Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2022 р

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**РЕГРЕСІЙНІ МОДЕЛІ ПРОГНОЗУВАННЯ ТРАФІКУ КОМП'ЮТЕРНИХ МЕРЕЖ**»

Захищено на засіданні
Атестаційної комісії № 44
протокол № __ від __.12.2022 р.
Оцінка _____ / _____
Голова Атестаційної комісії
д.т.н., професор
_____ **МІНУХІН С. В.**

Виконав:
студент 2 курсу, групи КІ– 61
Спеціальність: 123 –Комп'ютерна інженерія
Галузь знань: 12 – Інформаційні технології
ГОЛОВЧЕНКО Віталій Борисович

Керівник:
к. т. н., доцент кафедри ТПС
СТРІЛЕЦЬ Вікторія Євгенівна

Рецензент:
д.т.н., доцент, професор кафедри математичного моделювання і штучного інтелекту
СКОБ Юрій Олексійович

Харків – 2022

АНОТАЦІЯ

Дана кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та чотирьох додатків. Загальний обсяг роботи становить 73 сторінки, де на 55 сторінках викладена основна частина, враховуючи 10 рисунків та 1 таблицю. Документ містить 25 посилань на використані джерела, що займають 3 сторінки, та 4 додатки на 13 сторінках.

Кваліфікаційне дослідження присвячене проблемі прогнозування трафіку комп'ютерних мереж. Його метою є підвищення точності прогнозування трафіку комп'ютерних мереж на основі алгоритму випадкових лісів та алгоритмів лінійної регресії. Під час виконання даної роботи були розглянуті класичні методи прогнозування стану систем на основі машинного навчання, проведено аналіз методів лінійної регресії, методу випадкових лісів. Для досягнення поставленої мети був знайдений відповідний набір даних, обрано пакет програмного забезпечення, а також метрики для оцінки її ефективності.

Результатом даного дослідження є декілька натренованих моделей, які здатні спрогнозувати стан комп'ютерної мережі, що базуються на методі випадкових лісів та методах лінійної регресії. Областю використання запропонованої моделі та методу є розробка моделей, обчислювальних методів та засобів, направлених на спрощення прогнозування трафіку та стану комп'ютерних мереж.

Ключові слова: прогнозування трафіку, лінійна регресія, випадковий ліс, штучний інтелект, машинне навчання, навчання з вчителем.

ABSTRACT

This qualification work consists of introduction, three sections, conclusions, a list of used sources and four appendices. The total volume of work is 73 pages, where the main part is laid out on 55 pages, taking into account 10 figures and 1 table. The document contains 25 sources in the link list, which were reflected on 3 pages, and 4 appendices on 13 pages.

This research is devoted to the problem of prediction traffic of computer networks. Its purpose is to increase the accuracy of computer network traffic prediction on the basis of random forests algorithm and linear regression algorithms. During the performance of this work classical methods of prediction the state of systems on the basis of machine learning were considered, the analysis of linear regression and random forests methods were carried out. To achieve this purpose, a corresponding set of data was found, a software package was selected, and to evaluate its effectiveness, metrics were selected.

The result of this investigation is a number of trained models that can predict the state of the computer network, which is based on random forests methods, the method of gradient boosting of decision tree and linear regression methods. The area of use of the proposed model and method is development of an effective model, computing methods and means aimed at simplification of traffic prediction and the state of computer networks.

Key words: traffic forecasting, linear regression, random forest, artificial intelligence, machine learning, teacher learning.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AUC – area under curve.

FP – false positive.

FN – false negative.

ROC – receiver operating characteristic.

OOB – Out-Of-Bag.

GB – Gradient boosting.

MAE – Mean Absolute Error.

MSE – Mean Squared Error.

MAPE – Mean Absolute Percentage Error.

SMAPE – Symmetric Mean Absolute Percentage Error.

MASE – Mean absolute scaled error.

CV – Cross-Validation.

ВСТУП

Прогнозування мережного трафіку представляє значний інтерес у таких областях як відстеження перевантажень у мережі, контроль потоків даних та мережеве управління. Явище та класифікація аномалій мережевого трафіку з багатьох подій мережі стає корисним при створенні моделі роботи мережі, оптимізації налаштувань обладнання та підтримці заданого QoS. Ретельно підібрана модель трафіку здатна виявити та передбачити найважливіші характеристики мережевого трафіку, такі як короткочасно та довготривалі залежні процеси, самоподібність на великих тимчасових масштабах. В цей час велика увага приділяється методам прогнозування роботи комп'ютерних мереж у різних умовах. У роботі пропонується розроблення методу прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання. А саме з використанням алгоритму випадкових лісів та лінійного алгоритму регресії.

Таким чином, **об'єктом** цього дослідження є процес прогнозування трафіку комп'ютерних мереж, під час якого виконується автоматизований аналіз вхідних даних. **Задачу** прогнозування трафіку комп'ютерних мереж можливо сформулювати як задачу аналізу даних, де в якості вхідних даних використовують інформацію про кількість підключених абонентів, показники використання IP, час підключення абонентів та час використання системи за якими будується набір правил, що становлять основу для подальшого аналізу трафіку мережі. На практиці часто виникає потреба в універсальній системі, яка могла б аналізувати різнорідні дані, як в структурованій формі, так і в неструктурованій і дозволяла б автоматизувати широкий спектр заходів в рамках аналізу стану системи, а тому **предмет** дослідження можливо сформулювати як моделі й методи прогнозування трафіку комп'ютерних

мереж на основі алгоритму випадкових лісів та на основі лінійних алгоритмів регресії.

Наразі процес аналізу комп'ютерного трафіку, діагностика стану системи та прогнозування проблем мережі є досить довгими й не завжди точними, тому **метою** даного дослідження є підвищення точності прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання.

Для досягнення мети необхідно вирішити такі задачі:

– Провести порівняльний аналіз методів прогнозування стану систем.

– Розробити моделі прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання.

– Розробити та протестувати програмний додаток, який реалізує регресійні моделі прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання.

Результати дослідження можна використати як основу для подальших теоретичних та практичних досліджень у даній сфері, а також як підґрунтя для розробки окремої комерційної системи автоматизованого методу прогнозування мережного трафіку.

РОЗДІЛ 1
МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ПРОГНОЗУВАННЯ СТАНУ СИСТЕМ

1.1 Задача прогнозування

Завдання прогнозування деякої величини Y за доступними значеннями змінних $X_1 \cdots X_n$ часто виникають в різних областях людської діяльності:

1) прогноз властивостей ще несинтезованого хімічного з'єднання по його молекулярній формулі;
2) діагностика ходу технологічного процесу;
3) діагностика стану технічного обладнання;
4) прогноз фінансових індикаторів.

У прикладних дослідженнях нерідко виникають ситуації, коли математичне моделювання, засноване на використанні точних законів виявляється досить складним, але дослідник вільний у виборі прецеденту – спостереження досліджуваного процесу або явища, що включає в себе значення прогнозованих величин Y та змінних $X_1 \cdots X_n$. У цих випадках методи, засновані на вивченні прецедентів, можуть бути використані для вирішення проблем прогнозування.

Прогнозування здійснюється за вибіркою прецедентів, яка зазвичай є випадковою вибіркою об'єктів з відомими значеннями Y, $X_1 \cdots X_n$. Вибірку прецедентів також заведено називати навчальною вибіркою.

В процесі навчання проводиться пошук емпіричних закономірностей, що пов'язують прогнозовану змінну Y зі змінними $X_1 \cdots X_n$. Дані закономірності далі використовуються при прогнозуванні. Методи, засновані на навчанні за прецедентів, також заведено називати методами машинного навчання. Багато таких методів використовуються для розв’язання задач

прогнозування. На сьогоднішній день існують різні методи прогнозування трафіку комп'ютерних мереж, але майже жоден з них не використовує такий досить потужний інструмент сьогодення як машинне навчання. Це, наприклад, метод опорних векторів, регресійні дерева, лінійна регресія, випадкові ліса прийняття рішень.

1.2. Задача регресії

Регресія – один з розділів машинного навчання, присвячений прогнозуванню цільовій змінній місця на числовій прямій. Завдання передбачення дійсної змінної за початковою вибіркою, тобто за набором пар предикат-відгук, називають завданням *регресії* [1].

