше, ніж її стали використовувати в економіці, а ось машинним навчанням це стало тільки тоді, коли всі ці методи почали розглядатися не як відокремлені механізми вирішення завдань зі спеціальних областей, а як методи перетворення інформації взагалі.

На рис.2.2 представлена діаграма розподілу завдань, які доводиться вирішувати у сфері машинного навчання.

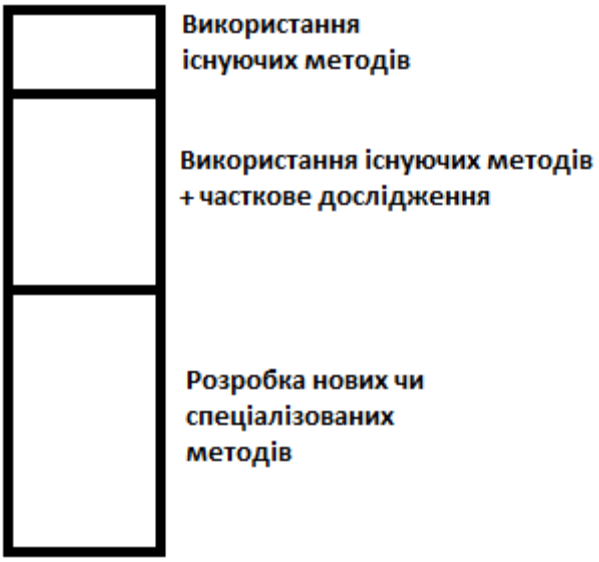

Рис.2.2. Діаграма розподілу завдань у сфері машинного навчання

Зрозуміло, що розробка програм в області машинного навчання досить дорога. Тому застосовувати його слід лише у випадках, коли це дійсно необхідно. У таблиці 1 представлені деякі ознаки, за якими можна прийняти рішення про необхідність застосування до задачі методів машинного навчання.

Таблиця 2.1 Ознаки застосування машинного навчання

Доцільно використання класичних методів	**Доцільно використання методів машинного навчання**
Наявність чітко формалізованого алгоритму.	Відсутність чітко сформованого формалізованого алгоритму з огляду на високу варіативність рішення.
Не допускається невизначеність поведінки моделі.	Допускається невизначеність поведінки моделі.
Економічна доцільність	

Крім того, для розробки рішень на основі МН необхідні оцифровані дані. Вони є ключовим аспектом, від якого залежить якість і повнота рішення. Таким чином, дуже важливо розуміти, що для застосування методів машинного навчання потрібні зібрані і спеціально підготовлені дані.

Як правило, процес машинного навчання проходить кілька етапів. На рис.2.3 представлений узагальнений алгоритм.

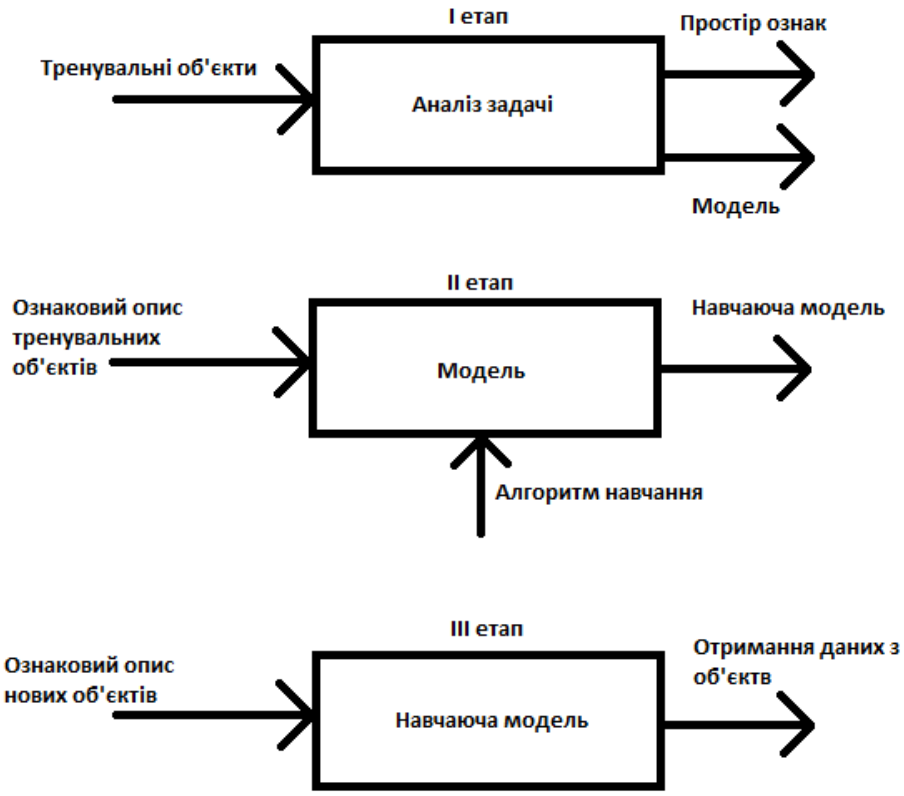

Рис.2.3. Узагальнений алгоритм процесу машинного навчання

Необхідно відзначити, що дана ілюстрація відображає ідеальний випадок, коли після навчання ми відразу отримали працюючу модель. Однак в реальності, як правило, між другим і третім етапом є проміжне, але дуже важлива дія - оцінка якості моделі. І саме за його результатами приймається рішення про перехід на наступний етап або про повернення до одного з попередніх.

Серед узагальнених задач машинного навчання можна виділити наступні:

1. Регресія (інколи зустрічається термін «апроксимація»);
2. Класифікація;

3. Кластеризація;

Завдання регресії - наближення невідомої цільової залежності на деякій множині даних. Нехай X - безліч даних - описів деяких об'єктів. Y - безліч можливих відповідей для X.

У задачі регресії передбачається, що існує невідома цільова залежність y: $X \to Y$, значення яких відомі тільки на об'єктах навчальної вибірки $XY = \{(x_1, y_1) \dots (x_n, y_n)\}, x \in X, y \in Y$. Необхідно отримати алгоритм $a: X \to Y$, що наближує цільову залежність як на множині XY, так і на X. Тобто вирішити задачу регресії - означає знайти алгоритм, який має здатність до узагальнення емпіричних фактів (здатністю до висновку загальних знань з приватних спостережень, прецедентів).

Завдання класифікації - розподіл деякої множини об'єктів по заданій множині груп (класів). При цьому є деяка підмножина об'єктів, для яких розподіл по класах відомий, класова приналежність решти - невідома.

Потрібно побудувати алгоритм, який вказував би класову приналежність для будь-якого об'єкта з початкової множини.

Формально постановку задачі класифікації можна описати таким чином. Нехай X - безліч даних - описів деяких об'єктів. Y - кінцева множина класів, відмічених мітками. Існує невідома цільова залежність - відображення y: $X \to Y$, чиї значення відомі тільки на об'єктах навчальної вибірки значення яких відомі тільки на об'єктах навчальної вибірки $XY = \{(x_1, y_1) \dots (x_n, y_n)\}, x \in X, y \in Y$. Необхідно отримати алгоритм $a: X \to Y$, що здатний класифікувати довільний об’єкт $x \in X$.

Як можна помітити, дана задача схожа з попередньою. Однак головна особливість завдання регресії полягає в тому, що функція a: $X \to Y$ є безперервною речовою функцією. Завдання класифікації відрізняється тим, що Y - дискретна множина. Крім того, на відміну від задачі апроксимації у задачі класифікації виділяють кілька типів. За кількістю класів можна виділити:

1. Класифікацію на два класи: множина Y містить всього дві мітки.

2. Класифікацію на безліч класів: *У* містить від трьох до кількох тисяч міток.

По характеру розділення об’єктів на класи можна виділити:

1. Класифікацію на непересічні класи: один об'єкт належить тільки одному класу.

2. Класифікацію на пересічні класи: один об'єкт може належати кільком класам.

3. Класифікацію на нечіткі множини: об'єкт належить всім класам з певним ступенем приналежності.

Завдання кластеризації - поділ дуякої множини об'єктів на непересічні групи (кластери) таким чином, щоб кожна група складалася з подібних об'єктів, а об'єкти різних кластерів істотно відрізнялися.

Формально постановку задачі кластеризації можна описати таким чином. Нехай X - множина даних - описів деяких об'єктів. Y - безліч кластерів, відмічених мітками. Визначена функція відстані між об'єктами з початкової множини X: $f(x, x`)$, і є деяка навчальна вибірка об'єктів $X_o = \{x_n, y_n\}, x \in X$. Необхідно розбити навчальну вибірку на кластери, приписавши кожному x номер кластера y_i, так, щоб близькі по метриці f об'єкти належали одному кластеру, а об'єкти різних кластерів істотно відрізнялися за метрикою f. Тобто необхідно побудувати алгоритм a: $X \to Y$, який будь-якому $x \in X$ ставить у відповідність номер кластера $y \in Y$. Причому, іноді безліч Y відомо заздалегідь, але частіше все-таки ставиться завдання отримати оптимальне число кластерів, виходячи з характеру даних. Оптимальність оцінюється по якомусь критерію якості кластеризації.

Завдання кластеризації складніше апроксимації і класифікації. Це обумовлено наступними причинами:

1. Немає однозначного критерію якості кластеризації. Існує ряд евристичних критеріїв, що виконують цілком осмислену кластеризацію, але дають на одних і тих же даних різні результати.

2. Число кластерів, як правило, заздалегідь невідомо і задається суб'єктивно.

3. На результат кластеризації істотно впливає обрана метрика відстані, яка, як правило, також вибирається суб'єктивно.

Однак, незважаючи на описані вище складності, кластеризація допомагає досягти наступні цілі:

1. Покращити розуміння даних за рахунок виявлення їх кластерної структури: розбиття об'ємної вибірки на групи схожих об'єктів може спростити подальшу обробку даних за рахунок застосування до кожного кластеру своїх методів аналізу.

2. Здійснити стиснення даних. В даному випадку мається на увазі скорочення об'ємної вибірки за рахунок роботи з найбільш яскравими представниками кластерів (груп схожих об'єктів).

3. Виявити новизну в масиві даних: виявити нетипові об'єкти, які не вдається віднести ні до одного кластеру.

4. Вирішити задачу таксономії: побудувати деревоподібну ієрархічну структуру, що упорядковує вихідні дані. Її побудова досягається за рахунок дроблення великих кластерів на більш дрібні, які в свою чергу також дробляться на ще більш дрібні. Візуально таксономія відображається у вигляді графіка дендрограмми.

2.3 Способи навчання і оцінка їх якості

Як було сказано вище, основна характеристика систем, що розробляються за допомогою методів машинного навчання, - здатність до навчання. Залежно від видів розв'язуваних завдань застосовують різні алгоритми реалізації цієї ключової особливості. В рамках даного посібника розглянемо три основних види навчання, а також визначимо класи завдань, які підходять для кожного з цих видів.

Перший тип - навчання з учителем. Формально він був описаний раніше, тому тут не будемо на цьому зупинятися детально. Коротенько ж навчання з учителем можна описати таким чином: дано деякий безліч об'єктів і безліч можливих реакцій системи на ці об'єкти. При цьому відповіді і об'єкти пов'язані між собою деякої невідомої залежністю. Є кінцева сукупність пар об'єкт-відповіді (прецедентів), звана навчальної вибіркою. На її основі необхідно виявити алгоритм, який згодом для будь-якого об'єкта з початкової множини дасть досить точну відповідь. Для вимірювання точності відповідей використовується один з функціоналів якості, як правило, зав'язаний на обчисленні відхилення отриманої відповіді від очікуваного, тобто обчисленні помилки. Розглянемо деякі їх види. У наведених нижче формулах можуть використовуватися такі символи:

$$XY = \{(x_1, y_1) \dots (x_n, y_n)\} \tag{2.1}$$

де n – кількість прецедентів;

y_i – фактичне значення в *i*-му прецеденті.

1. Середня помилка є усереднення помилок для кожного зразка і обчислюється:

$$\text{СО} = \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})}{n} \tag{2.2}$$

2. Середня абсолютна помилка є усереднення абсолютних помилок на кожному кроці і обчислюється:

$$\text{САО} = \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})}{n} \tag{2.3}$$

3. Середньоквадратична помилка обчислюється як сума середніх квадратів помилок:

$$\text{СКО} = \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{n} \tag{2.4}$$

4. Середня відносна помилка обчислюється як середнє відносних помилок:

$$\text{СО} = \frac{\sum_{i=1}^{n} \frac{(y_i - \hat{y_i})}{y_i}}{n} \times 100\% \quad (2.5)$$

5. Середня абсолютна відносна помилка обчислюється як середнє відносних помилок по модулю:

$$\text{MAPE} = \frac{\sum_{i=1}^{n} \left| \frac{(y_i - \hat{y_i})}{y_i} \right|}{n} \times 100\% \quad (2.6)$$

6. Симетрична середня абсолютна відносна помилка:

$$SMAPE = \frac{\sum_{i=1}^{n} \left| \frac{(y_i - \hat{y_i})}{y_i/2} \right|}{n} \times 100\% \quad (2.7)$$

У всіх представлених оцінок якості є свої переваги і недоліки. Наприклад, у першій і п'ятій недолік полягає в тому, що позитивні і негативні помилки анулюють одна одну, тому в деяких випадках вони не є достатньо хорошими індикаторами якості. У зв'язку з цим найчастіше використовується третя або четверта заходи.

Навчання з учителем використовується при вирішенні задач апроксимації та класифікації. У першому випадку відповіді є дійсними числами або векторами, у другому - вибираються з кінцевого безлічі міток-класів. Необхідно відзначити, що при-ведені вище формули підходять тільки для випадків, коли відповідь системи і необхідний відповідь - дійсні числа, одержувані при вирішенні задачі апроксимації. У задачах класифікації же оцінка якості найчастіше зав'язана на співвідношенні кількостей правильно і неправильно зарахованих до класів об'єктів.

Другий тип навчання - навчання без учителя. Формально постановку задачі навчання без вчителя можна описати таким чином. Нехай *X* - безліч даних - описів деяких об'єктів. Необхідно знайти безліч *У*, що складається з взаємозв'язків *f*: *(x, x ')* між об'єктами з *X (x, x' є X, f є У).* Якість виявлення

взаємозв'язків перевіряється деякої метрикою, обраної виходячи з розв'язуваної задачі.

Навчання без вчителя використовується для вирішення наступних типів завдань:

1. Завдання кластеризації.
2. Пошук правил асоціації.
3. Скорочення розмірності даних.
4. Візуалізація даних.

Певною мірою, кожна з останніх трьох завдань є похідною від першої або її окремим випадком. Розглянемо докладніше формулювання названих завдань.

Під завданням пошуку правил асоціацій мається на увазі виявлення в ознакових описах об'єктів (вихідних даних) таких наборів і значень ознак, які особливо часто (не випадково часто) зустрічаються у вихідних даних. Якщо ж проводити аналогію з першим завданням, то кожне правило в даному випадку може бути представлено як кластер.