Задачу регресії можливо сформувати так: нехай X – множина даних – описів деяких об'єктів, Y – множина можливих відповідей для X. У задачі регресії передбачається, що існує деяка цільова залежність *y:* $X \rightarrow Y$, значення якої відомі лише на об'єктах навчальної вибірки $XY = \{(x_1, y_1), \dots, (x_n, y_n)\}, x \in X, y \in Y$. Необхідно отримати модель (алгоритм) *a:* $X \rightarrow Y$, який наближає цільову залежність як на множині *XY*, так і на *X*.

Тобто розв'язати задачу регресії – це знайти алгоритм або модель, які можуть узагальнити емпіричні факти (вивід загальних знань з окремих спостережень). Можна відмітити, що задача регресії є подібною до задачі апроксимації.

Головна особливість задачі регресії полягає у тому, що функція *a:* $X \rightarrow Y$ є безперервною дійсною функцією.

Якщо в якості матриці *X* взяти вибірку з даними по використанню мережевого трафіку, отриману після попередньої обробки даних, а в якості відгуку у вектор зі певним значенням, то початкове завдання потрапляє під визначення задачі регресії. Такая модель є функцією, яка перетворює вхідний простір ознак у вихідний простір залежних ознак. На виході має вийти число

(6, 25, 236.58), наприклад вартість квартири, ціна цінного паперу після пів року, очікуваний дохід магазину наступного місяця, якість вина при сліпому тестуванні, прогнозоване значення навантаження мережі тощо.

1.3. Метод опорних векторів для побудови регресорів

Метод опорних векторів [2] це – алгоритм машинного навчання, що може бути використаний як для задач прогнозування, так і для задач регресії. Суть методу полягає в побудові такої гіперплощини, яка ідеально розділяє вхідний набір даних на два класи, зберігаючи максимальну відстань до опорних векторів. Якщо замінити один опорний вектор іншим або навіть видалити його, гіперплощина змінить своє положення в просторі. Ця властивість методу опорного вектора дозволяє досягти найкращої можливої точності, зберігаючи властивість адаптації до нових точок у міру використання моделі. Схематично метод можна зобразити на рис 1.1 [3].

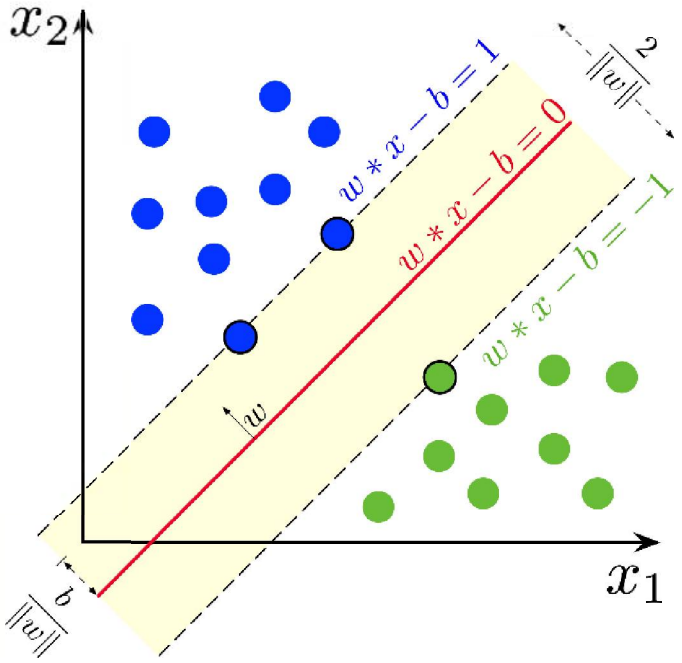

Рис. 1.1 – Схема методу опорних векторів.

На рис. (1.1) зображено два класи, що відділяються гіперплощиною $w * x - b = 0$. На площинах $w * x - b = 1$ та $w * x - b = -1$ розташовуються відповідно опорні вектори. Загальний розмір межі, де немає ніяких точок, визначається формулою $\frac{2}{||\bar{w}||}$, де $||\bar{w}||$ – норма вектора ваги ознак. Відстань від точки початку координат до оптимальної гіперплощини дорівнює $\frac{b}{||\bar{w}||}$, де b – елемент зміщення. Відповідними параметрами моделі, які потрібно визначити, є значення b та вектор $||\bar{w}||$. Варто зазначити, значення елементів векторів ознак повинні бути нормалізованими, оскільки вектор із надзвичайно великими значеннями з великою ймовірністю спотворить гіперплощину, що не є бажаним.

Як правило, метод опорних векторів використовується для вирішення задачі бінарної класифікації, як і логістична регресія. Виділяють наступні особливості, притаманні методу опорних векторів:

1. Неможливо розглядати результати як імовірнісні, оскільки в ньому не задіяна логістична функція.

2. У методі опорних векторів значення мітки може набувати значення $(-1, +1)$.

3. Метод опорних векторів використовує функцію завісних втрат:

$$J(w, y, x) = C \sum_{i=1}^{m} \left(1 - y_i (w^T x_i - b)\right)_+ + \frac{\|w\|_2^2}{2}, \qquad (1.1)$$

де C – гіперпараметр, що керує регуляризацією, і з збільшенням його значення, зменшується вплив регуляризації.

Таким чином, отримуємо, що модель штрафується тоді, коли дійсне та обчислене значення мають різні величини.

Даний метод має наступні переваги:

1) внаслідок вибору оптимальної гіперплощини, метод має досить гарне узагальнення без підбору гіперпараметрів;

2) метод володіє високою ефективністю обробки як для структурованих даних, так й для напівструктурованих, чи неструктурованих при попередній підготовці даних;

3) висока точність. Алгоритм опорних векторів будує оптимальну гіперплощину, на що не здатні деякі інші методи.

Проте, слід приділити увагу недолікам даного методу:

– при неструктурованих даних необхідна попередня обробка перед їх використанням;

– через відсутність використання логістичної функції, результати роботи моделі не неможливо використовувати як вірогідність належності об’єкту тому чи іншому класу;

– при наявності шумів у вхідній виборці, метод показує високу неефективність.

1.4 Дерева регресії

Дерево регресії – це вид дерев прийняття рішень. Воно використовує обчислення суми квадратів та регресійний аналіз для прогнозування значень цільової змінної. У дереві регресії обчислюється передбачене середнє значення для кожного вузла в дереві. Цей тип дерева генерується, коли призначення є безперервним.

Кожен вузол розділений на два або кілька дочірніх вузлів, щоб зменшити суму квадратів для вузла[5]. Сума квадратів – це функція, яка відсіює цільові значення, віддалені від середнього. Для кожного вузла виводиться середнє арифметичне разом зі стандартним відхиленням. Сума квадратів безпосередньо пов'язана зі стандартним відхиленням та числом рядків даних, що відповідають даному вузлу. Дочірні вузли, що відповідають чинним

категоріям предикатів, будуть об'єднуватися, якщо зростання суми квадратів не перевищує заданої межі. Для поділу кожного вузла вибирається предикат, який найсильніше знижує суму квадратів.

Процес побудови дерева рішень починається з кореневого вузла, який відповідає всім рядкам даних. Вузол поділятиметься на дочірні вузли доти, доки не перестане покращуватися сума квадратів або кількість рядків, що відповідає цьому вузлу, не стане надто маленькою. Процес також буде зупинено, якщо кількість вузлів у дереві рішень стане занадто великою.

Для оцінки прогностичної сили дерева регресії використовується R^2. Про надійне прогностичне дерево регресії говорять, коли його прогностична сила перевищує поріг 10%.

Дерева класифікації та регресії є одним із найпопулярніших методів вирішення багатьох практичних завдань, що обумовлено наступними причинами[6]:

1) дерева рішень дозволяють отримувати легко інтерпретовані моделі, що є набором правил виду “якщо…, то…”. Інтерпретація полегшується, зокрема, за допомогою можливості подати ці правила у вигляді наочної деревоподібної структури;

2) у силу своєї природи регресійні дерева дозволяють працювати зі змінними будь-якого типу без необхідності будь-якої попередньої підготовки цих змінних для введення в модель;

3) досліднику нема потреби явно задавати форму взаємозв'язку між відгуком і предикторами, як це, наприклад, відбувається у випадку зі звичайними регресійними моделями. Це виявляється особливо корисним при роботі з даними великого обсягу, про властивості яких мало відомо;

4) дерева рішень, по суті, автоматично виконують добір інформативних предикторів та враховують можливі взаємодії між ними. Це зокрема робить дерева рішень корисним інструментом аналізу даних;

5) дерева рішень можна ефективно застосовувати до даних із пропущеними значеннями, що дуже корисно при вирішенні практичних завдань, де наявність пропущених значень – це, скоріше, правило, ніж виняток.