Завдання скорочення розмірності даних полягає в наступному. Існує великий (значно великий) обсяг ознакових описів об'єктів. Причому цей обсяг обумовлюється значною кількістю вимірів простору ознак. Необхідно уявити ті ж дані в просторі меншої розмірності, при цьому мінімізувавши втрати інформації. Угруповання по кластерам якраз і буде одним з варіантів вирішення проблеми.

Завдання візуалізації даних є по суті окремим випадком попередньої: її мета - представити вихідні дані в видимій частині просторі, тобто просторі розмірності 2 або 3.

Як випливає з описаного вище, навчання без учителя в якійсь мірі так чи інакше зводиться до кластеризації. Тому для оцінки якості навчання даними способом, як правило, використовують метрики якості кластеризації. Причому при їх виборі враховується, що ці метрики не повинні залежати від вихідних

даних, а тільки від результатів розбиття. Всі оцінки якості можна розділити на зовнішні і внутрішні.

Перші використовують зовнішню інформацію про справжній розбитті об'єктів на кластери, другі спираються тільки на набір вихідних даних, тобто дані метрики можуть працювати з нерозміченою вибіркою, коли заздалегідь не відомо справжнє розбиття об'єктів на групи. І саме з їх допомогою визначають оптимальне число кластерів.

Наведемо як приклад метрики, виділені компанією ODS[2]:

1. Adjusted RandIndex (ARI). Дана метрика відноситься до групи зовнішніх: передбачається, що справжні мітки об'єктів відомі (наприклад, задані експертом), однак від самих значень міток вона не залежить. Тільки від розбиття вибірки на кластери. Розраховується міра наступним чином: нехай n - число об'єктів у вибірці, a - число пар об'єктів, що мають однакові мітки і знаходяться в одному кластері, b - число пар об'єктів, що мають різні мітки і знаходяться в різних кластерах. Тоді можна порахувати частку об'єктів, для яких поточна й отримане в результаті кластеризації розбиття узгоджені:

$$RI = \frac{2(a+b)}{n(n-1)} \tag{2.8}$$

Отримана величина називається RandIndex і висловлює схожість двох різних кластеризації однієї і тієї ж вибірки. Щоб цей індекс давав значення, близькі до нуля, для випадкових кластеризації при будь-якому n і числі кластерів, необхідно провести його нормування, тобто отримати Adjusted RandIndex:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[R1]} \ , \tag{2.9}$$

де E – математичне очікування.

Міра *ARI* симетрична і не залежить від перестановок і значень міток. По суті цей індекс є мірою відстаней між різними розбивками вибірки. Його область значень - [-1,1]. Інтерпретувати її інтервали можна наступним чином: для «незалежних» розбиття на кластери - негативні значення, для випадкових розбиття - близькі до нуля, для подібних розбиття - поклади тільні значення, причому, *ARI* = 1 говорить про збіг розбиття.

2. Adjusted MutualInformation. Дана метрика схожа з попередньою: вона також симетрична і не залежить від значень і перестановок міток. Для її визначення використовується функція ентропії. Розбиття вибірки інтерпретуються як дискретні ймовірності: ймовірність віднесення до кластеру дорівнює частці об'єктів в ньому. Як і в попередньому випадку, для даної метрики необхідно обчислити спеціальний індекс - MutualInformation (MT). Дана величина визначається як взаємна інформація для двох розподілень, відповідних розбиття вибірки на кластери. Інтерпретувати це можна як частку інформації, загальною для обох розбиття: наскільки інформація про одного з них зменшує невизначеність щодо іншого. Цей індекс розраховується за такою формулою:

$$MI(XY) = \sum_{y \epsilon Y} \sum_{x \epsilon X} p(xy) \log \frac{p(xy)}{p(x)p(y)} \ , \qquad (2.10)$$

де *p(x, y)* – спільна функція розподілення ймовірностей для *X* та *Y;*

p(x) та *p(y)* – функції розподілення граничної ймовірності для *X* та *Y.*

Областю значення індексу *AMI* є діапазон [0,1]. Інтерпретується він у такий спосіб: значення, близькі до нуля, кажуть про незалежність розбиття, а близькі до одиниці - про їх схожості (або збігу при *AMI* = 1).

3. Гомогенність, повнота, V-міра. Дані метрики розглядають розбиття вибірки як дискретні розподіли і визначаються з використанням функції ентропії і умовної ентропії.

$$h = 1 - \frac{H(\mathrm{C}/K)}{H(C)} \ , \tag{2.11}$$

де *K* – результат кластеризації;

C – істинне розбиття;

H – функція ентропії.

Ці заходи приймають значення в діапазоні [0,1]. Інтерпретувати їх можна наступним чином: чим більше значення, тим точніша кластеризація. Ці заходи не є симетричними і не є нормалізованими, на відміну від розглянутих раніше, і тому вони залежать від числа кластерів. Згідно ресурсу [2]: «Випадкова кластеризація не даватиме нульові показники при великому числі класів і малому числі об'єктів. У цих випадках перед-шанобливе використовувати *ARI*. Однак при числі об'єктів більше 1000 і числі кластерів менше 10 дана проблема не так явно виражена і може бути проігнорована ».

Щоб врахувати значення обох величин, вводиться симетрична К-міра, що показує, наскільки кластеризації схожі між собою. Її розрахунок відбувається за формулою:

$$h = 2\frac{hc}{h + c} \tag{2.12}$$

Силует. Дана метрика відноситься до типу внутрішніх: вона не передбачає знання про справжній розбитті і спочатку розраховується для кожного об'єкта в такий спосіб. Нехай *a* - середня відстань від даного об'єкта до об'єктів з того ж кластера, *b* - середня відстань від даного об'єкта до об'єктів з найближчого кластера (але не того, в якому знаходиться сам об'єкт). Тоді силуетом даного об'єкта буде величина, що обчислюється за формулою:

$$h = \frac{b - a}{\max(ab)} \tag{2.13}$$

Силуетом же всієї вибірки буде середня величина силуетів її об'єктів. Інтерпретувати метрику можна наступним чином: вона показує, наскільки

середня відстань до об'єктів свого кластера відрізняється від середньої відстані до об'єктів інших кластерів. Область значення цієї величини - діапазон [-1,1]. При цьому значення, близькі до лівої кордоні, відповідають поганим (розрізненим) кластеризації; значення, близькі до середини, кажуть про пере-перетині і накладення кластерів; значення, близькі до правої кордоні, відповідають «щільним», чітко виділеним кластерам. Згідно [2]: «чим більше силует, тим чіткіше виділені кластери, і вони є компактними, щільно згруповані хмари точок».

Дана величина може бути використана для вибору оптимального числа кластерів, якщо воно заздалегідь невідомо: воно буде дорівнює числу, максимізуючому значення силуету. На відміну від розглянутих раніше метрик, дана величина залежить від форми кластерів. Силует досягає великих значень при більш опуклих кластерах, які виходять за допомогою алгоритмів, заснованих на відновленні щільності розподілу.

2.4 Підготовка вихідних даних та навчання моделей

2.4.1 Врахування пропусків

В реальних задачах дані можуть містити пропуски. Це може бути викликано тим, що клієнти не заповнювали всі поля в анкеті або акаунті, або тим, що не всі параметри були оцифровані за весь час роботи системи.

Таким чином, постає питання про те, як інтерпретувати пропуски. Найпростіший варіант полягає у виключенні об'єктів, що мають неповні відомості (тобто видалення тих рядків з матриці ознак, в шпальтах яких є пропуски), або виключення ознак, що містять неповні відомості (тобто видалення тих стовпців, в яких є пропуски). Плюси цього варіанту в тому, що він дуже простий і його можна відразу випробувати на будь-якої моделі. Мінуси цілком очевидні. Якщо багато об'єктів будуть мати про-пуски, то ми ризикуємо

видалити важливі аспекти закономірності, яка прихована в даних, і, як наслідок, отримаємо низьку якість апроксимації. Якщо ми видалимо стовпець, який містить вкрай важлива ознака, ми ризикуємо зовсім втратити ключову інформацію про розділення об'єктів.

Другий варіант боротьби з пропусками полягає в тому, щоб замінити їх за допомогою інтерполяції. Це може бути середнє або медіанне значення по стовпцю. У разі якщо ознака є функцією від часу, як і окремі об'єкти, то можна інтерполювати пропуски тільки по сусідніх тимчасовим значенням (наприклад, якщо об'єкт є ймовірність покупки квартири, а ознака - середню заробітну плату програміста в цей рік, то за пропущений рік цифру можна наблизити по двом сусіднім).

Третій варіант можна також розглянути на прикладі з квартирою, який був описаний вище. Якщо мова йде про якісь відкритих економічних, соціальних або інших параметрах, то їх можна знайти в інших джерелах і тим самим доповнити дані.

Четвертий варіант полягає в тому, щоб закодувати пропуски спеціальним числовим значенням (це може бути число, не зустрічається в інших об'єктів) або категоріальним значенням (подання категоріальних ознак буде розглянуто нижче). Такий підхід, як мінімум, дозволить внести інформацію про те, що ці об'єкти за цією ознакою відрізняються від інших (тобто ми не повідомимо точної інформації, але не видалимо об'єкти повністю, як в першому підході).

П'ятий підхід полягає в залученні експерта у відповідній темі, який зможе сказати, як саме найкраще інтерполювати дані (на основі специфічних математичних моделей, які, ймовірно, існують або можуть підійти). Наприклад, для якихось ознак можна згенерувати псевдовипадкові значення, що підкоряються нікому розподілу з урахуванням інших відомих ознак. Сюди ж можна віднести генерацію синтетичних даних (тобто штучних) на підставі власного розуміння предметної області. Довгий і ретельний аналіз даних може прояснити поведінку прихованого параметра, а також виявити способи, як

симулювати його з певною вірогідністю. Однак цей підхід небезпечний тим, що неправильне розуміння даних призведе до того, що ми закладемо в дані інші закономірності і потім їх же і знайдемо за допомогою методів машинного навчання. Іншими словами, замість вирішення вихідної задачі ми вирішимо штучно створену.

У складних випадках на практиці застосовується деяка комбінація всіх перерахованих вище підходів. Якісь об'єкти або стовпці повністю виключаються, тому що їх занадто складно інтерполювати або симулювати. Як правило, це об'єкти або ознаки з великим числом пропусків (наприклад, де пропусків більше, ніж реальних значень). Якісь ознаки позначаються особливим кодом, якісь дані знаходяться ззовні, а якісь емулюються синтетичними даними.

Багато нюанси залежать від специфіки завдання. Але якщо ніякі ідеї не працюють, то, ймовірно, даних просто недостатньо і потрібно отримувати ще безпосередньо вихідні дані (проводити фізичні експерименти, опитувати клієнтів, вимірювати якісь статистичні метрики поведінки користувачів на веб-сайтах тощо).

2.4.2 Кодування не числових ознак

Очевидно, що далеко не всі ознаки об'єктів природно описуються чисельним значенням. Якщо говорити про розміри об'єкта або вартості якогось товару, то ці ознаки, безсумнівно, будуть числовими. Якщо ж мова йде про колір, тип товару (категорії) або взагалі про текстовому описі деякого об'єкту, то подібні ознаки, як правило, надходять неоцифрованими.

Нечислові ознаки з неврегульованими значеннями (в яких між значеннями не визначена дистанція, тобто не можна сказати, що більше або менше) називають категоріальним, або номінальними. Типовий підхід до їх обробки - кодування категоричної ознаки з m можливими значеннями за допомогою m бінарних ознак. Кожна бінарна ознака відповідає одному з можливих значень

категоріального ознаки і є індикатором того, що на даному об'єкті він приймає дане значення. Такий підхід іноді називають one-hot-кодуванням. Наприклад, у нас є три варіанти матеріалу вироби: дерево, пластик, сталь. Причому ми не знаємо напевно, що краще, а що гірше і як це матеріал впливає на підсумкове якість товару. Тоді замість того, щоб закодувати набір матеріалів в один стовпець як значення 1, 2, 3, one-hot-кодування дасть три стовпці і наступні коди: 001, 010, 100. Зауважте, що це не двійкове кодування.

Якщо мова йде не просто про категорії, а про цілі пропозиціях або великих текстах природною мовою, то завдання істотно ускладняється. Кодування тексту побуквенно в більшості випадків нічого не дасть, тому що різноманітність слів і смислів настільки велике, що на такому рівні абстракції модель не побудує потрібну функцію. Обробка і розуміння тексту - це багато в чому напрямок для досліджень.

Проте на сьогоднішній день існує ряд підходів, які дозволяють вирішувати реальні завдання. Перший варіант - поставитися до тексту як до категорій. Припустимо, в текстових даних за все зустрічається 10 тисяч унікальних слів. Тоді кожний окремий текстовий опис (властиве конкретному об'єкту) є така категорія, де зустрічаються відповідні слова, тобто кожен текст буде кодуватися в вектор з 10 тисяч елементів, де на місці відповідних слів-елементів стоятимуть одиниці, а на місці слів, яких немає в даному тексті - нулі. В результаті отримаємо досить розріджену матрицю (що складається в основному з нулів) і при цьому досить велику. Через це виникає проблема її зберігання і обробки. На сьогоднішній день існує ряд технік оптимізації роботи з розрідженими матрицями як на фундаментальному рівні (наприклад, сингулярне розкладання), так і на рівні бібліотек (наприклад, в пакеті scipy).

Однак враховувати всі слова часто буває надмірно і навіть безглуздо. Тому на практиці застосовується підрахунок статистичних характеристик тексту.

2.4.3 Приведення даних до єдиного маштабу і стандартизація

«Сирі» дані мають різний масштаб і різний розподіл за кожною ознакою. Наприклад, якийсь хімічний показник суміші може мати значення в діапазоні від 0.0001 до 0.2, а інший показник від -100 до 100. Або, скажімо, вік клієнтів може бути від 16 до 40, причому набагато більше клієнтів від 18 до 25, іншими словами, математичне очікування зміщене до центру розподілу. Подібні відмінності в ознаках можуть вносити суттєву помилку для множини моделей (наприклад, для регресії, нейронних мереж), і тому потрібно привести всі ознаки до єдиного вигляду.

Існує деяка плутанина в термінах «стандартизація» і «нормалізація». Дуже часто під стандартизацією і нормалізацією розуміються різні речі, а іноді стандартизацію розглядають як частину нормалізації. Тому важливо зрозуміти загальну суть і мету цих методів.

Стандартизація даних - це процес приведення вектора кожної ознаки до такого виду, що його математичне очікування стане нульовим, а дисперсія - одиничною.

Нормалізація даних - це процес масштабування вектора кожної ознаки, тобто приведення його до такого виду, що вектор матиме одиничну норму (при цьому є різні способи оцінки \ підрахунку норми).

Для того щоб досягти однакового масштабу всіх векторів, необхідна нормалізація. Є кілька видів норм і, відповідно, нормалізації.