Недоліками цього методу іноді називають нестабільність прогнозу та низьку точність, але це не завжди підтверджуються. За своєю природою дерева використовують «наївний підхід» у певному сенсі, починаючи з припущення, що ознаки незалежні одна від одної. Таким чином, моделі дерева регресії є статистично найбільш ефективними, коли набір змінних, що аналізуються, не надто мультиколінеарний.

1.5. Теорія випадкових лісів для задач регресії

Випадковий ліс — це один із алгоритмів машинного навчання, який можна використовувати для вирішення різних проблем, таких як регресія, класифікація, кластеризація, вибір функцій тощо.

Перша робота Лео Бреймана та Адель Катлер, пов'язана з побудовою ансамблів дерев рішень, була опублікована на початку 1990-х років. Їх робота заснована на використанні евристичних процедур. Також використовувався підхід, заснований на бустінгу дерев рішень. Більш строгий підхід, названий методом випадкового підпростору, був розроблений у роботі Тін Кам Хо [4]. Суть методу полягає у використанні лише фіксованої частки випадково вибраних ознак для побудови кожного дерева. Хо назвав ансамбль побудованих дерев лісом випадкових рішень.

Пізніше, Лео Брейман запропонував метод, відомий як випадковий ліс [7]. Визначення, дане ним, є досить загальним: випадковим лісом називається класифікатор, що складається з набору дерев $\{h(x, \Theta k), k = 1,...\}$, де Θk – незалежні однаково розподілені випадкові вектори й кожне дерево вносить один голос при визначенні класу x.

Метод заснований на побудові великої кількості дерев рішень (один із параметрів методу), кожне з яких будується на вибірках, отриманих із початкових навчальних вибірок за допомогою процедури бутстрепу (тобто зворотної вибірки).

Метод випадкових лісів має такі *переваги*:

– гарантований захист від перенавчання, навіть якщо кількість ознак перевищує кількість спостережень. Ця властивість є унікальною серед інших методів машинного навчання і надзвичайно цінна для вирішення прикладних задач;

– щоб побудувати випадковий ліс на основі навчальної вибірки, вам потрібно лише встановити два параметри: розмір ансамблю та кількість випадково вибраних ознак, після чого відбувається розбиття на вершини відповідно до цих параметрів;

– метод Out-Of-Bag (OOB), запропонований Брейманом [8], забезпечує оцінку ймовірності помилкового прогнозування випадкового лісу на основі спостережень, які не включені в навчальні початкові зразки, але використані для побудови дерева;

– випадкові ліси можна використовувати не тільки для завдань прогнозування та регресії, але також для таких завдань, як ідентифікація найбільш інформативних ознак, кластеризація, вибір аномальних спостережень та визначення прототипів класів;

– метод допускає легку паралелізацію (тобто програмну реалізацію, придатну для паралельних обчислень), що має велике значення при великому розмірі навчальної вибірки;

– навчальні вибірки для побудови випадкових лісів можуть містити ознаки, виміряні в різних шкалах: числовій, порядковій та номінальній, що є неприйнятним для багатьох інших методів;

– метод дозволяє легко розпаралелювати процеси прогнозування, що важливо, коли розмір навчальної вибірки великий;

Крім описаних вище переваг методу випадкового лісу, можна виділити особливості, які сприяють поширенню методу:

1. Для випадкового лісу використовується подвійна ін'єкція випадковості: баггінг та метод використання випадкового підпростору при розбитті кожної вершини. В результаті точність методу значно підвищується
2. Нескладність організувати паралельні обчислення
3. Невелика кількість параметрів методу гарантує зручність використання
4. Відсутній процес обрізки дерева, що призводить до збільшення обчислювальної ефективності.
5. Відсутня проблема перенавчання.

1.5.1. Інформативні ознаки випадкових лісів

Найбільш використовуваною сферою застосування випадкових лісів (крім регресії та класифікації) є задача виділення найбільш інформативних ознак. Брейманом було запропоновано 4 міри інформативності ознак [7].

Нехай x_i – деяка ознака. Три перші міри інформативності засновані на оцінці впливу випадкової перестановки значень цієї ознаки в OOB-вибірках на результати прогнозуваня.

Перша міра обчислюється таким чином:

1) побудувати випадковий ліс і отримати оцінку ймовірності помилкового прогнозування e методом OOB;

2) у OOB-вибірках для кожного дерева з побудованого випадкового лісу зробити випадкову перестановку значень ознаки x_i;

3) отримати оцінку ймовірності помилкового прогнозування $\hat{e}_i$ по модифікованим OOB-вибіркам;

4) визначити інформативність ознаки x_i, як

$$I_1(x_i) = \max(0, \hat{e}_i - e_i). \quad (1.2)$$

Друга і третя міри використовують поняття відступу (margin). Нехай (x, y) — елемент навчальної вибірки. Відступ marg(x, y) визначається як визначається як різниця між часткою дерев в лісі, що правильно передбачаються, (x, y) і максимумом з часткою дерев, які відносять (x, y) в інші класи. В результаті перестановки значень ознаки x_i відступ зменшується. Мірою інформативності береться середнє значення (за всіма спостереженнями) зменшення відступу, тобто

$$I_2(x_i) = \frac{1}{l}\sum_{j=1}^{l} 1\, marg(x_i, y_i) - marg_i(x_i, y_i) \quad (1.3)$$

де $marg_i(x, y)$ означає відступ, обчислений за OOB-вибіркою з випадковою перестановкою значень ознаки x_i .

Міра 3 дорівнює різниці між кількістю відступів, які зменшилися, і кількістю відступів, які збільшилися в результаті перестановки значень ознаки x_i:

$$I_3(x_i) = [marg(x, y) > marg_i(x, y)] - [\, marg(x, y) < marg_i(x, y)\,] \quad (1.4)$$

Міра 4 визначається як середнє зменшення забрудненості вершин, обумовлене даними ознакою, а саме

$$I_4(x_i) = \frac{1}{k}\sum_{j=1}^{l} \Delta i(t)\, I(i, t), \quad (1.5)$$

де підсумовування здійснюється за всіма вершинами дерев випадкового лісу, $\Delta i(t)$— зменшення забрудненості в вершині t, а $I(i, t)$ — індикаторна функція, яка дорівнює 1, якщо ознака x_i була обрана для розщеплення в вершині t.

Незважаючи на численні дослідження як теоретичного, так і експериментального характеру [18], поки невідомо, яка з запропонованих мір краще в тій чи іншій ситуації. В роботі Бреймана [12] обговорюються деякі питання, що стосуються відмінності між мірами інформативності та наведено ряд прикладів їх застосування. Детальний список питань, пов'язаних із мірами інформативності, міститься в роботі [18].

Графік приватної залежності

Для випадкових лісів регресії можлива побудова графіків приватних залежностей відгуку від деяких незалежних змінних [13].

Нехай X_S — підвектор вхідних ознак $x_1, x_2, \dots, x_n$ з індексами з множини $S \subset \{1, 2, \dots, n\}$ та C — додаткова йому множина $S \cup C = \{1, 2, \dots, n\}$. Оцінка функції приватної (маргінальної) залежності відгуку від значень підвектора вхідних ознак X_S визначається як

$$f_S(x_s) = \frac{1}{l} \sum_{i=1}^{l} f_S(x_s, x_{iC}). \quad (1.6)$$

Ця функція і відповідний графік дозволяють виділити ефект впливу підвектора вхідних ознак X_S на відгук після видалення ефекту впливу X_C.

Побудова матриці близькості

Побудова випадкового лісу може бути використана як проміжний етап у побудові матриці близькості (proximity) спостереження [14]. Елемент (i, j) матриці близькості дорівнює частці дерев у лісі, таких, що елементи x_i та x_j класифікуються в одну термінальну вершину (лист). Зрозуміло, що елементи, які часто потрапляють в ту саму кінцеву вершину, є «схожими» (близькими) у певному сенсі. Матрицю близькості можна використовувати для:

1) кластеризації спостережень;

2) багатомірного шкалювання;

3) знаходження прототипів класів;

4) виявлення аномальних спостережень.

Кластеризація спостережень здійснюється за допомогою генерації штучних даних. Перший клас формується початковими вибірками, які підлягають кластеризації, а спостереження другого класу створюються штучно. Генерацію можна здійснити двома способами:

– Створення незалежних початкових бутстрепів для кожної ознаки;

– Значення кожної ознаки генерується окремо на основі рівномірного розподілу її значень у вихідній вибірці.