Найбільш очевидний і простий метод - це max норма. Щоб всі значення лежали в одному діапазоні, потрібно знайти максимальне з можливих значень і всі інші поділити на нього. Таким чином, максимальне значення буде одиницею, а всі інші ляжуть в діапазон від 0 до 1. Але це за умови, що немає негативних значень.

2.4.4 Розмітка даних

Якщо мова йде про те, щоб навчити модель прогнозувати майбутні показники по минулому досвіду (наприклад, прогноз середньої виручки за кількістю відвідувачів магазинів), то такі дані, як правило, приходять з відомої цільової змінної *У* (тобто середньої виручки). Ці дані необхідно буде обробити відповідно до пунктів вище, але їх не буде потрібно додатково розмічати. Однак не рідкість, коли спеціалісту з аналізу даних необхідно буде самостійно розмічати вибірку. Ця потреба може виникнути через відсутність розмічених вихідних даних, таке може бути обумовлене експериментами, що виникають в ході вирішення загальної задачі. Наприклад, для розпізнавання візуальних образів машин або світлофорів потрібно створити спеціальний файл, в якому будуть проставлені асоціації образів на ім'я файлів (тобто 001 - «світлофор»). Це завдання може вирішувати і не фахівець з аналізу даних або машинного навчання, а інші співробітники (наприклад, з боку замовника), але важливо відзначити, що цей етап може також виникнути в ході розробки рішення, і його не можна уникнути в разі навчання з учителем.

Однак варто зазначити, що у випадках навчання і без вчителя може знадобитися часткова розмітка даних з метою тестування і оцінки якості моделі, так як в противному випадку ми просто отримуємо «чорний ящик» без розуміння того, як ми вирішили поставлене завдання.

2.5 Недонавчання та перенавчання моделей

Нами були розглянуті типові завдання з підготовки даних, тепер розглянемо одну з основних проблем алгоритмів машинного навчання - перенавчання. При підготовці даного пункту використовувався матеріал (в тому числі ілюстративний) з джерела [3].

Ми пам'ятаємо, що однією з важливих характеристик алгоритмів машинного навчання є узагальнююча здатність. Однак з нею пов'язані ще два поняття: недонавчання і перенавчання.

Недонавчання виникає при навчанні по прецедентах і характеризується тим, що алгоритм не дає задовільно малої середньої помилки на навчальній множині. Як правило, це явище з’являється внаслідок використання недостатньо складних моделей.

Протилежне цьому явищу - перенавчання. Його суть полягає в тому, що ймовірність помилки натренованого алгоритму на об'єктах тренувальної вибірки виявляється істотно менше, ніж на об'єктах тестової. Найчастіше перенавчання з'являється через використання занадто складних моделей.

Розглянемо графік, який ілюструє ефект перенавчання (рис.2.1).

Точки на графіку з рис.2.1 відповідають різним способам навчання, і кожна з них отримана усередненням на великій кількості розбиття вихідної вибірки об'ємом в 72 зразка на тестову і навчальну частини. Як видно з рисунка, точки мають постійний зсув вгору щодо діагоналі графіка. Тобто спостерігається ефект перенавчання: помилки на тестовій вибірці з'являються частіше, ніж на навчальній.

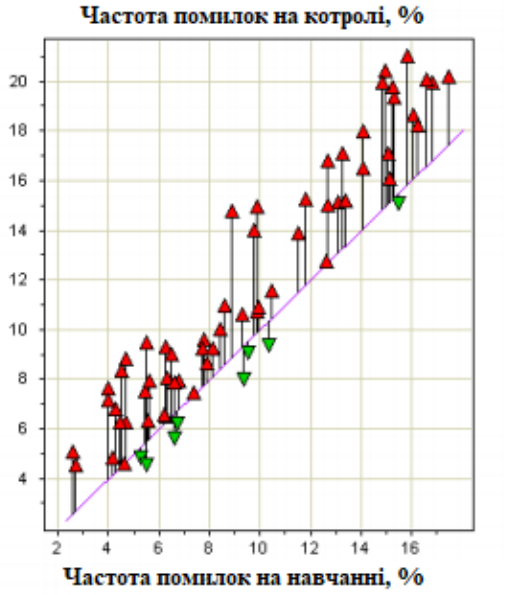

Рис.2.1. Ілюстрація перенавчання

Найчастіше при побудові алгоритмів навчання використовується метод мінімізації емпіричного ризику (середній помилки алгоритму на навчальній вибірці). Його суть полягає в тому, щоб для поточної моделі підібрати алгоритм, що мінімізує значення середньої помилки на даній навчальній вибірці.

З перенавчанням методу мінімізації емпіричного ризику пов'язані три твердження, що пояснюють його причину:

1. Мінімізація емпіричного ризику не є гарантією малої ймовірність помилки на тестових даних. Можна легко побудувати алгоритм, який мінімізує емпіричний ризик до нуля, однак не буде здатний до навчання. Суть полягає в тому, що цей алгоритм запам'ятовує навчальну вибірку, потім порівнює з пропонованим. При збігу пред'явленого об'єкта і зразка навчальної вибірки алгоритм видає правильну відповідь, інакше - виведеться довільний. Тобто емпіричний ризик дорівнює нулю, однак узагальнюючої здатності у алгоритму немає.

2. Відповідно до [3]: «Перенавчання з'являється саме внаслідок мінімізації емпіричного ризику. Нехай задано кінцеву безліч з алгоритмів, які допускають помилки незалежно та з однаковою ймовірністю. Число помилок будь-якого з цих алгоритмів на заданій навчальній вибірці підкоряється тому ж біноміальному розподілу. Мінімум емпіричного ризику - це випадкова величина, що дорівнює мінімуму D з незалежних однаково розподілених біноміальних випадкових величин, очікуване значення якої зменшується з ростом D. Відповідно, з ростом D збільшується перенавчання - різниця ймовірності помилки і частоти помилок на навчанні.

В даному модельному прикладі легко побудувати довірчий інтервал перенавчання, так як функція розподілу мінімуму відома. Однак в реальній ситуації алгоритми мають різні ймовірності помилок, не є незалежними, а безліч алгоритмів, з якого вибирається найкращий, може бути нескінченним. З цих причин висновок кількісних оцінок перенавчання є складним завданням, якої

займається теорія обчислювального навчання. До сих пір залишається відкритою проблема сильної завищеності верхніх оцінок імовірності перенавчання ».

3. Перенавчання з'являється в зв'язку з надмірною складністю моделі. Завжди можна знайти оптимальне значення складності моделі, при якому перенавчання буде мінімальним. Для прикладу наведемо кілька графіків, на яких буде видно залежність перенавчання від складності моделі. При ступені 2 полінома (рис.2.2) модель є недонавченою. При ступені 40 (рис.2.4) - перенавченою і нестійкою, а ось ступінь 20 (рис.2.3) - оптимальна.

Рис.2.2. Недонавчена модель

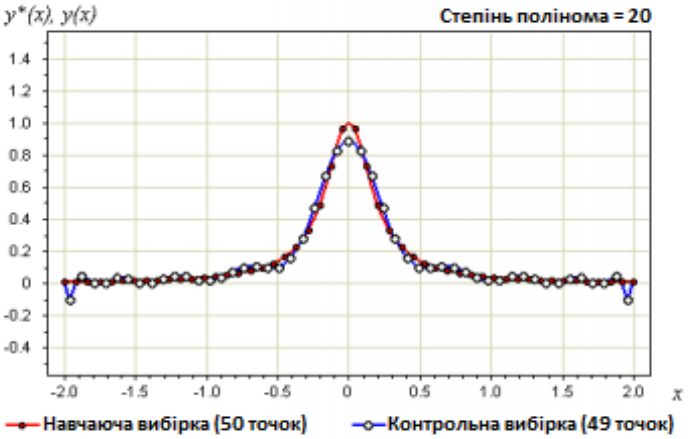

Рис.2.3. Оптимальна модель

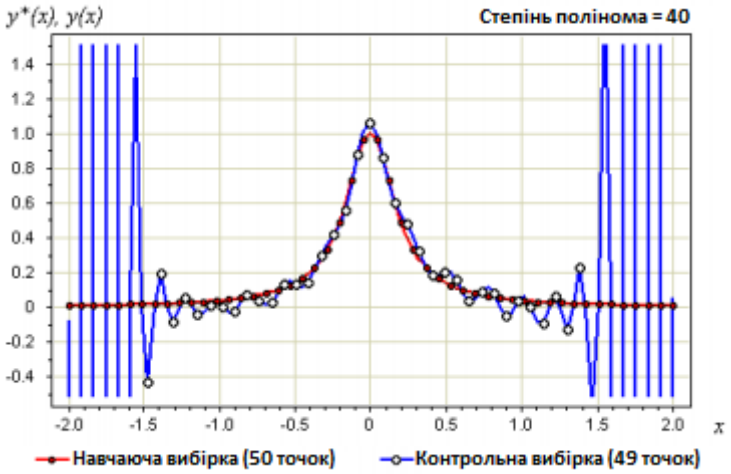

Рис.2.4. Перенавчена модель

Одним із способів вимірювання ймовірності перенавчання є емпіричний метод Монте-Карло, або метод ковзаючого контролю. У літературі також зустрічаються назви крос-перевірка, або крос-валідація. Суть даного методу полягає в наступному: проводиться певна кількість розбиття вихідної вибірки на навчальну і контрольну. Для кожного розбиття відбувається навчання алгоритму на навчальній підвибірки і оцінка середньої помилки на контрольній. Потім обчислюється оцінка змінного середнього, як середня по всім розбиття величина обчисленої помилки. Для незалежної вибірки цей показник дає несмещенную оцінку ймовірності помилки. Метод змінного контролю є стандартним способом порівняння і оцінювання алгоритмів класифікації, регресії, прогнозування.

При використанні крос-валідації можна зробити наступні висновки:

1. Якщо помилка велика на більшості ділянок, то швидше за все проблема в моделі.

2. Якщо дані навчальної вибірки характеризуються сильно зміщеними математичним очікуванням і дисперсією, то рівень узагальнення буде низьким, що може бути пов'язано з перенавчанням.

3. Якщо знайдені сильні відхилення на певних підвибірках, то, ймовірно, проблема в цих ділянках даних або модель недонавчена.

4. При малому обсязі навчальної вибірки крос-валідація може стати способом боротьби з перенавчанням.

Якщо ж говорити про прямі способи боротьби з перенавчанням, то можна виділити наступні:

1. Спрощення моделі.

2. Підготовка більшого числа навчальних даних (можливо, за допомогою генерації).

3. Регуляризація.

Регуляризація полягає в додаванні деякої додаткової інформації для умови мінімізації помилки. Виконується це, щоб вирішити некоректно поставлене завдання або запобігти перенавчання. Найчастіше така інформація набуває вигляду штрафу за складність моделі. Наприклад, введені обмеження по нормі векторного простору або гладкості результуючої функції. Або ж, з байєсівської точки зору, додані апріорні розподілу на параметри моделі.

Згідно [4] існують такі основні види регуляризації:

1. L1 регуляризація (англ. Lasso regression):

$$RSS(w) = \frac{1}{2}\sum_{i=1}^{N}\left(f(x_{i,}w) - y_i\right)^2 + \lambda\sum_{j=0}^{M}\left|w_j\right|, \tag{2.14}$$

де w – вектор ваг поліному;

λ – коефіцієнт регуляризації.

2. L2 регуляризація, або регуляризація Тихонова для інтегральних рівнянь дозволяє балансувати між відповідністю даними і малою нормою рішення:

$$RSS(w) = \frac{1}{2}\sum_{i=1}^{N}\left(f(x_{i,}w) - y_i\right)^2 + \frac{\lambda}{2}\sum_{j=0}^{M}\left|w_j\right|^2, \tag{2.15}$$

де $\left|w_j\right|^2$ – квадратична норма вектору ваг.

Іншими словами, перенавчання в більшості випадків виявляється в тому, що в які утворюються многочленах занадто великі коефіцієнти. Відповідно і

боротися з цим можна досить природним способом: потрібно просто додати в цільову функцію штраф, який би карав модель за надто великі коефіцієнти ».

З приводу застосування тієї чи іншої регуляризації наведемо матеріал джерела [5]: «регуляризації можна застосовувати з будь-яким методом МО-класифікації, який заснований на математичному рівнянні. Приклади включають лінійну, логістичну регресію і нейронні мережі. Оскільки це зменшує величину вагових значень в моделі, регуляризацію іноді називають скороченням ваг. Основна перевага застосування регуляризації в тому, що вона часто призводить до створення більш точної моделі. Головний недолік полягає у введенні додаткового параметра, значення якого потрібно визначити, - вагового значення регуляризації. У разі логістичної регресії це не дуже серйозно, так як в цьому алгоритм зазвичай використовується лише параметр швидкості навчання, але при використанні більш складного методу класифікації, зокрема нейронних мереж, додавання ще одного так званого гіперпараметра може принести масу додаткової роботи для підбору комбінації значень двох параметрів.

L1 та L2-регуляризації - процеси схожі. Дослідниками сформульовані деякі теоретичні правила щодо того, яка регуляризація краще для певних завдань, але на практиці доведеться проекспериментувати, щоб визначити, який тип регуляризації краще у конкретному випадку і чи варто взагалі використовувати будь-яку регуляризацію.

Застосування L1-регуляризації іноді може давати корисний побічний ефект, що викликає прагнення одного або більше вагових значень до 0.0, а це означає, що відповідний ознака не надає значного впливу на результуючий, тобто включення його в модель потрібно. Це одна з форм того, що називають «селекцією ознак». На відміну від L2-регуляризація обмежує вагові значення моделі, але зазвичай не призводить до повного обнулення цих значень. Тому може здатися, що L1-регуляризація краще L2-регуляризації. Однак недолік застосування L1-регуляризації в тому, що цей метод не так-то просто використовувати з деякими алгоритмами машинного навчання. Наприклад, з

тими, в яких використовуються чисельні методи для обчислення так званого градієнта. L2-регуляризацію можна використовувати з будь-яким типом алгоритму навчання.

2.6 Моделі та алгоритми машинного навчання

2.6.1 Методи теорії ймовірностей

Перший зріз методів, який необхідно розглянути, - методи теорії ймовірностей. Теорія ймовірностей - розділ математики, в якому вивчаються випадкові величини і події, їх властивості та можливі операції над ними. Таким чином, ключове поняття, що лежить в основі цієї дисципліни, - ймовірність події P_i . Спираючись на нього, можна дати визначення повної групи подій. Вона визначається як система випадкових подій, яка має такі властивості:

1. В результаті випадкового експерименту неодмінно матиме місце одне і тільки одне зі складових її подій.

2. Сума ймовірностей всіх подій повної групи дорівнює 1.

За допомогою методів теорії ймовірностей можна проводити як прості, так і складні операції з аналізу даних. Серед простих можна виділити підрахунки імовірнісних характеристик вибірки: медіани, математичного очікування, дисперсії, а також величини середньоквадратичного відхилення.

Однак розглянуті показники характеризують випадкову величину лише з якоїсь однієї сторони. Найбільш же повно і вичерпно її описує закон розподілу - функція, що визначає для вибірки *X* ймовірність попадання в певний інтервал або ймовірність отримання певного значення Якщо який-небудь закон розподілу описує випадкову величину, то кажуть, що вона йому підпорядковується або по ньому розподілена. Іншими словами, закон розподілу описує область значень випадкової величини і ймовірності їх отримання.

Серед законів розподілу найбільш часто використовується закон нормального розподілу або розподілу Гаусса, який характеризує більшість процесів, що зустрічаються в світі. Звідси і пішла його назва (нормальний). Закон (щільність розподілу ймовірності) описується наступною формулою:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.16)$$

Для аналізу даних дуже часто використовується одна з основних теорем теорії ймовірностей - теорема Байеса.

Вона дозволяє визначити ймовірність настання якої-небудь події за умови, що відбулася інша подія, статистично взаємопов'язане з досліджуваним. Ключовими поняттями в даній теоремі є поняття трансцендентальної і апостеріорної вірогідності. Розглянемо їх докладніше.

Апріорна ймовірність - призначена події ймовірність, за умови відсутності знань, які підтримують його наступ. Апостеріорна ймовірність - призначена події ймовірність за умови наявності знань, що підтримують його наступ і отриманих дослідним шляхом.

2.6.2 Дерево рішень

Дерево рішень - засіб підтримки прийняття рішень для прогнозних моделей. Суть його роботи полягає в послідовному розбитті безлічі даних на непересічні класи, які в свою чергу також піддаються розбиттю по будь-яким критеріям з оцінкою ефективності розбиття. Як правило, дерево рішень складається з «вузлів», «листя» і «гілок». «Гілки» містять записи атрибутів, від яких залежить цільова функція, «листя» - значення цільової функції, а «вузли» - інші атрибути, за якими відбувається класифікація. Найчастіше виділяють два типи дерев: для класифікації (в цьому випадку що передбачається результат - клас, якому належать дані) і для регресії (результат - прогнозоване значення цільової функції).

Узагальнений алгоритм побудови дерева рішень по навчальній вибірці складається з наступних кроків:

1. Беремо наступний атрибут і поміщаємо його в корінь.

2. Для всіх значень цього атрибута - залишаємо в «листі» даної «гілки» тільки ті значення, які відповідають певній умові.

3. Продовжуємо будувати дерево серед залишених на попередньому кроці «листя».

Для вибору наступного атрибута може бути використаний один з наступних основних алгоритмів:

1. ID3. Атрибут вибирається на основі приросту інформації та мінімізації ентропії. Нагадаємо, що під ентропією в даному випадку мається на увазі міра невпорядкованості системи. Чим менше її величина, тим більше впорядковані складові системи.

2. C4.5 - поліпшена версія попереднього алгоритму, в якій використовується нормалізований приріст інформації.

3. CART - алгоритм, який використовується для побудови бінарних дерев, що виробляє розбиття на основі модальних значень ознак.

Розглянемо роботу по побудові дерева рішень з використанням першого алгоритму.

Коротко алгоритм ID3 може бути описаний трьома кроками:

1. Порахувати ентропію множини, що розбивається.

2. Вибрати критерій з мінімальною ентропією і, відповідно, максимальною інформаційної вигодою.

3. На основі отриманого критерію створити вузол дерева і повторити процедуру.

Варто відзначити, що частіше за все дерева рішень - бінарні. У вузлах вони містять умови, а гілки відповідають істинність або хибність цієї умови.

2.6.3 Статистичні моделі та методи

Було розглянуто методи машинного навчання, які надаються теорією вірогідності, а також модель дерева рішень. Тепер звернемося до статистики і коротко опишемо її моделі та методи, які можуть бути використані для вирішення завдань машинного навчання, а саме методи регресійного аналізу.

В контексті машинного навчання під регресійний аналізом розуміється процес побудови математичної моделі, яка описує залежність деякої цільової характеристики об'єкта або процесу від інших його характеристик. Наприклад, залежність числа нових клієнтів від величини зарплати працюючого на вулиці промоутера.

У задачі регресійного аналізу завжди є навчальна вибірка, що складається з вхідних параметрів і відгуків, а також початкова параметрична модель, в найпростішому випадку - лінійна, проте це не обов'язково.

Після отримання адекватної моделі ми можемо вирішувати проблему прогнозування, підставляючи в отриману формулу величину x і обчислюючи величину y (із задовольняючою нас похибкою). Наведений вище приклад моделі - модель парної лінійної регресії, але крім неї існує множинна та нелінійна регресії.

Припустимо, ми хочемо описати залежність між двома факторами моделлю виду $y = \beta_0 + \beta_1 \times x$. Перше, що необхідно врахувати, - побудована лінія ніколи не буде точно проходити по дослідженим точкам, тому істинний вид регресійній моделі буде $y = \beta_0 + \beta_1 \times x + e$, де e - помилки спостережень. Друге - перш ніж переходити до оцінювання параметрів моделі, доцільно побудувати діаграму розсіювання, щоб переконатися, що обрана модель дійсно може описати залежність між факторами. На діаграмі розсіювання кожній парі «залежний-впливаючи параметр» відповідає точка на площині. Як правило, залежний фактор відкладається по осі ординат, а другий - по осі абсцис. Модель парної лінійної регресії графічно являє собою лінію, отже, використовувати її

для опису залежності доцільно, якщо на діаграмі розсіювання точки розташовуються навколо (і досить близько) будь-якої прямої. Відхилення реальних точок від модельних - залишки або помилки спостережень, які позначаються *e*. Приклад діаграми розсіювання показаний на рис.2.5.

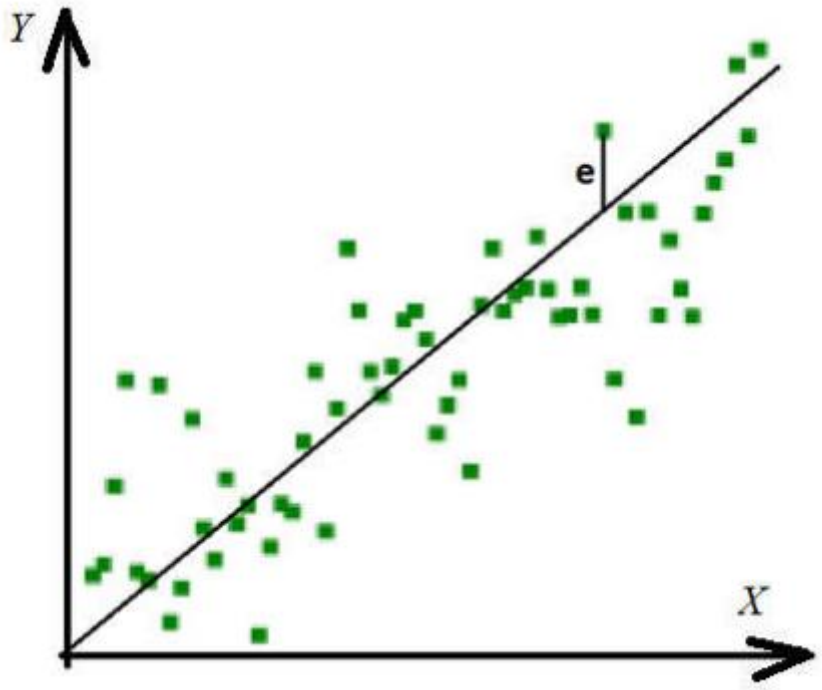

Рис.2.5. Діаграма розсіювання

Якщо після аналізу діаграми розсіювання приймається рішення використовувати лінійну парну регресію для моделювання залежності, то наступним кроком буде знаходження параметрів моделі β_0 та β_1. Для вирішення цього завдання в дев'яносто дев'яти відсотках випадків використовується метод найменших квадратів, запропонований Гауссом більше двохсот років тому. Суть даного методу полягає в мінімізації суми квадратів відхилень досвідчених даних від модельних. Формально ця задача описується так:

$$Q = \sum e_i^2 \to min \tag{2.17}$$

В даному випадку e_i розраховується як:

$$e_i = \beta_0 + \beta_1 * x_i - y_i \ ,$$

де y_i – реальне вихідне значення для вхідного значення x_i.

Висновки до розділу 2

В результаті виконання даного розділу, було розглянуто основні задачі машинного навчання, а саме типи задач машинного навчання, способи навчання та оцінки його якості, типові задачі при підготовці даних для навчання моделі. Також, в ході опрацювання даного розділу, було проведено аналіз моделей та основних алгоритмів машинного навчання, зокрема методи теорії ймовірностей, дерева рішень та статистичні моделі та методи. Розкрито важливі характеристики алгоритмів машинного навчання, а саме недонавчання та перенавчання та наведено способи та методи боротьби з ними.

У наступному розділі буде проведено побудову системи управління на базі машинного навчання, на прикладі моделі прогнозування майбутніх погодних показників.

3. РОЗРОБКА МОДЕЛІ ПРОГНОЗУВАННЯ ПОГОДНИХ ПОКАЗНИКІВ

3.1 Вихідні дані

Для отримання вихідних даних було використано API веб – сервісу Weather Underground. Після отримання дані мають бути опрацьовані та об’єднані у формат, що підходить для аналізу даних, і згодом очищені. Weather Underground являє собою компанію, що займається збором та агрегацією різноманітних погодних даних, що надходять з датчиків мереж IoT.

Компанія надає безліч API-інтерфейсів, які доступні як для комерційного, так і для некомерційного використання. Для нас є цікавим інтерфейс історії, що являє собою зведення погодних вимірювань за визначений день.

Для запитів до API історії Weather Underground та наступної обробки отриманих даних використовується кілька стандартних бібліотек, а також деякі популярні сторонні (табл. 3.1).

Таблиця 3.1. Використані бібліотеки

Назва	Опис	Джерело
datetime	Використовується для роботи з датою	Стандартна бібліотека
time	Використовується для роботи з часом	Стандартна бібліотека
collections	Використовується для структурування даних	Стандартна бібліотека
pandas	Використовується для обробки, організації та очистки даних	Стороння бібліотека

requests	Використовується для генерації запитів до API	Стороння бібліотека
matplotlib	Використовується для графічного аналізу	Стороння бібліотека

В результаті підготовки описаних вище бібліотек, отримуємо:

```
from datetime import datetime, timedelta
import time
from collections import namedtuple
import pandas as pd
import requests
import matplotlib.pyplot as plt
```

Рис.3.1. Вихідний код налаштування бібліотек

Наступним кроком є встановлення необхідної дати для отримання показників(рис.3.2). Нас цікавлять наступні показники:

- Середня температура;
- Середня точка роси;
- Середній тиск;
- Максимальна вологість;
- Мінімальна вологість;
- Максимальна точка роси;
- Мінімальна точка роси;
- Максимальний тиск;
- Мінімальний тиск;
- Атмосферні опади;

```
target_date = datetime(2019, 5, 1)
features = ["date", "meantempm", "meandewptm", "meanpressurem", "maxhumidity", "minhumidity", "maxtempm",
            "mintempm", "maxdewptm", "mindewptm", "maxpressurem", "minpressurem", "precipm"]
DailySummary = namedtuple("DailySummary", features)
```

Рис.3.2. Вихідний код встановлення цільової дати

3.2 Підготовка вихідних даних

Наступним кроком, як було описано вище, є фільтрація даних. Для забезпечення якості даних необхідно визначити непотрібні дані, пропущені значення, узгодженість типів даних. Перше, що необхідно зробити, це видалити стовбці DataFrame для зменшення об’єму даних. Оскільки ціллю завдання є прогнозування майбутньої температури на основі останніх трьох днів вимірювань погоди, то є доцільним зберігати мінімальну, максимальну і середню температури для кожного дня, та всі наступні похідні змінні, що будуть додані в ході виконання завдання.

Таблиця 3.2 Вихідні дані

	Кількість	Значення	min	25%	50%	75%	max
maxhumidity_1	999	88.107107	47.0	83.0	90.0	93.0	100.0
maxhumidity_2	998	88.102204	47.0	83.0	90.0	93.0	100.0
maxhumidity_3	997	88.093280	47.0	83.0	90.0	93.0	100.0
maxpressurem_1	999	1019.924925	993.0	1015.0	1019.0	1024.0	1055.00
maxpressurem_2	998	1019.922846	993.0	1015.0	1019.0	1024.0	1055.00
maxpressurem_3	997	1019.927783	993.0	1015.0	1019.0	1024.0	1055.00
minpressurem_1	999	1012.329329	956.0	1008.0	1012.0	1017.0	1035.0
minpressurem_2	998	1012.326653	956.0	1008.0	1012.0	1017.0	1035.0
minpressurem_3	997	1012.326981	956.0	1008.0	1012.0	1017.0	1035.0
precipm_1	889	2.908211	0.0	0.0	0.0	0.51	95.76
precimp_2	889	2.908211	0.0	0.0	0.0	0.51	95.76
precimp_3	888	2.888885	0.0	0.0	0.0	0.51	95.76

Оцінка потенційного впливу викидів є складною задачею будь-якого аналітичного проекту. З одної сторони, нам необхідно брати до уваги можливість появи хибних артефактів даних, що можуть істотно вплинути на наші моделі або змістити їх. З іншої сторони, викиди можуть бути надзвичайно значущими в прогнозуванні результатів, що виникають в особливих обставинах.