Після формування таких двокласових вибірок будується випадковий ліс. Суть цього методу полягає в тому, що близькі точки вихідних зразків мають тенденцію потрапляти в ті самі кінцеві вершини, тому матрицю близькості можна використовувати для кластеризації. Деякі теоретичні результати такого підходу можна знайти в [20], а практичні приклади його використання в [14].

Визначення прототипів класів проводиться таким чином[14]: для кожного класу серед k його найближчих сусідів (визначених матрицею близькості) знаходиться спостереження, яке має найбільшу кількість спостережень цього класу. Потім для кожної ознаки знаходиться медіана цих k найближчих сусідів. Ці медіани розглядаються як прототипи для розглянутих класів. Міра аномалії для спостереження визначається як величина, обернена до суми квадратів близькості між цим спостереженням та іншими спостереженнями такого ж типу.

1.5.2 Різновиди випадкових лісів

Існує кілька типів випадкових лісів:

1) *випадковий ліс виживання*. Узагальнення випадкових лісів до правоцензурованих даних, важливим прикладом яких є дані про виживання, називається Random Survival Forest. З їх допомогою можна побудувати

непараметричні регресійні криві виживання, які не передбачають пропорційності ризиків.

2) *ліс квантильної регресії.* У випадку безперервних відповідей випадкові ліси є надійним способом побудови непараметричних регресій, через те, що вони забезпечують хороше наближення умовної середньої реакції. У роботі [20] запропоновано важливе узагальнення – квантильні регресійні ліси, які дають більш повну картину умовного розподілу відгуку, тобто можливість непараметричних оцінок квантилів умовного розподілу. Це особливо дозволяє використовувати їх для побудови довірчих інтервалів і виявлення викидних значень відповіді.

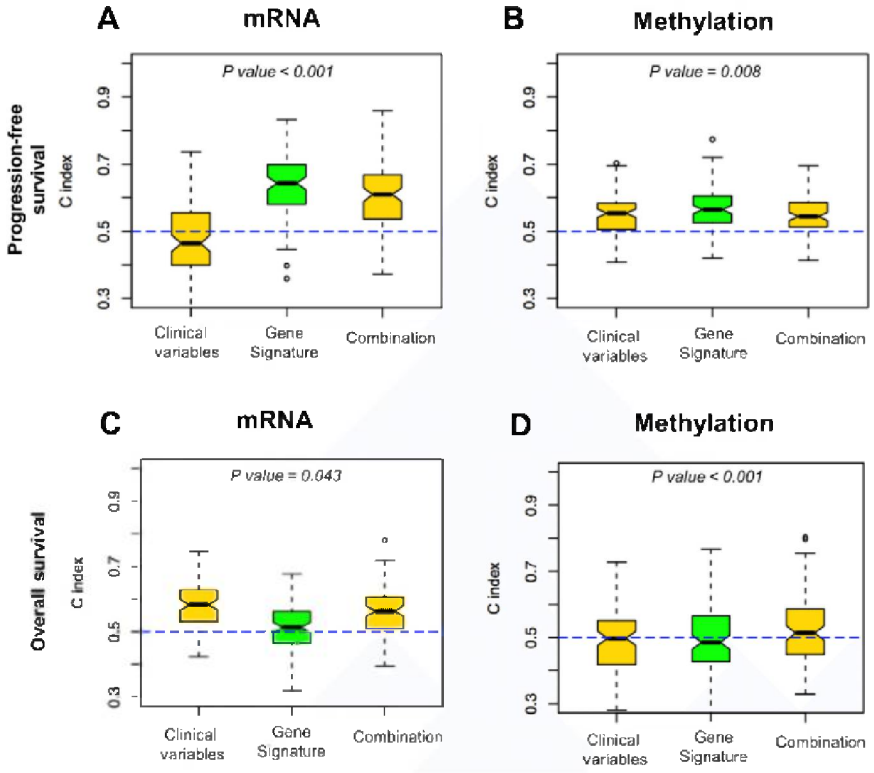

Рис. 1.2 – Модель випадкового лісу виживання.

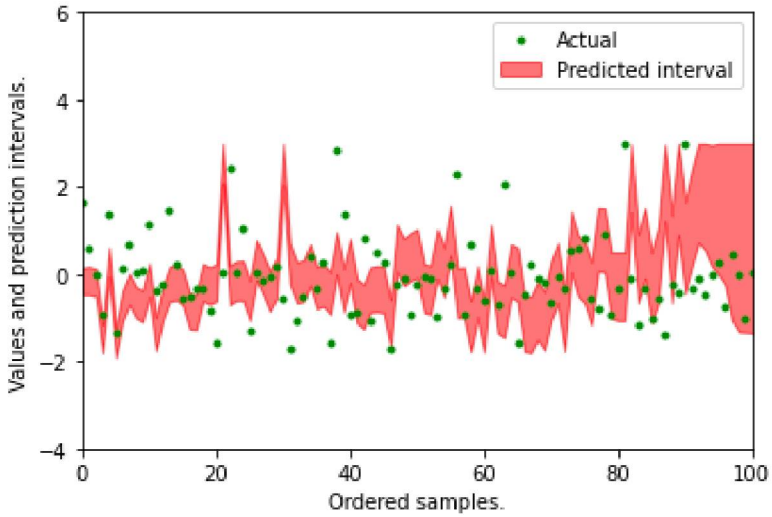

Рис. 1.3 – Регресія квантильних лісів.

3) *логічний ліс*. В роботі [21] запропоновано метод побудови ансамблю класифікаторів на основі логістичних дерев рішень. Теоретичною основою є логістична регресія [9], яка передбачає, що всі ознаки, включаючи категоріальні ознаки, є бінарними. Прогнозування відгуку в логістичній регресії виконується з використанням логістичних комбінацій бінарних ознак, що дозволяє будувати гнучкі моделі взаємодії складних ознак. Однак за наявності шуму точність роботи логістичної регресії зменшується. Окрім класифікації, логічні ліси можна використовувати для визначення найбільш інформативних ознак та їх взаємодії.

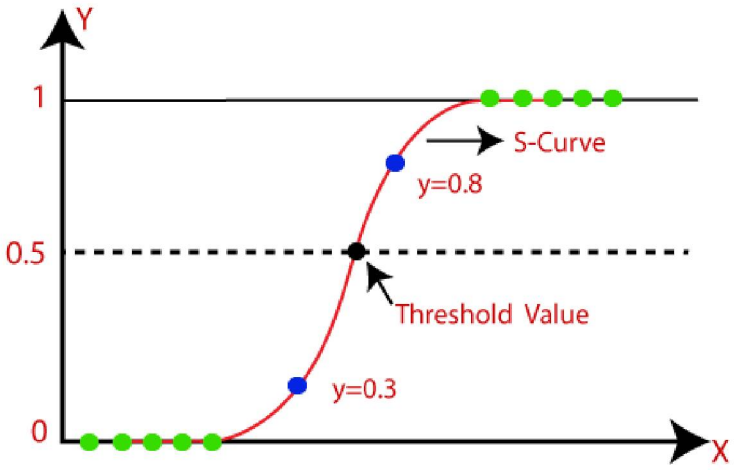

Рис. 1.4 – Приклад логічної регресії.

4) *випадковий наївний байєсівський класифікатор і випадкова поліноміальна логіт-модель*. Експериментальні дослідження та практика з випадковими лісами показали, що метод має багато хороших якостей - високу точність, стійкість до шуму в тренувальній виборці та відсутність перенавчання. У зв'язку з цим можна припустити, що основні елементи алгоритму Бреймана – підвибірка, сформована методом бегінгу та випадковий вибір ознак на верхній стадії дерева рішень про розщеплення - можуть бути успішно використані в якомусь іншому базовому класифікаторі. Ця ідея описана та реалізована в роботі [10]. У цій роботі базовими класифікаторами розглядаються наївні байєсовські класифікатори та мультиноміальні логістичні моделі. Автори провели серію експериментів, порівнюючи точність класифікації відповідних ансамблів з точністю випадкових лісів і опорних векторних машин, показавши, що випадковий наївний байєсівський класифікатор та випадкова поліноміальні логіт-моделі перевершують ці методи в деяких випадках.

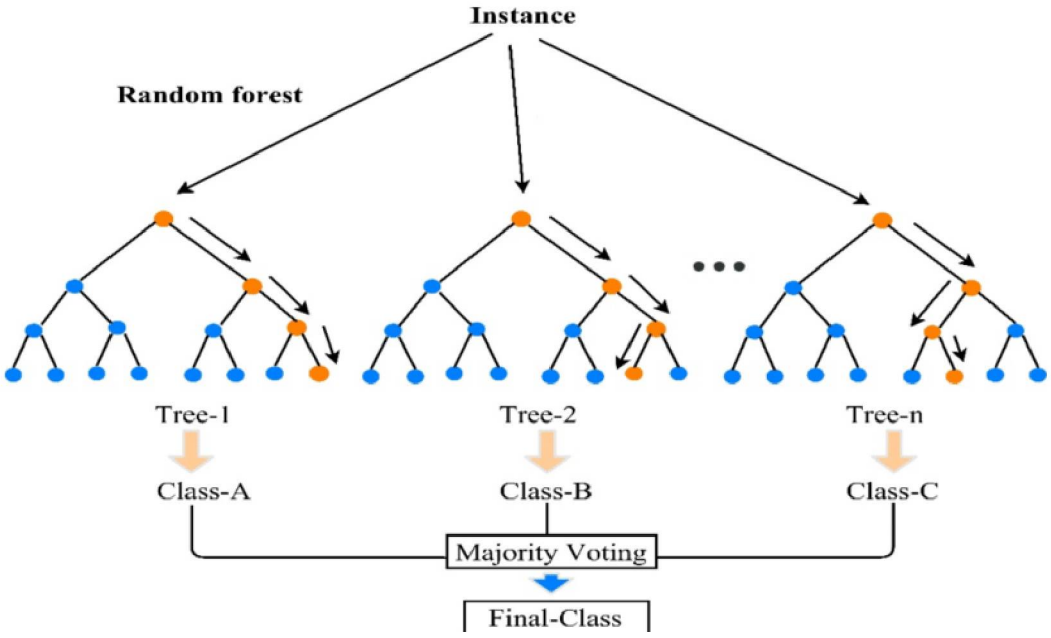

Рис. 1.5 – Випадковий наївний байєсівський класифікатор.

1.6 Лінійна регресія

Лінійна регресія — це метод машинного навчання, задача якого моделювання зв'язку між скалярною змінною y та векторною (зазвичай) змінною X. Якщо змінна X також є скаляром, це називається простою регресією. Лінійна регресія - це широко використовувана модель, що використовується для оцінки реальних величин.

У лінійній регресії зв'язок між незалежною та залежною змінними розглядається через лінію, яка зазвичай представляє зв'язок між двома змінними.

Загальна модель лінійної регресії виглядає так:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u, \tag{1.7}$$

де y — залежна змінна, (x_1, x_2, x_k) — незалежні змінні, u — випадкова помилка, розподіл якої зазвичай залежить від незалежної змінної, але математичне очікування якої дорівнює нулю.

Відповідно до цієї моделі математичне сподівання залежної змінної є лінійною функцією незалежної змінної:

$$E(y) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u. \tag{1.8}$$

Вектор параметрів $(\beta_0 + \beta_1 + \beta_k)$ – невідомий, і саме у підборі параметрів на основі деяких тестових значень y та (x_1, x_2, x_k) і полягає задача лінійної регресії. Тобто для деяких n експериментів повинні бути відомими значення $(x_{i1} \ldots, x_{iK})\,{}_{i=1}^{n}$ незалежних змінних і відповідні їм значення y_i залежної змінної.

Відповідно до класичної моделі до специфікацій моделі та відомих експериментальних даних додаються такі вимоги:

1) $\forall i \neq j \;\; E(u_i u_j | x_i) = 0$ (залишки повинні бути некорельовані);

2) $\forall i \;\; E(u_i^2 | x_i) = \sigma^2$ (гомоскедастичність);

3) ранг матриці X є K+1;

4) усі елементи матриці X є невипадковими;

5) також часто додається умова нормальності для випадкових відхилень, що дозволяє ширше аналізувати оцінки параметрів та їх значущість, хоча і не є обов'язковим для можливості використання, наприклад, методу найменших квадратів:

$$u_i | x_i \sim N(0, \sigma^2); \tag{1.9}$$

6) для асимптотичних властивостей оцінювання, коли розмірність матриці X прагне до нескінченності, необхідно виконати деякі додаткові умови. Одна з цих умов може полягати в тому, що існує межа, коли вимір прямує до нескінченності:

$$\lim_{n \to \infty} \lambda_{(X'X)} = \infty, \tag{1.10}$$

де $\lambda_{\_}$ позначає найменше власне значення матриці.

Залежно від того, що досліджується за допомогою лінійної регресії та конкретних цілей дослідження, для оцінки невідомих параметрів можна використовувати різні методи. Найбільш популярним є звичайний метод найменших квадратів. Він приймає як оцінку параметра значення, яке мінімізує суму квадратів залишків за всіма спостереженнями:

$$\hat{\beta} = \arg\min \sum_{i=1}^{n} |y_i - \beta_0 - \sum_{j=1}^{K} X_{ij} \beta_j|^2 = argmin||y - X\beta||^2. \tag{1.11}$$

Метод найменших квадратів можна застосувати до будь-якої задачі, де ранг матриці X дорівнює кількості її стовпців. Крім того, метод дає прості аналітичні вирази для оцінок параметрів:

$$\hat{\beta} = (X'X)^{-1} X' y. \tag{1.12}$$

У випадку класичної моделі лінійної регресії оцінка методом найменших квадратів є незміщеною, значущою та найкращою лінійною незміщеною оцінкою.

У ситуаціях, коли деякі умови класичної лінійної регресії не виконуються, метод найменших квадратів може бути не оптимальним. Тому для моделі узагальненої лінійної регресії найкращою лінійною незміщеною оцінкою є оцінка, отримана за так званим узагальненим методом найменших квадратів:

$$\hat{\beta} = (X^T W^{-1} X)^{-1} X^T W^{-1} y. \tag{1.13}$$

Узагальнений метод найменших квадратів також отримується шляхом мінімізації деякої норми вектору зміщення:

$$\hat{\beta} = argmin(y - X\beta)^T W^{-1} (y - X\beta). \tag{1.14}$$

Інші методи оцінювання включають:

- Метод найменших модулів, що знаходить мінімум суми не квадратів відхилень, а їх абсолютних значень:

$$\hat{\beta} = \arg\min \sum_{i=1}^{n} |y_i - \beta_0 - \sum_{j=1}^{K} X_{ij}\beta_j|. \tag{1.15}$$

Цей підхід найкращий с точки зору максимальної ймовірності, коли відхилення мають розподіл Лапласа. Метод найменшого модуля набагато менш чутливий до викидів, ніж метод найменших квадратів, але він може мати кілька рішень і не має простої формули для визначення оцінки.

- Метод максимальної вірогідності. Використовується, коли відомі всі розподіли зміщення для всіх спостережень. Для лінійної регресії класична та узагальнена моделі з умовами нормальності, що відхиляються, дають ті самі результати, що й найменші квадрати та узагальнені найменші квадрати відповідно.
- Ортогональна регресія. Він використовується у випадках, коли значення пояснювальних змінних також можуть містити випадкові компоненти, а можливе зміщення всіх змінних розглядається в процесі оцінювання.

1.7 Бустинг дерев прийняття рішень

Бустинг – це метод, який використовується у машинному навчанні для зменшення кількості помилок при прогностичному аналізі даних. Фахівці по роботі з даними навчають програмне забезпечення з алгоритмами машинного навчання, які називаються моделями машинного навчання, на розмічених даних, щоб зробити припущення про непомічені дані. Одна модель машинного навчання може робити помилки прогнозування залежно від точності навчального набору даних. Бустинг покращує точність прогнозування та продуктивність моделей шляхом перетворення слабких класифікаторів на єдину сильну модель навчання. Бустинг перетворює систему слабких моделей на єдиний сильний алгоритм навчання.

Бустинг будує ансамблі моделей шляхом послідовного поєднання кількох дерев рішень. Вихідним даним окремих дерев надаються ваги. Потім неправильним класифікаціям з першого дерева рішень надається більша вага,

після чого дані передаються в наступне дерево. Після численних циклів бустинг поєднує слабкі класифікатори в потужний алгоритм прогнозування.