Перший набір даних зв’язаний з максимальною вологістю. Дивлячись на дані, можна зробити висновок, що викид для даної категорії об’єктів

обумовлений дуже низьким мінімальним значенням. Для наглядної демонстрації цього, побудуємо гістограму (рис.3.3) відповідно з даними.

```
%matplotlib inline
plt.rcParams['figure.figsize'] = [14, 8]
df.maxhumidity_1.hist()
plt.title('Distribution of maxhumidity_1')
plt.xlabel('maxhumidity_1')
plt.show()
```

Рис.3.3. Вихідний код побудови гістограми

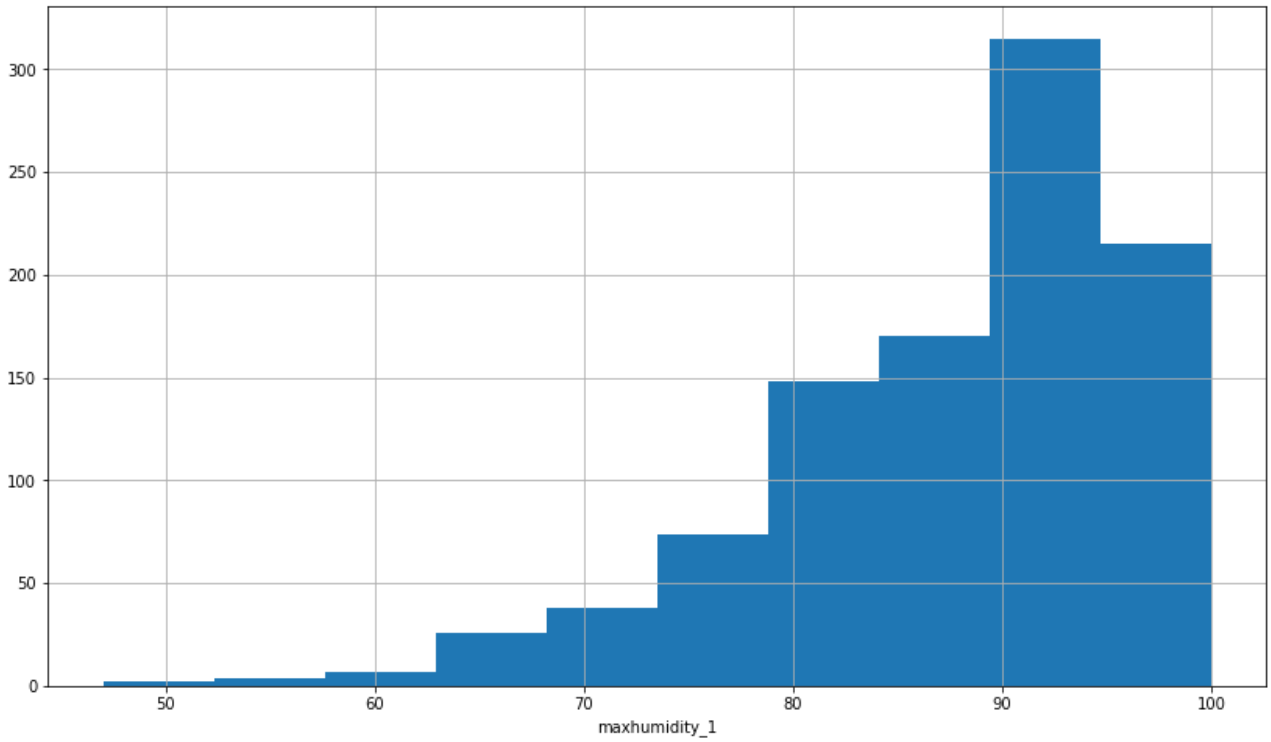

Рис.3.4. Гістограма даних максимальної вологості

При дослідження отриманої гістограми значень є помітним цілком негативний перекіс. Це є необхідним для врахування при виборі моделей прогнозування і оцінки сили взаємодії максимальної вологості. Багато із базових статистичних методів припускають, що дані зазичай розподіляються.

Далі переглянемо розподіл мінімального тиску:

```
df.minpressurem_1.hist()
plt.title('Distribution of minpressurem_1')
plt.xlabel('minpressurem_1')
plt.show()
```

Рис.3.5. Вихідний код побудови гістограми

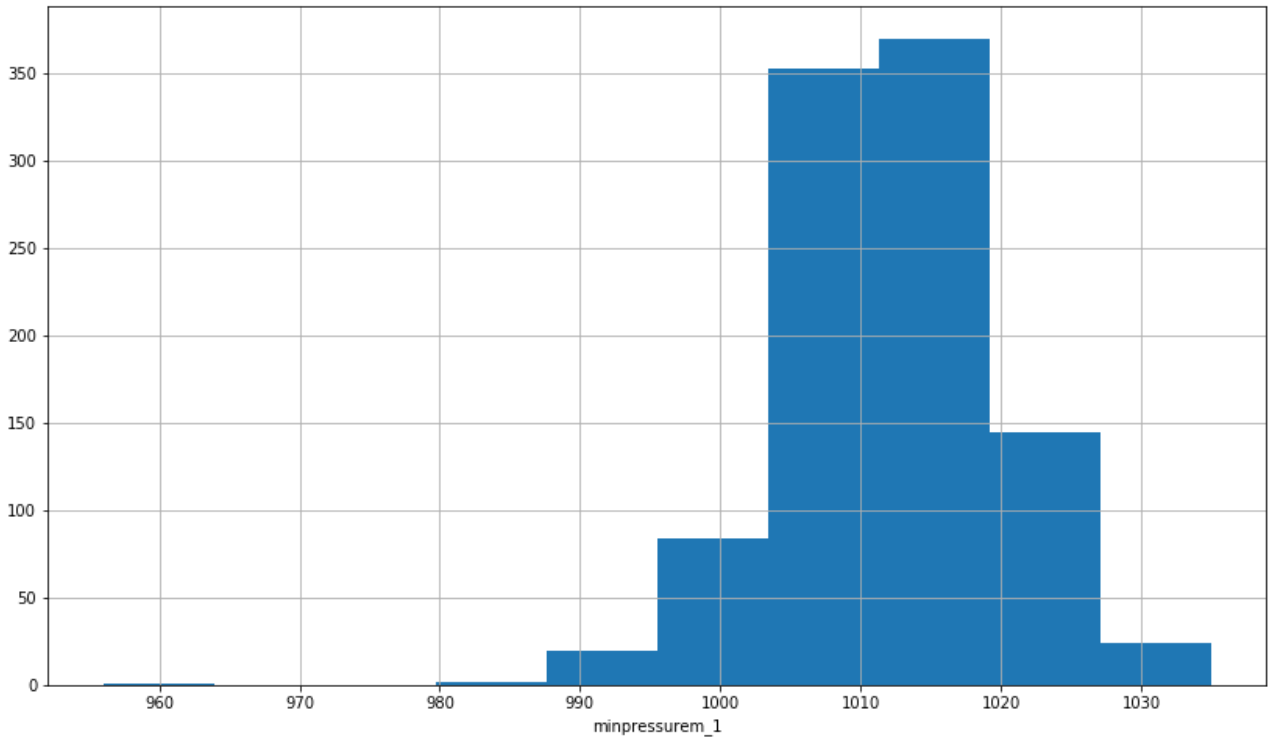

Рис.3.6. Гістограма даних мінімального тиску

Останньою проблемою якості даних, яку необхідно вирішити, є проблема пропусків значень. Відсутність даних створює проблему, оскільки для більшості методів машинного навчання потрібні повні набори даних. Для вирішення цієї проблеми є два основні способи:

1. Видалити рядки, що мають пропущені значення;
2. Заповнити пропущені значення інтерпольованим значенням, що є розумною оцінкою значень;

Оскільки є необхідним збереження як можна більшої частини даних, де має місце мінімальний ризик помилкових значень, можна заповнити дані значення значенням 0:

```
for precip_col in ['precipm_1', 'precipm_2', 'precipm_3']:
    # create a boolean array of values representing nans
    missing_vals = pd.isnull(df[precip_col])
    df[precip_col][missing_vals] = 0
```

Рис.3.7. Заповнення даних циклом

Лінійна регресія спрямована на застосування набору вихідних припущень, що стосуються лінійних відносин і чисельних методів, для прогнозування результату (*Y*, або залежною змінною) на основі одного або декількох предикторів (незалежних змінних *X*) з кінцевою метою створення моделі (математичної), формули для прогнозування результатів з урахуванням тільки значень предиктору з деяким ступенем невизначеності.

Узагальнена формула для моделі лінійної регресії:

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \cdots + \beta_{(p-n)} x_{(p-n)} + E \ ,$$

де $\hat{y}$ – змінна прогнозуючого результату (залежна змінна);

x_j – змінні предиктору (незалежні змінні) для параметрів 1,2,3..p-1;

β_0 – значення $\hat{y}$ коли кожний x_j дорівнює нулю;

β_j – зміна $\hat{y}$ на основі однієї зміни одиниці в одному із відповідних x_j;

E – випадкова помилка, що пов’язана із різницею прогнозуючого значення та фактичного.

3.3 Побудова моделі лінійної регресії

Ключове припущення, що необхідне методом лінійної регресії, полягає в тому, що між залежною змінною і кожною незалежною змінною є лінійна залежність. Одним із способів оцінки лінійності між нашою незалежною змінною, яка поки буде являти собою середню температуру, і іншими незалежними змінними є обчислення коефіцієнта кореляції Пірсона.

Коефіцієнт кореляції Пірсона (r) являє собою вимір величини лінійної кореляції між масивами рівної довжини, яка приймає значення в діапазоні від -1 до 1. Значення кореляції в діапазоні від 0 до 1 представляють як позитивну кореляцію. Під цим слід розуміти, що два ряди даних позитивно корелюють, коли значення в одному ряду даних збільшуються одночасно зі значеннями в іншому ряду, тобто значення кореляції Пірсона буде наближатися до 1.

Кажуть, що значення кореляції від 0 до -1 є негативно корельовані в тому сенсі, що коли значення одного ряду збільшується, відповідні значення в протилежному ряду зменшуються, але в міру того, як зміни величини між рядами стають рівними (з протилежним напрямком) значення кореляції наблизиться до -1.

Думки варіюються серед статистиків і статистичних книг за чіткими границями для рівнів коефіцієнта кореляції. Однак, загальноприйнятим набором класифікацій кореляції є наступний:

Таблиця 3.3 Значення коефіцієнту кореляції

Значення	Інтерпритація
0.8 – 1.0	Дуже сильна
0.6 – 0.8	Сильна
0.4 – 0.6	Середня
0.2 – 0.4	Слабка
0.0 – 0.2	Дуже слабка

Щоб оцінити кореляцію в цих даних, створемо corr () метод об'єкта Pandas DataFrame. Тут виведемо значення кореляції від найбільш негативно корельованих до найбільш позитивно корельованих.

```
df.corr()[['meantempm']].sort_values('meantempm')
```

Рис.3.7. Отримання значень кореляції

Таблиця 3.4 Значення кореляції для вихідних даних

maxpressurem_1	-0,519699
maxpressurem_2	-0,425666
maxpressurem_3	-0,408902
meanpressurem_1	-0,365682
meanpressurem_2	-0,269896
meanpressurem_3	-0,263008
minpressurem_1	-0,201003
minhumidity_1	-0,148602

minhumidity_2	-0,143211
minhumidity_3	-0,118564
minpressurem_2	-0,104455
minpressurem_3	-0,102955
precipm_2	0.084394
precipm_1	0.086617
precipm_3	0.098684
maxhumidity_1	0.132466
maxhumidity_2	0.151358
maxhumidity_3	0.167035
maxdewptm_3	0.829230
maxtempm_3	0.832974
mindewptm_3	0.833546
meandewptm_3	0.834251
mintempm_3	0.836340
maxdewptm_2	0.839893
meandewptm_2	0.848907
mindewptm_2	0.852760
mintempm_2	0.854320
meantempm_3	0.855662
maxtempm_2	0.863906
meantempm_2	0.881221
maxdewptm_1	0.887235
meandewptm_1	0.896681
mindewptm_1	0.899000
mintempm_1	0.905423
maxtempm_1	0.923787
meantempm_1	0.937563
mintempm	0.973122
maxtempm	0.976328
meantempm	1.000000

При виборі об'єктів для включення в цю модель лінійної регресії будемо видаляти об'єкти, значення кореляції яких менше абсолютного значення 0,6. Крім того, оскільки змінні «mintempm» і «maxtempm» відносяться до того ж дня, що і змінна прогнозування «meantempm», їх також будемо видаляти.

```
predictors = ['meantempm_1',  'meantempm_2',  'meantempm_3',
              'mintempm_1',   'mintempm_2',   'mintempm_3',
              'meandewptm_1', 'meandewptm_2', 'meandewptm_3',
              'maxdewptm_1',  'maxdewptm_2',  'maxdewptm_3',
              'mindewptm_1',  'mindewptm_2',  'mindewptm_3',
              'maxtempm_1',   'maxtempm_2',   'maxtempm_3']
df2 = df[['meantempm'] + predictors]
```

Рис.3.8. Вихідні дані для побудови графіків

Для того щоб наглядно продемонструвати наявність лінійного зв’язку, складемо графіки для кожного із вибраних предикторів, використовуючи модуль pyplot від matplotlib.

Для цього графіка залежна змінна "meantempm" буде послідовною віссю Y по всім 18 графікам змінних предиктору. Один із способів зробити це - створити сітку графіків. Pandas поставляється з корисною функцією побудови графіків, яка називається scatter_plot (), але її зазвичай використовують тільки тоді, коли існує всього близько 5 змінних, тому що вона перетворює графік в матрицю N x N (в нашому випадку 18 x 18), що ускладнює аналіз даних. Замість цього можна створити сіткову структуру з шістьма рядками з трьох стовпців, щоб не спотворювати чіткість графіків.

```
%matplotlib inline

# manually set the parameters of the figure to and appropriate size
plt.rcParams['figure.figsize'] = [16, 22]

# call subplots specifying the grid structure we desire and that
# the y axes should be shared
fig, axes = plt.subplots(nrows=6, ncols=3, sharey=True)

# Since it would be nice to loop through the features in to build this plot
# let us rearrange our data into a 2D array of 6 rows and 3 columns
arr = np.array(predictors).reshape(6, 3)

# use enumerate to loop over the arr 2D array of rows and columns
# and create scatter plots of each meantempm vs each feature
for row, col_arr in enumerate(arr):
    for col, feature in enumerate(col_arr):
        axes[row, col].scatter(df2[feature], df2['meantempm'])
        if col == 0:
            axes[row, col].set(xlabel=feature, ylabel='meantempm')
        else:
            axes[row, col].set(xlabel=feature)
plt.show()
```