Бустинг і бегінг — два поширені ансамблеві методи, що підвищують точність прогнозування. Основна відмінність між ними – метод навчання. У випадку бегінга фахівці з роботи з даними підвищують точність слабких моделей, паралельно навчаючи деякі з них на різних наборах даних. Бустинг навчає слабкі моделі послідовно.

Метод навчання залежить від типу бустингу, що називається алгоритмом. При цьому алгоритм виконує такі спільні кроки для навчання моделі бустингу:

1) алгоритм бустингу надає рівну вагу кожній вибірці даних. Він передає дані першої моделі, яка називається базовим алгоритмом. Для кожної вибірки даних базовий алгоритм робить прогнози;

2) алгоритм бустингу оцінює прогнози моделі та збільшує вагу вибірок зі значною помилкою. Також вага надається на основі продуктивності моделі. Модель із кращими прогнозами матиме великий вплив на остаточне рішення;

3) алгоритм передає зважені дані до наступного дерева рішень;

4) алгоритм повторює кроки 2 і 3 до тих пір, поки помилки навчання не опустяться нижче за певний поріг.

Нижче наведено три основні види бустингу:

Адаптивний бустинг (AdaBoost) – одна з найперших моделей бустингу. Він адаптується та самостійно коригує класифікатори в кожній ітерації бустингу.

AdaBoost спочатку надає однакову вагу кожному набору даних. Потім він автоматично коригує ваги точок вибірки після кожного кроку на дереві рішень. Елементи, які були класифіковані невірно, набувають більшої ваги у наступній ітерації. Процес повторюється, поки залишкова помилка чи різниця між фактичними та прогнозованими значеннями не опуститься нижче допустимого рівня.

AdaBoost можна використовувати з багатьма предикторами. Крім того, він менш чутливий, ніж інші алгоритми бустингу. AdaBoost не такий ефективний при кореляції між ознаками або використанням даних великої розмірності. В цілому, AdaBoost справляється із задачами класифікації.

Градієнтний бустинг (GB) схожий на AdaBoost: він також є методом послідовного навчання. Різниця між AdaBoost та GB у тому, що GB не присвоює неправильно класифікованим елементам більшу вагу. Натомість програмне забезпечення GB оптимізує функцію втрат через послідовне генерування базових моделей, внаслідок чого поточна базова модель завжди стає ефективнішою за попередню. На відміну від AdaBoost, метод GB намагається одразу генерувати точні результати, а не виправляти помилки. Тому метод GB дає більш точні результати. Градієнтний бустинг підходить і задач класифікації, і регресії.

Екстремальний градієнтний бустинг (XGBoost) у різний спосіб покращує градієнтний бустинг, фокусуючись на швидкості обчислень та масштабах моделі. XGBoost розроблено для ефективної багатоядерної паралельної обробки прямо під час навчання. Цей алгоритм бустингу є ефективним інструментом роботи з великими даними. Ключовими особливостями XGBoost є розпаралелювання, розподілені обчислення, оптимізація кешу та зовнішні обчислення.

Основними перевагами бустингу є:

1) простота реалізації. Бустинг має прості для розуміння та інтерпретації алгоритми, здатні вчитися на своїх помилках. Ці алгоритми вимагають попередньої обробки даних, і навіть мають вбудовані процедури обробки відсутніх значень. Крім того, більшість мов вбудовані бібліотеки для реалізації алгоритмів бустингу з безліччю параметрів, що дозволяють точно задавати продуктивність;

2) зменшення зміщення. Зміщення – це наявність невизначеності чи неточності у результатах машинного навчання. Алгоритми бустингу

поєднують кілька слабких моделей у послідовний метод, що ітеративно покращує спостереження. Такий підхід допомагає зменшити високе усунення, яке поширене в моделях машинного навчання;

3) ефективність алгоритмів. Алгоритми бустингу фокусуються на елементах, що підвищують точність прогнозування під час навчання. Вони здатні зменшувати кількість атрибутів даних та ефективно обробляти великі набори.

Нижче наведено основні недоліки бустингу:

1) вразливість до викидів у даних.

Моделі бустингу вразливі до викидів чи значень даних, що відрізняється від інших наборів даних. Викиди можуть суттєво спотворювати результати, оскільки кожна модель намагається виправити помилки попередньої.

2) реалізація у режимі реального часу.

Оскільки цей алгоритм складніший за інші процеси, при реалізації бустингу в режимі реального часу можуть виникнути труднощі. Бустинг виявляє високу адаптивність, тому можна використовувати різноманітні параметри моделі, які безпосередньо впливають на її продуктивність.

Висновки до розділу 1

У розділі була сформульована задача регресії, наведено приклади із прикладних областей, де може знадобитися вирішення задачі регресії. Також була сформована задача прогнозування, та визначено, як вона пов’язана с задачею регресії. А також обумовлена необхідність та значимість прогнозування трафіку комп’ютерної мережі. Після цього були розглянуті основні методи для побудови регресійних моделей. Крім огляду особливостей побудови моделі кожного алгоритму, було обгрунтовано їхні переваги та недоліки.

Також був детально розглянутий метод випадкових лісів. Визначенні головні переваги методу та його особливості, що сприяли його поширенню в використанні. Наведений алгоритм побудови випадкових лісів і їх використання. Були визначені інформаційні ознаки випадкових лісів та міри інформативності. Наостанок, була приведена характеристика видів випадкових лісів.

Були детально розглянуті методи лінійної регресії а також метод бустингу дерев прийняття рішень. Приведений порівняльний аналіз з методом випадкових лісів.

На основі проведеного аналізу методів випадкових лісів, лінійної регресії, а також методу градієнтного бустингу дерев було прийняте рішення використати дані методи для розв'язання задачі прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання.

РОЗДІЛ 2
МЕТОД ПРОГНОЗУВАННЯ ТРАФІКУ КОМП'ЮТЕРНИХ МЕРЕЖ НА ОСНОВІ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

2.1 Алгоритми машинного навчання для розв'язання задачі прогнозування стану систем

Розглянемо прикладні алгоритми побудови методів, які будуть використовуватися для побудови моделі прогнозування трафіку комп'ютерних мереж.

2.1.1 Алгоритм побудови випадкового лісу

Алгоритм індуктивної побудови й використання випадкового лісу вперше був розроблений Л. Брейманом і А. Катлер і реалізований в ряді комерційних пакетів. Алгоритм побудови випадкового лісу може бути представлений таким чином:

1. Для i = 1,2,...,B (де B — кількість дерев в ансамблі) виконати:

- сформувати бутстреп вибірку S розміру l по початковій навчальній вибірці

$$D = \{x_i, y_i\}_{i=1}^{l}. \quad (2.1)$$

- по бутстреп вибірці S викликати не усічене дерево рішень T_i з мінімальною кількістю спостережень в термінальних вершинах рівним n_{min}, рекурсивно виконуючи наступний алгоритм:

a) з початкового набору n ознак випадково вибрати p ознак;

b) з p ознак вибрати ознаку, яка забезпечує найкраще розщеплення;

c) розщепити вибірку, відповідну оброблюваній вершині, на дві підвибірки;

2. В результаті виконання кроку 1 отримуємо ансамбль дерев рішень

$$\{T_i\}_{i=1}^{B}. \quad (2.2)$$

3. Прогнозування нових спостережень для регресії здійснювати так:

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{i=0}^{B} T_i\,(x) \quad (2.3)$$

2.1.2 Градієнтний бустинг дерев рішень

Градієнтний бустинг – це техніка машинного навчання для завдань класифікації та регресії, яка будує модель передбачення у формі ансамблю слабких моделей, що передбачають, зазвичай дерев рішень. Навчання ансамблю проводиться послідовно на відміну, наприклад, від беггинга. На кожній ітерації обчислюються відхилення передбачання уже навченого ансамблю на навчальній вибірці. Наступна модель, яка буде додана в ансамбль, буде передбачати ці відхилення. Таким чином, додавши передбачення нового дерева до передбачень навченого ансамблю, ми можемо зменшити середнє відхилення моделі, яке є ціллю оптимізаційної задачі. Нові дерева додаються в ансамбль доти, доки помилка зменшується, або доки виконується одне з правил "ранньої зупинки".

Розглянемо ілюстрацію бустингу. На ній розглядається поведінка моделі на одній точці абстрактного завдання лінійної регресії. Припустимо, перша модель ансамблю F завжди видає вибіркове середнє передбачуваної величини f_0. Таке прогнозування досить грубе, тому середньоквадратичне відхилення на обраній нами точці буде досить великим. Ми спробуємо це виправити, навчивши модель Δ_1, яка "коригуватиме" передбачення попереднього ансамблю F_0. Таким чином ми отримаємо ансамбль F_1, передбачення якого підсумовуватиметься з передбачень моделей f_0 та Δ_1. Продовжуючи таку

послідовність ми приходимо до ансамблю F_4, передбачення якого підсумовується з передбачень f_0, Δ_1, Δ_2, Δ_3, Δ_4 і передбачає в точності значення заданої цілі.

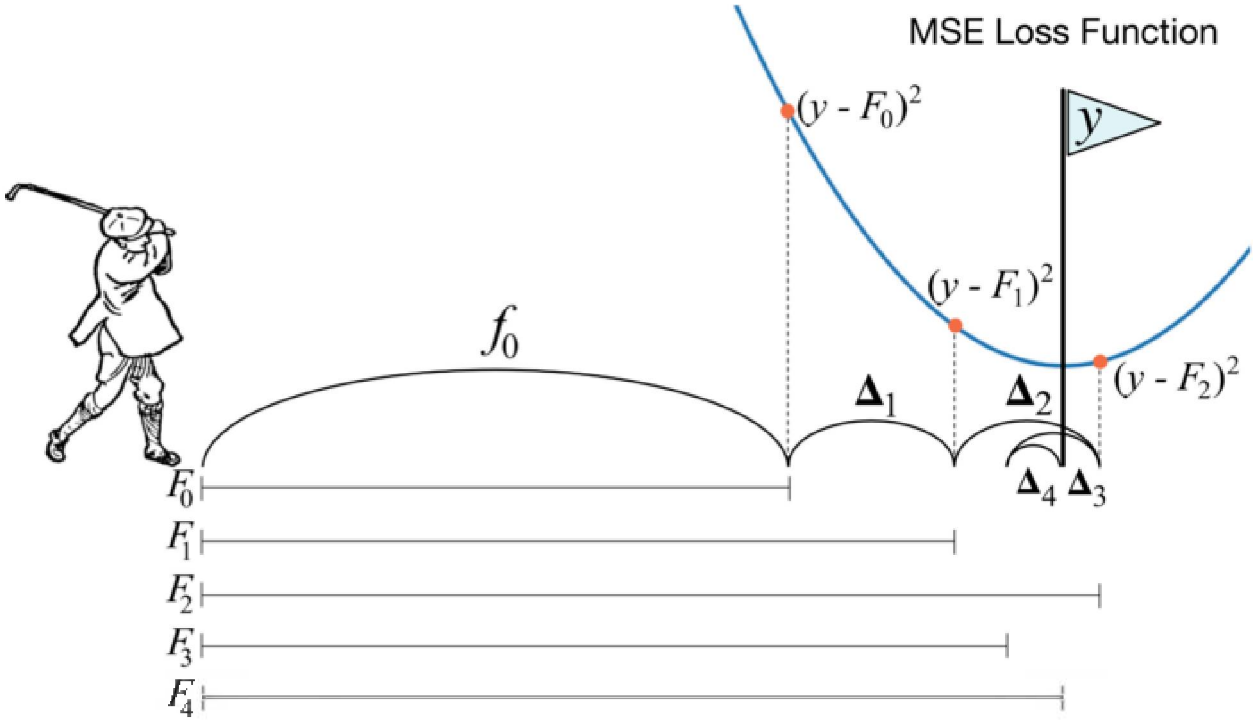

Рис. 2.1 – Іллюстрація бустингу.

Математично алгоритм можна описати так. Задана функція для оптимізації градієнтного бустингу:

$$L^{(t)} = \arg\min \sum_{i=1}^{n} l\left(y_i \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega\left(f_t\right) \quad (2.4)$$

де l – функція втрат;

$y_i \hat{y}_i^{(t)}$ – значення і-го елемента навчальної вибірки та сума передбачень перших t дерев відповідно;

x_i – набір ознак і-го елемента навчальної вибірки;

f_t – функція (у нашому випадку дерево), яку ми хочемо навчити на кроці t;

$f_t(x_i)$ – передбачення на і-му елементі навчальної вибірки;

$\Omega(f)$ – регулювання функції.

Далі за допомогою розкладання Тейлора до другого члена можемо наблизити функцію $L^{(t)}$, що оптимізується, наступним виразом:

$$L^{(t)} = \arg\min \sum_{i=1}^{n} l\left(y_i \hat{y}_i^{(t-1)} + g_i f_t(x_i) + 0.5 h_i f_t^2(x_i)\right) + \Omega(f_t) \quad (2.5)$$

де

$$g_i = \frac{\partial l(y_i \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}, \quad (2.6)$$

$$h_i = \frac{\partial^2 l(y_i \hat{y}_i^{(t-1)})}{\partial^2 \hat{y}_i^{(t-1)}}. \quad (2.7)$$

Оскільки мінімізується помилка моделі на навчальній вибірці, потрібно знайти мінімум $L^{(t)}$ для кожного *t*.

Мінімум цього виразу щодо $f_t(x_i)$ знаходиться у точці $f_t(x_i) = -\frac{g_i}{h_i}$

Кожне окреме дерево ансамблю $f_t(x_i)$ навчається стандартним алгоритмом.

2.1.3 Матричне представлення лінійної регресії

Нехай задана вибірка розміром M спостережень змінних y та x. Позначимо через t – номер спостереження у вибірці. Тоді y_t – значення змінної y в t-му спостереженні, x_{tj} – значення j-го фактору в t-ому спостереженні. Відповідно, $x_t^T = (x_1, x_2, \dots, x_{tk})$ – вектор регресорів в t-му спостереженні. Тоді лінійна регресійна залежність має місце у кожному спостереженні:

$$y_t = b_1 x_{t1} + b_2 x_{t2} + \cdots + b_k x_{tk} = \sum_{j=1}^{k} b_j x_{tj} = x_t^T b + \varepsilon_t. \quad (2.8)$$

Введемо позначення:

$y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_M \end{pmatrix}$ – вектор спостережень залежною змінної y;

$X = \begin{pmatrix} x_{11}x_{12}x_{13}x_{1k} \\ x_{21}x_{22}x_{23}x_{2k} \\ \dots\ \dots\ \dots\ \dots \\ x_{M1}x_{M2}x_{M3}x_{Mk} \end{pmatrix}$ – матриця факторів;

$\varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_M \end{pmatrix}$ – вектор випадкової помилки.

Тоді модель лінійної регресії можна задати у матричної формі:

$$y = Xb + \varepsilon. \tag{2.9}$$

2.2 Метрики оцінювання ефективності моделей і методів машинного навчання

Завдяки матриці похибок (confusion matrix) можна оцінити точність роботи моделі. Існує два типи помилок: перший рід (False Positives – FP) і другий рід (False Negatives – FN). Матриця похибок дозволяє візуально оцінити їх кількість і розподіл. Схема такої матриці показана на малюнку 1. 2.2.

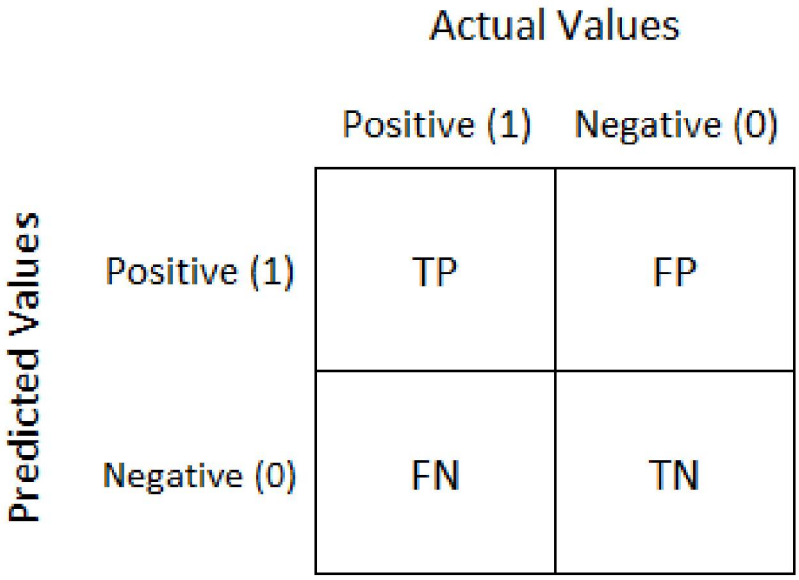

Рис. 2.2 – Схема матриці похибок.