Рис.3.9. Побудова графіків для кожного із предикторів

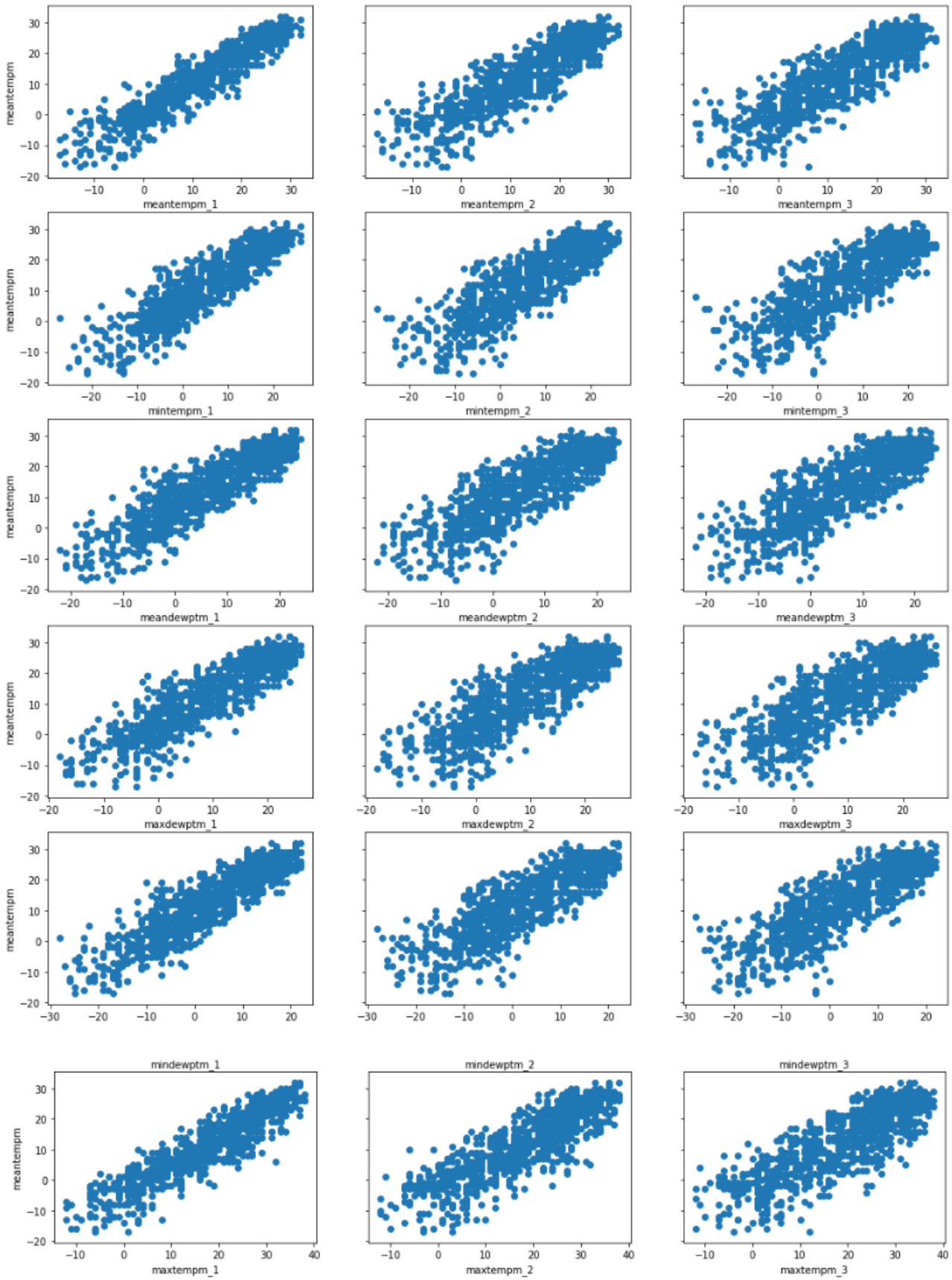

Рис.3.10. Отримані графіки для кожного з вибраних предикторів

З наведених вище графіків (рис.3.10) видно, що всі прогнозуючі змінні демонструють хороший лінійний зв'язок зі змінною відгуку ("meantempm"). Крім того, варто відзначити, що всі відношення виглядають рівномірно розподілені. Під цим слід розуміти, що існує відносно рівномірний розподіл значень, позбавлених будь-якої форми роздування або конуса. Рівномірний випадковий розподіл розкиду по точкам також є іншим важливим припущенням про лінійної регресії з використанням алгоритму звичайних найменших квадратів.

Надійна модель лінійної регресії повинна використовувати статистичні тести для вибору значущих, статистично значущих предикторів для включення. Щоб вибрати статистично значущі функції, будемо використовувати statsmodels бібліотеку Python. Ключовим аспектом використання статистичних методів, таких як лінійна регресія, в аналітичному проекті є створення і перевірка тестів гіпотез для перевірки значущості припущень, зроблених щодо досліджуваних даних. Існує безліч тестів гіпотез, які були розроблені для перевірки стійкості моделі лінійної регресії по відношенню до різних зроблених припущень. Одним з таких тестів гіпотез є оцінка значущості кожної з включених змінних предикторів.

Формальне визначення перевірки гіпотези для значущості параметрів полягає в наступному:

- H_0: нульова гіпотеза стверджує, що предиктор не впливає на значення вихідної змінної $\beta_j = 0$;
- H_a: Альтернативна гіпотеза полягає в тому, що провісник надає істотний вплив на значення змінної результату в $\beta_j \neq 0$;

У багатьох наборах даних можуть існувати взаємодії між змінними, які можуть привести до помилкових інтерпретацій цих простих перевірок гіпотез. Щоб перевірити вплив взаємодій на значимість якої-небудь однієї змінної в моделі лінійної регресії, часто застосовується метод, відомий як покрокова регресія. Використовуючи покрокову регресію, ви додаєте або видаляєте змінні

з моделі і оцінюєте статистичну значущість кожної змінної в результуючій моделі. У даному завданні використаємо техніку, що має назву зворотнє виключення. Дана техніка працює наступним чином:

1. Вибирається рівень значущості, для якого провіряється власна гіпотеза, для того, щоб визначити чи має змінна залишатись в моделі;
2. Підганяється модель зі всіма змінними предиктору;
3. Оцінюються p-значення коефіцієнтів, і для значення з найбільшим значенням переходимо до наступного пункту;
4. Видаляємо предиктор, вказаний в пункті 3;
5. Встановлюємо модель знову, але на цей раз без видалення змінної і повертаємось на крок 3.

Побудуємо дану повністю завантажену узагальнену модель, використовуючи statsmodels:

```
# import the relevant module
import statsmodels.api as sm

# separate our my predictor variables (X) from my outcome variable y
X = df2[predictors]
y = df2['meantempm']

# Add a constant to the predictor variable set to represent the Bo intercept
X = sm.add_constant(X)
X.ix[:5, :5]
```

Рис.3.11. Побудова моделі

Таблиця 3.5 Вихідні значення моделі

	Const	meantempm_1	meantempm_2	meantempm_3	mintempm_1
2019-01-01	1,0	-4,0	-6,0	-6,0	-13,0
2019-01-02	1,0	-14,0	-4,0	-6,0	-18,0
2019-01-03	1,0	-9,0	-14,0	-4,0	-14,0
2019-01-04	1,0	-10,0	-9,0	-14,0	-14,0
2019-01-05	1,0	-16,0	-10,0	-9,0	-19,0

```
# (1) select a significance value
alpha = 0.05

# (2) Fit the model
model = sm.OLS(y, X).fit()

# (3) evaluate the coefficients' p-values
model.summary()
```

Рис.3.12. Отримання значень регресії OLS

При виклику summary () будуть отримані наступні дані Jupyter:

Таблиця 3.6 Результати регресії OLS

R - квадрат	0.895
Відрегульований R - квадрат	0.893
F – статистика	462.7
AIC	5396

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.0769	0.526	2.049	0.041	0.046	2.108
meantempm_1	0.1047	0.287	0.364	0.716	-0.459	0.669
meantempm_2	0.3512	0.287	1.225	0.221	-0.211	0.914
meantempm_3	-0.1084	0.286	-0.379	0.705	-0.669	0.453
mintempm_1	0.0805	0.149	0.539	0.590	-0.213	0.373
mintempm_2	-0.2371	0.149	-1.587	0.113	-0.530	0.056
mintempm_3	0.1521	0.148	1.028	0.304	-0.138	0.443
meandewptm_1	-0.0418	0.138	-0.304	0.761	-0.312	0.228
meandewptm_2	-0.0121	0.138	-0.088	0.930	-0.282	0.258
meandewptm_3	-0.0060	0.137	-0.044	0.965	-0.275	0.263
maxdewptm_1	-0.1592	0.091	-1.756	0.079	-0.337	0.019
maxdewptm_2	-0.0113	0.091	-0.125	0.900	-0.189	0.166
maxdewptm_3	0.1326	0.089	1.492	0.136	-0.042	0.307
mindewptm_1	0.3638	0.084	4.346	0.000	0.200	0.528
mindewptm_2	-0.0119	0.088	-0.136	0.892	-0.184	0.160
mindewptm_3	-0.0239	0.086	-0.279	0.780	-0.192	0.144
maxtempm_1	0.5042	0.147	3.438	0.001	0.216	0.792
maxtempm_2	-0.2154	0.147	-1.464	0.143	-0.504	0.073

maxtempm_3	0.0809	0.146	0.555	0.579	-0.205	0.367

Тут ми маємо:

1. P>|t| - значення, що було описано вище та використовується для оцінки тесту гіпотези, а саме є критерієм доцільності виключення змінної у покроковій техніці зворотнього видалення;
2. R – квадрат – міра, що показує, скільки від загального відхилення в результаті наша модель може пояснити;
3. Відрегульований R – квадрат – таж міра, що і R – квадрат, але для множинної лінійної регресії.

Тепер, коли ми пройшли етапи вибору статистично значущих предикторів (функцій), ми можемо використовувати SciKit-Learn для створення моделі прогнозування та перевірки її здатності прогнозувати середню температуру. SciKit-Learn - популярна бібліотека машинного навчання, яка широко використовується як в промисловості, так і в наукових колах. Одна річ, яка дуже вражає в SciKit-Learn, це те, що вона підтримує API «підгонки», «передбачення» і «тестування» у багатьох чисельних методах і алгоритмах, що робить її використання дуже простим. На додаток до цього узгодженим дизайну API SciKit-Learn також поставляється з декількома корисними інструментами для обробки даних, загальними для багатьох проектів машинного навчання.

Ми почнемо з використання SciKit-Learn, щоб розділити наш набір даних на набори для тестування і навчання шляхом імпорту train_test_split () функції з sklearn.model_selection модуля. Розділимо набори даних навчання і тестування на 80% навчання і 20% тестування і призначимо random_state 12.

```
X = X.drop('const', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)
```

Рис.3.13 Розділення даних на набори для тестування

Наступна дія - побудова регресійної моделі з використанням набору даних для навчання. Для цього імпортуємо LinearRegression класс з sklearn.linear_model модуля. Як згадувалося раніше, Scikit-Learn впроваджує загальний fit () і predict

() API-інтерфейс в своїх чисельних методах, що робить використання бібліотеки дуже зручним для користувача.

```
# instantiate the regressor class
regressor = LinearRegression()

# fit the build the model by fitting the regressor to the training data
regressor.fit(X_train, y_train)

# make a prediction set using the test set
prediction = regressor.predict(X_test)

# Evaluate the prediction accuracy of the model
from sklearn.metrics import mean_absolute_error, median_absolute_error
print("The Explained Variance: %.2f" % regressor.score(X_test, y_test))
print("The Mean Absolute Error: %.2f degrees celsius" % mean_absolute_error(y_test, prediction))
print("The Median Absolute Error: %.2f degrees celsius" % median_absolute_error(y_test, predicti
on))
```

```
The Explained Variance: 0.90
The Mean Absolute Error: 2.69 degrees celsius
The Median Absolute Error: 2.17 degrees celsius
```

Рис.3.14 Отримання результатів прогнозування

3.4 Створення нейронної мережі з використанням Google Open Source TensorFlow

Щоб отримати інтерпретоване розуміння валідності моделей, було використано score () функцію моделі регресорів, щоб визначити, що модель здатна пояснити близько 90% дисперсії, що спостерігається в підсумковій змінній, середній температурі. Крім того, було використано mean_absolute_error () і median_absolute_error () з sklearn.metrics модуля, щоб визначити, що в середньому передбачене значення становить близько 3 градусів за Цельсієм.

У науці про дані і машинному навчанні існує безліч інших алгоритмів, які долають це припущення про лінійність. В останні роки одним з найбільш популярних напрямків стало застосування нейронних мереж для вирішення широкого кола завдань машинного навчання. Нейронні мережі мають потужний спосіб використання методів навчання, заснованих як на лінійних, так і

нелінійних операціях. Нейронні мережі натхненні біологічними нейронами в мозку, які працюють в складній мережі взаємодій, щоб передавати, збирати і вивчати інформацію, засновану на історії вже зібраної інформації. Цікавлять нас обчислювальні нейронні мережі аналогічні нейронам головного мозку тим, що вони являють собою сукупність нейронів (вузлів), які приймають вхідні сигнали (числові величини), обробляють вхідні сигнали і передають оброблені сигнали іншим нижчестоящим агентам в мережу. Обробка сигналів у вигляді числових величин, які проходять через нейронну мережу, є дуже потужною функцією, яка не обмежується лінійними відносинами. Графічно нейронна мережа, подібна до тієї, що описана вище, показана на рисунку нижче.

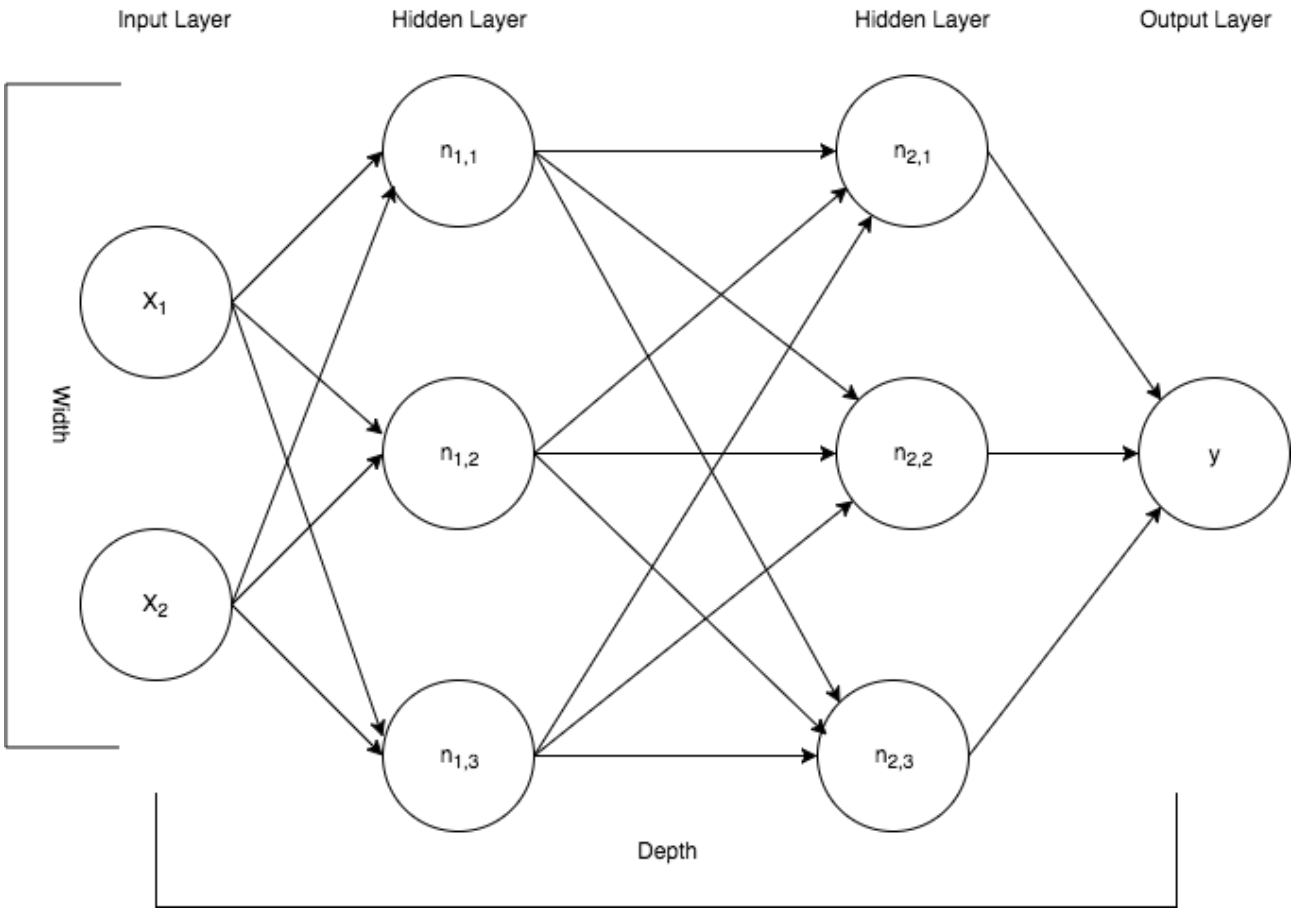

Рис.3.15. Графічне відображення нейронної мережі

Нейронна мережа, зображена вище, містить зліва вхідний шар, що представляє дві особливості, x1 і x2, які живлять нейронну мережу. Ці дві функції передаються в нейронну мережу, обробляються і передаються через два шари

нейронів, які називаються прихованими шарами. Це зображення показує два прихованих шари з кожним шаром, що містить три нейрона (вузла). Потім сигнал виходить з нейронної мережі і агрегується на вихідному шарі у вигляді єдиного числового прогнозованого значення.

Кожна стрілка представляє математичне перетворення значення, що починається біля основи стрілки, та потім множиться на вагу, що характерна для цього шляху. Кожен вузол в шарі отримує значення таким чином. Потім всі значення, що сходяться у вузлі, підсумовуються. Саме ця сукупність множення на ваги і підсумовування продуктів визначає лінійні операції нейронної мережі, про які було сказано раніше.

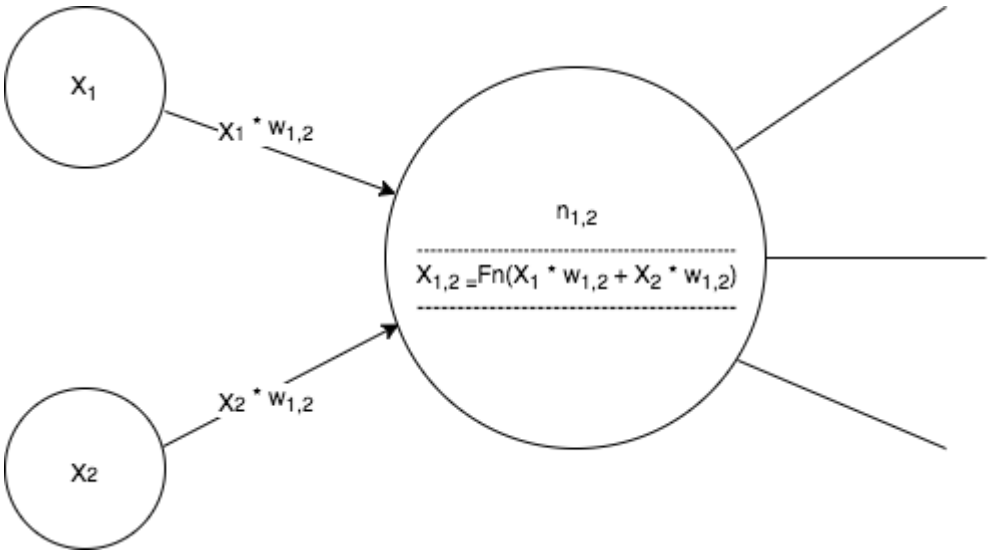

Рис.3.16. Графічне відображення лінійних операцій

Після виконання підсумовування в кожному вузлі до суми застосовується спеціальна нелінійна функція, яка зображена на зображенні вище (рис.3.16) як Fn (...). Ця спеціальна функція вводить нелінійні характеристики в нейронну мережу та називається функцією активації. Саме ця нелінійна характеристика, викликана функціями активації, надає багатошаровим нейронним мережам свою

силу. Якби не нелінійність, додана до процесу, тоді всі вітки фактично просто алгебраїчно об'єдналися б в одну постійну операцію, що складається з множення вхідних даних на деяке плоске значення коефіцієнта (тобто лінійну модель).

У світі алгоритмів машинного навчання регресорів оцінюється точність з використанням функції вартості (інакше кажучи, «втрати» або «цілі»), а саме суми квадратів помилок (SSE). Звернемо увагу, що це є узагальнене твердження на весь континуум машинного навчання, а не тільки на нейронні мережі. Наш нейромережевий регресор буде виконувати ітерацію по вводу даних навчання в значеннях параметрів, обчислювати функцію вартості (використовуючи SSE) і вносити коригування у ваги таким чином, щоб мінімізувати функцію вартості. Цей процес ітераційного проштовхування об'єктів за алгоритмом і оцінки того, як регулювати вагові коефіцієнти на основі функції вартості, по суті, є тим, що відомо як оптимізація моделі.

Алгоритми оптимізації моделей дуже важливі для побудови надійних нейронних мереж. Оскільки приклади передаються через мережеву архітектуру (тобто ширину і глибину), а потім оцінюються по функції вартості, ваги коректуються. Кажуть, що моделі «навчаються», коли функція оптимізатора ідентифікує, що коригування ваги була проведена таким чином, щоб не поліпшити (знизити) функцію вартості, яка зареєстрована оптимізатором так, щоб вона не змінювала ваги в цьому напрямку знову.

Бібліотека Google TensorFlow складається з декількох API, найбільш популярним з яких є Core API, який надає користувачеві низькорівневий набір інструментів для визначення і навчання практично будь-якого алгоритму машинного навчання з використанням символічних операцій. Це називається TensorFlow Core. Хоча TensorFlow Core - це дивовижний API з широкими можливостями додатків, ми зосередимось на більш новому, більш високому рівні - API, розробленому командою TensorFlow, який в сукупності називається API Estimator. Команда TensorFlow розробила API Estimator, щоб зробити бібліотеку більш доступною для повсякденного розробника. Ця високорівнева API надає

загальний інтерфейс для train (...) моделей, evaluate (...) моделей і predict (...) результатів невідомих випадків, аналогічний (і знаходиться під впливом) популярної бібліотеки Sci-Kit Learn, що досягається шляхом реалізації загального інтерфейсу для різних алгоритмів. Крім того, у даному API вбудовано безліч передових методів машинного навчання, абстракцій і можливостей для масштабування.

Все це гарне машинне навчання призводить до появи набору інструментів, реалізованих в базовому класі Estimator, а також кількох попередньо підготовлених типів моделей, які знижують бар'єр для входу при використанні TensorFlow, тому його можна застосовувати для безлічі повсякденних проблем (або можливостей). Абстрагуючись від багатьох звичних і ручних аспектів таких речей, як написання циклів навчання або робота з сесіями, розробник може зосередитися на більш важливі речі, таких як швидке використання декількох моделей і архітектур моделей, щоб знайти ту, яка найкраще відповідає їхнім потребам. Тут ми будемо використовувати один із дуже потужних інструментів оцінки нейронних мереж DNNRegressor:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.metrics import explained_variance_score, \
    mean_absolute_error, \
    median_absolute_error
from sklearn.model_selection import train_test_split
```

Рис.3.17. Імпорт необхідних бібліотек

Тепер давайте візьмемося за дані, щоб ознайомитися з ними.

```
# read in the csv data into a pandas data frame and set the date as the index
df = pd.read_csv('end-part2_df.csv').set_index('date')

# execute the describe() function and transpose the output so that it doesn't overflow the width
of the screen
df.describe().T
```

Рис.3.18. Імпорт вихідних даних для дослідження

Таблиця 3.7 Вихідні дані

	count	mean	Std	Min	25%	50%	75%	max
meantempm	997,0	13.129388	10.971591	-17,0	5.0	15,0	22,00	32,00
maxtempm	997,0	19.509529	11.577275	-12,0	11,0	22,0	29,00	38,00
mintempm	997,0	6.438315	10.957267	-27,0	-2,0	7,0	16,00	26,00
meantempm_1	997,0	13.109328	10.984613	-17,0	5.0	15,0	22,00	32,00
meantempm_2	997,0	13.088265	11.001106	-17,0	5.0	14,0	22,00	32,00
meantempm_3	997,0	13.066199	11.017312	-17,0	5.0	14,0	22,00	32,00
meandewptm_1	997,0	6.440321	10.596265	-22,0	-2,0	7,0	16,00	24,00
meandewptm_2	997,0	6.420261	10.606550	-22,0	-2,0	7,0	16,00	24,00
meandewptm_3	997,0	6.393180	10.619083	-22,0	-2,0	7,0	16,00	24,00
meanpressurem_1	997,0	1016.139418	7.582453	989,0	1011,0	1016,0	1021,00	1040,00
meanpressurem_2	997,0	1016.142427	7.584185	989,0	1011,0	1016,0	1021,00	1040,00
meanpressurem_3	997,0	1016.151454	7.586988	989,0	1011,0	1016,0	1021,00	1040,00
maxhumidity_1	997,0	88.107322	9.280627	47,0	83,0	90,0	93,00	100,00
maxhumidity_2	997,0	88.106319	9.280152	47,0	83,0	90,0	93,00	100,00
maxhumidity_3	997,0	88.093280	9.276775	47,0	83,0	90,0	93,00	100,00
minhumidity_1	997,0	46.025075	16.108517	9,0	35,0	45,0	56,00	92,00
minhumidity_2	997,0	46.021063	16.105530	9,0	35,0	45,0	56,00	92,00
minhumidity_3	997,0	45.984955	16.047081	9,0	35,0	45,0	56,00	92,00
maxtempm_1	997,0	19.489468	11.588542	-12,0	11,0	22,0	29,00	38,00

maxtempm_2	997,0	19.471414	11.603318	-12,0	11,0	22,0	29,00	38,00
maxtempm_3	997,0	19.455366	11.616412	-12,0	11,0	22,0	29,00	38,00
mintempm_1	997,0	6.417252	10.974433	-27,0	-2,0	7,0	16,00	26,00
mintempm_2	997,0	6.394183	10.988954	-27,0	-2,0	7,0	16,00	26,00
mintempm_3	997,0	6.367101	11.003451	-27,0	-2,0	7,0	16,00	26,00
maxdewptm_1	997,0	9.378134	10.160778	-18,0	1,0	11,0	18,00	26,00
maxdewptm_2	997,0	9.359077	10.171790	-18,0	1,0	11,0	18,00	26,00
maxdewptm_3	997,0	9.336008	10.180521	-18,0	1,0	11,0	18,00	26,00
mindewptm_1	997,0	3.251755	11.225411	-28,0	-6,0	4,0	13,00	22,00
mindewptm_2	997,0	3.229689	11.235718	-28,0	-6,0	4,0	13,00	22,00
mindewptm_3	997,0	3.198596	11.251536	-28,0	-6,0	4,0	13,00	22,00
maxpressurem_1	997,0	1019.913741	7.755590	993,0	1015,0	1019,0	1024,00	1055,00
maxpressurem_2	997,0	1019.917753	7.757705	993,0	1015,0	1019,0	1024,00	1055,00
maxpressurem_3	997,0	1019.927783	7.757805	993,0	1015,0	1019,0	1024,00	1055,00
minpressurem_1	997,0	1012.317954	7.885743	956,0	1008,0	1012,0	1017,00	1035,00
minpressurem_2	997,0	1012.319960	7.886681	956,0	1008,0	1012,0	1017,00	1035,00
minpressurem_3	997,0	1012.326981	7.889511	956,0	1008,0	1012,0	1017,00	1035,00
precipm_1	997,0	2.593180	8.428058	0.0	0.0	0.0	0,25	95,76
precipm_2	997,0	2.593180	8.428058	0.0	0.0	0.0	0,25	95,76
precipm_3	997,0	2.573049	8.410223	0.0	0.0	0.0	0,25	95,76

Тепер видалимо стовпці «mintempm» і «maxtempm», оскільки вони не мають ніякого значення для передбачення середніх значень середньої температури. Ми намагаємося передбачити майбутнє, тому у нас явно не може бути даних про майбутнє. Як і у всіх контрольованих додатках машинного навчання, ми будем ділити свій набір даних на набори для навчання і тестування. Однак, щоб краще пояснити ітеративний процес навчання цієї нейронної мережі, ми будемо використовувати додатковий набір даних, який називається «набором перевірки». Для навчального набору використаємо 80 відсотків даних, а для набору для тестування і перевірки кожен з них буде складати 10% від решти даних.

```
# split data into training set and a temporary set using sklearn.model_selection.traing_test_spl
it
X_train, X_tmp, y_train, y_tmp = train_test_split(X, y, test_size=0.2, random_state=23)
```

```
# take the remaining 20% of data in X_tmp, y_tmp and split them evenly
X_test, X_val, y_test, y_val = train_test_split(X_tmp, y_tmp, test_size=0.5, random_state=23)

X_train.shape, X_test.shape, X_val.shape
print("Training instances   {}, Training features   {}".format(X_train.shape[0], X_train.shape
[1]))
print("Validation instances {}, Validation features {}".format(X_val.shape[0], X_val.shape[1]))
print("Testing instances    {}, Testing features    {}".format(X_test.shape[0], X_test.shape
[1]))
```

```
Training instances   797, Training features   36
Validation instances 100, Validation features 36
Testing instances    100, Testing features    36
```

Рис.3.19. Розділення даних на набори для тестування

Перший крок, який необхідно зробити при побудові моделі нейронної мережі, - створити екземпляр класу tf.estimator.DNNRegressor (...). Конструктор класу має кілька параметрів:

- feature_columns: Структура, подібна списку, що містить визначення імені та типів даних для функцій, які вводяться в модель;
- hidden_units: Структура типу списку, що містить визначення ширини і глибини нейронної мережі;

- optimizer: Примірник підкласу tf.Optimizer, який оптимізує вагаи моделі під час навчання; за замовчуванням використовується оптимізатор AdaGrad;
- activation_fn: Функция активації, що використовується для введения нелінійності в мережу на кажному рівні;
- model_dir: Створюваний каталог, який буде містити метадані та інші контрольні точки, збережені для моделі;

Почнемо з визначення списку стовпців числових функцій. Для цього використаємо tf.feature_column.numeric_column функцію, яка повертає FeatureColumn екземпляр для числових функцій з безперервними значеннями.

```
feature_cols = [tf.feature_column.numeric_column(col) for col in X.columns]
```

Рис.3.20. Визначення списку стовбців числових функцій

Визначивши стовпці об'єктів, можна створити екземпляр DNNRegressor і зберегти його в змінній регресорів. Звернемо увагу, що ми отримуємо нейронну мережу, яка має два шари глибини, де обидва шари мають ширину в 50 вузлів. Також необхідно, щоб дані моделі зберігалися в каталозі з іменем tf_wx_model.

```
regressor = tf.estimator.DNNRegressor(feature_columns=feature_cols,
                                      hidden_units=[50, 50],
                                      model_dir='tf_wx_model')
```

```
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_tf_random_seed': 1, '_save_checkpoints_steps': None, '_save_che
ckpoints_secs': 600, '_model_dir': 'tf_wx_model', '_log_step_count_steps': 100, '_keep_checkpoin
t_every_n_hours': 10000, '_save_summary_steps': 100, '_keep_checkpoint_max': 5, '_session_confi
g': None}
```

Рис.3.21. Задання параметрів необхідної нейронної мережі

Наступне, що необхідно зробити, це визначити багаторазову функцію, яка зазвичай називається «функцією введення», яку буде викликати wx_input_fn (...). Ця функція буде використовуватися для подачі даних в нейронну мережу на етапах навчання і тестування. Існує багато різних способів створення вхідних функцій, ми будемо використовувати одну з них на основі

tf.estimator.inputs.pandas_input_fn (...) оскільки наші дані знаходяться в структурах даних pandas.

```
def wx_input_fn(X, y=None, num_epochs=None, shuffle=True, batch_size=400):
    return tf.estimator.inputs.pandas_input_fn(x=X,
                                               y=y,
                                               num_epochs=num_epochs,
                                               shuffle=shuffle,
                                               batch_size=batch_size)
```

Рис.3.21. Визначення функції подачі даних в нейронну мережу

Звертаємо увагу, що ця wx_input_fn (...) функція приймає один обов'язковий і чотири необов'язкові параметри, які потім передаються вхідний функції TensorFlow спеціально для даних pandas, що повертаються. Це дуже потужна функція API TensorFlow.

Параметри функції визначаються наступним чином:

- X: Вхідна особливість, яка буде подаватися в один з трьох DNNRegressor методів інтерфейсу (train, evaluatei predict) y: Цільові значення X, які є необов'язковими і не будуть передані в predict виклику;
- y: Цільові значення X, які є необов'язковими і не будуть передані на predict виклик;
- num_epochs: Необов'язковий параметр. Передається у випадку, коли алгоритм виконується по всьому набору даних один раз.
- shuffle: Необов'язковий параметр, який вказує, чи слід випадковим чином вибирати пакет (підмножину) набору даних при кожному виконанні алгоритму;
- batch_size: Кількість вибірок, що включаються при кожному виконанні алгоритму;

Визначимо простий цикл навчання, щоб навчати модель на систему адаптації і періодично оцінювати її на даних оцінки.

```
evaluations = []
STEPS = 400
for i in range(100):
    regressor.train(input_fn=wx_input_fn(X_train, y=y_train), steps=STEPS)
    evaluations.append(regressor.evaluate(input_fn=wx_input_fn(X_val,
                                                               y_val,
                                                               num_epochs=1,
                                                               shuffle=False)))
```

```
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Saving checkpoints for 1 into tf_wx_model/model.ckpt.
INFO:tensorflow:step = 1, loss = 1.11335e+07
INFO:tensorflow:global_step/sec: 75.7886
INFO:tensorflow:step = 101, loss = 36981.3 (1.321 sec)
INFO:tensorflow:global_step/sec: 85.0322
... A WHOLE LOT OF LOG OUTPUT ...
INFO:tensorflow:step = 39901, loss = 5205.02 (1.233 sec)
INFO:tensorflow:Saving checkpoints for 40000 into tf_wx_model/model.ckpt.
INFO:tensorflow:Loss for final step: 4557.79.
INFO:tensorflow:Starting evaluation at 2017-12-05-13:48:43
INFO:tensorflow:Restoring parameters from tf_wx_model/model.ckpt-40000
INFO:tensorflow:Evaluation [1/1]
INFO:tensorflow:Finished evaluation at 2017-12-05-13:48:43
INFO:tensorflow:Saving dict for global step 40000: average_loss = 10.2416, global_step = 40000,
loss = 1024.16
INFO:tensorflow:Starting evaluation at 2017-12-05-13:48:43
INFO:tensorflow:Restoring parameters from tf_wx_model/model.ckpt-40000
INFO:tensorflow:Finished evaluation at 2017-12-05-13:48:43
INFO:tensorflow:Saving dict for global step 40000: average_loss = 10.2416, global_step = 40000,
loss = 1024.16
```

Рис.3.23. Визначення циклу навчання моделі

Вищезгаданий цикл повторюється 100 раз. У тілі циклу йде виклик train (...) методу об'єкта регресора, передаючи його власному повторно використовуваному об'єкту, що у свою чергу, проходить набір навчальних функцій. При виконанні ми залишаємо всі параметри за замовчуванням num_epochsправним, що в основному говорить: «Все-одно, скільки разів ви проходите навчальний набір, просто продовжуйте тренувати алгоритм для кожного значення batch_sizeпо замовчуванням 400» (приблизно половина розміру навчального набору). Також залишаємо параметр shuffle за замовчуванням, щоб під час навчання дані вибиралися випадковим чином, тобто уникнути будь-яких послідовних зв'язків в даних. Останній параметр train (...)

методу steps, що дорівнює 400, означає, що тренувальний набір буде упакований 400 раз за цикл.

Для кожної ітерації циклу було виконано evaluate (...) метод та записано кожний його висновок в список. Кожного разу, коли train (...) метод запускається, він вибирає випадкову партію тренувальних записів і проводить їх по мережі, поки не буде зроблений прогноз, і для кожного запису обчислюється функція втрат. Потім, виходячи з розрахованих втрат, ваги коректуються відповідно до логіки оптимізатора, який досить добре виконує налаштування в напрямку, який зменшує загальні втрати для наступної ітерації. Ці значення втрат, загалом, якщо швидкість навчання досить мала, з часом зменшуються з кожною ітерацією або кроком.

Однак після певної кількості цих ітерацій навчання, на ваги починають впливати не тільки загальні тенденції в даних, але і неінформативний шум, що присутній практично у всіх реальних даних. У цей момент мережа знаходиться під сильним впливом навчальних даних і стає нездатною узагальнювати прогнози по загальній сукупності даних (даних, які вона ще не бачила). Це відноситься до проблеми, про яку було згадано раніше. Дуже важливо періодично переривати навчання і оцінювати, як модель узагальнюється для оцінки або валідації набору даних. Давайте поглянемо на те, що функція evaluate (...) повертає, подивившись результати оцінки першої ітерації циклу:

```
{'average_loss': 31.116383, 'global_step': 400, 'loss': 3111.6382}
```

Рис.3.24. Результати оцінки першої ітерації циклу

Як ви можете бачити, вона виводить середню втрату (Mean Squared Error) і загальну втрату (Sum of Squared Error) для етапу навчання, який для цього є 400-м кроком. Те, що ви зазвичай бачите в добре навченої мережі, - це тенденція, коли втрати на навчання і оцінку більш-менш постійно знижуються паралельно. Однак в переоснащеній моделі в певний момент часу, фактично в точці, де починає відбуватися перебирання, навчальний набір перевірки перестане бачити

скорочення результатів свого evaluate (...) методу. Тут ми хочемо припинити подальше навчання моделі, бажано безпосередньо перед тим, як відбудуться ці зміни.

Тепер, коли у нас є набір оцінок для кожної з ітерацій, давайте побудуємо їх як функцію кроків навчання, щоб переконатися, що ми не перетренували нашу модель. Для цього будемо використовувати простий графік розсіювання з pyplo тмодуля matplotlib.

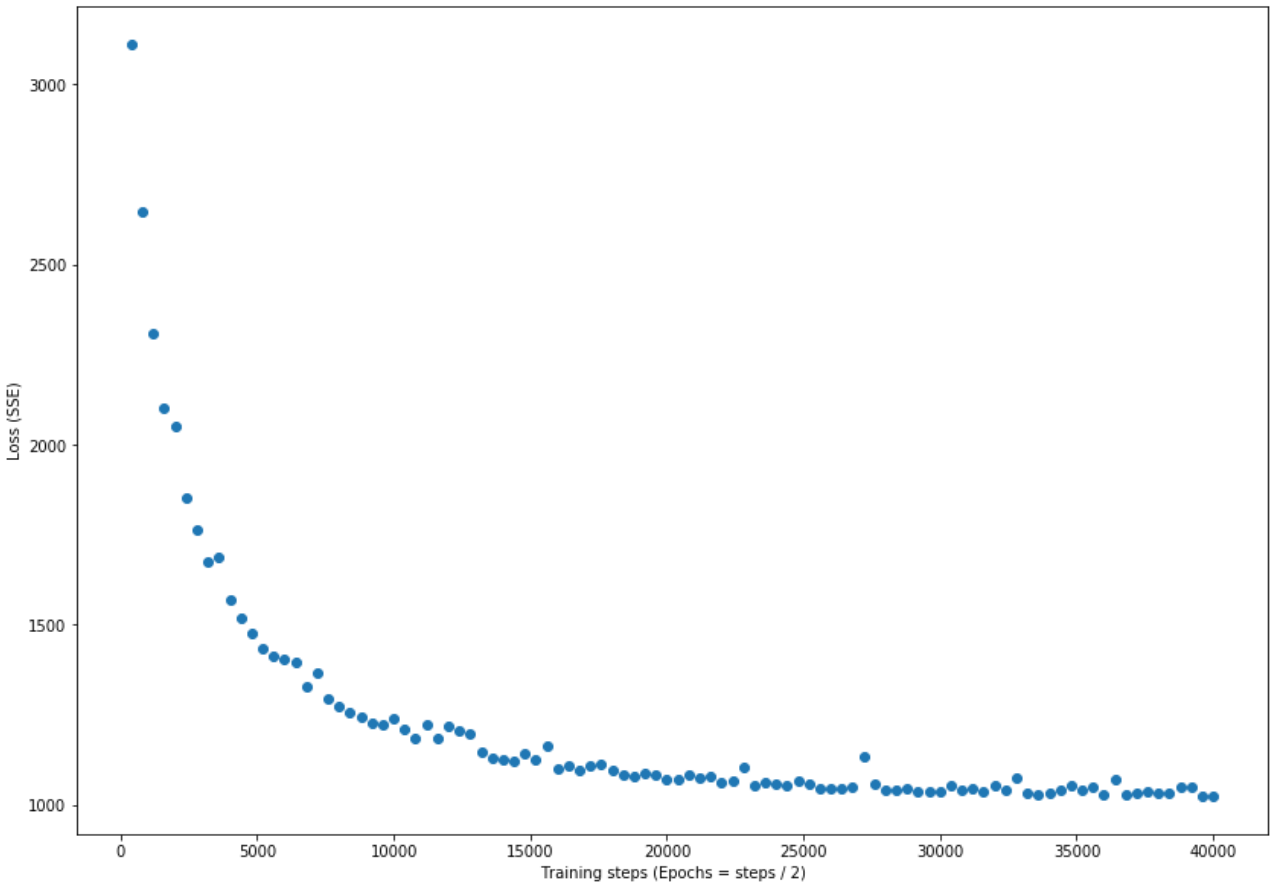

Рис.3.25. Розсіювання наборів оцінок для кожної ітерації

З наведеного вище графіка видно, що після всіх цих ітерацій ми не перенавчили модель, тому що втрати при оцінці ніколи не демонструють істотного зміни напрямку в бік збільшення значення. Тепер ми можемо сміливо переходити до прогнозування на основі набору тестових даних і оцінювати, як модель прогнозує середні погодні температури.

Подібно двом іншим методам регресорів, які було продемонстровано, predict (...) метод вимагає передачі input_fn, використовуючи wx_input_fn (...) та передаючи йому набір тестових даних, вказуючи, що він num_epochs рівний одиниці та вимкненим shuffle. Потім зробимо деяке форматування ітерацій, що повертаються з predict (...) методу, щоб у нас був цілий масив прогнозувань. Потім використовуючи масив передбачення з методами sklearn explained_variance_score (...), mean_absolute_error (...) і median_absolute_error (...) можна визначити, наскільки добре передбачення є сумісними по відношенню до відомих цілей y_test, що у свою чергу говорить розробнику, які прогнозні можливості моделі.

```
pred = regressor.predict(input_fn=wx_input_fn(X_test,
                                              num_epochs=1,
                                              shuffle=False))
predictions = np.array([p['predictions'][0] for p in pred])

print("The Explained Variance: %.2f" % explained_variance_score(
                                             y_test, predictions))
print("The Mean Absolute Error: %.2f degrees Celcius" % mean_absolute_error(
                                             y_test, predictions))
print("The Median Absolute Error: %.2f degrees Celcius" % median_absolute_error(
                                             y_test, predictions))
```

```
INFO:tensorflow:Restoring parameters from tf_wx_model/model.ckpt-40000
The Explained Variance: 0.88
The Mean Absolute Error: 3.11 degrees Celcius
The Median Absolute Error: 2.51 degrees Celcius
```

Рис.3.26. Отримання результатів прогнозування

Висновки до розділу 3

В результаті виконання даного розділу, було продемонстровано загальний процес виконання аналітичного проекту: отримання даних, обробка даних, дослідницький аналіз даних, вибір моделі, побудова моделі та її оцінка. Продемонстровано, як вибрати функції, що не порушують основні припущення техніки лінійної регресії, з використанням двох популярних бібліотек StatsModels и Scikit Learn. Також, було показано як використовувати API – інтерфейс TensorFlow високого рівня та описано, що відбувається під цими рівнями абстракції. Описано проблеми, що пов’язані із переповненням моделі та пояснено доцільність проведення експериментів з декількома типами моделей для найкращого вирішення проблеми.

ЗАГАЛЬНІ ВИСНОВКИ

В ході магістерської дисертації було виконано аналіз застосування машинного навчання в мережах Інтернет речей. Зокрема розкрито базову архітектуру мереж Інтернет речей та проаналізовано аналіз їх даних у хмарних платформах із можливістю впровадження машинного навчання.

В результаті виконання роботи отримано систему управління на базі машинного навчання. Дана система являє собою модель прогнозування майбутніх погодних показників на основі минулих даних. На прикладі даної реалізації можна зробити висновок про необхідність впровадження технологій машинного навчання в тому числі, як було продемонстровано, нейронних мереж у системах IoT для повного використання потенціалу даних та створення інтелектуальних додатків IoT.

Отже, в роботі були досягнуті наступні цілі:

1. Проведено загальний огляд IoT;
2. Проаналізовано основні методи та алгоритми машинного навчання;
3. Розроблено модель прогнозування погодних показників на основі даних, отриманих з мереж IoT;

ПЕРЕЛІК ПОСИЛАНЬ

1. [Електронний ресурс]: Матеріали вільної енциклопедії «Вікіпедія». URL: http://ru.wikipedia.org/wiki.
2. [Електронний ресурс]: Матеріали відкритого курсу по машинному навчанню від компанії ODS. URL: https://habrahabr.ru/company/ods/blog/325654/, (режим доступу – вільний).
3. [Електронний ресурс]: Матеріали сайту machinelearning. URL: http://www.machinelearning.ru/wiki/index.php?title=Переобучение, (режим доступу – вільний).
4. [Електронний ресурс]: Стаття по регуляризації. URL: https://ru.wikipedia.org/wiki/Регуляризация_(математика), (режим доступу – вільний).
5. [Електронний ресурс]: Матеріали сайту MSDN. URL: https://msdn.microsoft.com/ru-ru/magazine/dn904675.aspx, (режим доступу – вільний).
6. Теорія та практика машинного навчання: навчальний посібник / В. В. Воронина, А. В. Михеев, Н. Г. Ярушкина, К. В. Святов. – Ульяновск : УлГТУ, 2017. – 290 с.