Для класифікаторів існують метрики, які відображають справжню ефективність роботи, і їх вибір є важливим завдання під час розробки моделі[19].

Існують такі метрики:

1) accuracy (частка правильних відповідей);
2) recall (повнота);
3) precision (точність);
4) F_1 метрика;
5) Area under curve (AUC – площа під кривою похибок).

Метрика *accuracy* відображає частку правильних відповідей моделі, однак він не забезпечує достатніх оцінок для класів, які нерівномірно розподілені, оскільки класифікація об'єктів у класі, що найчастіше зустрічаються, має вищу точність. Через цей недолік дана метрика майже не використовується, але її значення може бути розраховане за наступною формулою:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (2.10)$$

Метрики *recall*, *precision*, F_1 тісно пов'язаними між собою, тому їх можна розглядати разом. Перша характеризує долю позитивних об'єктів, що було знайдено. Друга своєю чергою визначає долю об'єктів, що були охарактеризовані позитивно та дійсно такими є. Вони обчислюються за такими формулами:

$$recall = \frac{TP}{TP + FN}. \tag{2.11}$$

$$precision = \frac{TP}{TP + FP}, \tag{2.12}$$

Головна відмінність від accuracy полягає в тому, що ці метрики можна використовувати при незбалансованій кількості класів, оскільки розглядаються тільки значення, що належать до одного класу. Щоб отримати значення для всієї вибірки, обчислюється середнє значення для всіх класів об'єктів. Однак зрозуміло, що коли кількість правильно класифікованих об'єктів (precision) збільшується, загальна частка позитивних об'єктів (recall) зменшується, оскільки зростає вимога до правильної класифікації. Щоб мати можливість враховувати обидва значення одночасно, слід ввести додаткову метрику. Її роль й виконує F_1, яка використовує середнє гармонічне між precision та recall, а обчислюється наступним чином:

$$F_1 = 2\frac{recall * precision}{precision + recall}. \tag{2.13}$$

Щоб інтерпретувати ці значеня, необхідно ввести деякий поріг (threshold), який буде використовуватися для перетворення результатів роботи моделі у бінарний формат. Однак при спробі змінити цей поріг під час фактичного використання моделі вищевказані метрики стають марними. Для

уникнення цього недоліку, як варіант, можна використати метрику AUC. Отримане значення буде характеризувати якість моделі у загальному випадку. Ця метрика характеризує значення під Receiver operating characteristic (ROC) кривою, що своєю чергою відображає зміну recall при зміні False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN}. \tag{2.14}$$

Зі збільшенням FPR, тобто зменшенням порога, який відмежовує значення, буде зростати як і FPR, так і recall. Тобто, AUC дає оцінку здатності моделі працювати при високому порозі. З добре натренованою моделью, ця метрика одразу буде мати високий recall та низьку FPR. Приклад ROC кривої наведений нижче на рис. (2.3) [22].

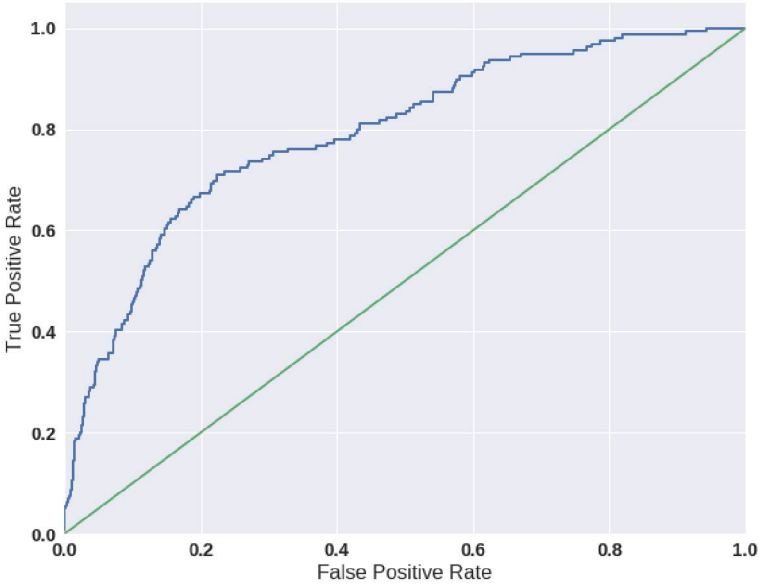

Рис. 2.3 – ROC крива.

Найбільш поширеними метриками для оцінки якості в задачах регресії є:

1) *середня квадратична помилка* (англ. Mean Squared Error, MSE)[15]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(a(x_i) - y_i)^2. \quad (2.15)$$

MSE використовується в ситуаціях, коли нам потрібно виділити великі помилки та вибрати моделі, які дають менші помилки передбачення. Оскільки ми зводимо помилку прогнозу в квадрат, загальна помилка стає більш помітною. І модель, яка дає нам менше значення середньоквадратичної похибки, має меншу грубу похибку.

2) *середня абсолютна помилка* (англ. Mean Absolute Error, MAE)[16]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|a(x_i) - y_i|. \quad (2.16)$$

Середньоквадратична функція штрафує великі відхилення більше, ніж абсолютне середнє. При використанні будь-якого з цих двох метрик може бути корисно проаналізувати, які об'єкти найбільше вносять загальну помилку – ці об'єкти можуть бути помилковими під час обчислення функцій або цільових значень. Середньоквадратична помилка корисна для порівняння двох моделей або для контролю якості під час навчання, але вона не дозволяє нам зробити висновки про те, наскільки добре дана модель вирішує проблему.

3) *коефіцієнт детермінації*:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(a(x_i)-y_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}. \quad (2.17)$$

Коефіцієнт детермінації вимірює дисперсію, пояснену моделлю, як частку загальної дисперсії цільової змінної. Насправді цей показник якості є нормалізованою середньоквадратичною помилкою. Якщо він близький до 1, модель добре пояснює дані, якщо він близький до 0, якість прогнозу можна порівняти з постійним прогнозом

4) *середня абсолютна процентна помилка* (англ. Mean Absolute Percentage Error, MAPE):

$$MAPE = 100\% * \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - a(x_i)|}{|y_i|}. \quad (2.18)$$

Це безрозмірний коефіцієнт з простою інтерпретацією. Його можна виміряти в частках або відсотках.

5) *симетрична MAPE* (англ. Symmetric MAPE, SMAPE)

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{2 * |y_i - a(x_i)|}{|y_i| + |a(x_i)|}. \quad (2.19)$$

6) *середня абсолютна масштабована помилка* (англ. Mean absolute scaled error, MASE)[17]:

$$MASE = \frac{\sum_{i=1}^{n}|Y_i - e_i|}{\frac{n}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}. \quad (2.20)$$

MASE є дуже хорошим вибором для обчислення точності, оскільки сама помилка не залежить від масштабу даних і є симетричною: тобто позитивні та негативні відхилення від істини розглядаються однаково. В MASE маємо справу з двома сумами: одна в чисельнику, що відповідає тестовій вибірці, і одна в знаменнику, що відповідає навчальній вибірці. Друга – фактично середня абсолютна помилка прогнозу. Він відповідає середньому абсолютному відхиленню ряду перших різниць. Насправді це значення показує, наскільки навчальні вибірки є передбачуваними. Воно може дорівнювати нулю тільки в тому випадку, якщо всі значення в навчальній вибірці рівні між собою, що відповідає відсутності змін в ряді даних, що майже неможливо на практиці. Крім того, якщо ряд спадний тренд, його перша різниця коливатиметься навколо певного фіксованого рівня.

Звичайно, все це є явною перевагою MASE, оскільки вона дозволяє підсумовувати різні значення з різних рядів і отримувати неупереджену оцінку.

Недоліком MASE є те, що його важко інтерпретувати. Наприклад, MASE=2,15 насправді нічого не означає. Це просто означає, що похибка прогнозу в 2,15 рази більша, ніж середнє абсолютне відхилення першого різницевого ряду, не більше.

2.3. Ковзний контроль методів машинного навчання

Ковзний контроль(крос-валідація) використовується для оцінки середньої помилки на об'єктах контрольної вибірки. Для цього фіксується деяка множина розбиття вхідної вибірки на дві підвибірки: навчальну і контрольну. По кожному розбиттю виконується настройка алгоритму за навчальною підвибіркою, потім оцінюється його середня помилка на об'єктах контрольної підвибірки. Середнє значення помилок контрольної підвибірки по всім розбиттям називається оцінкою ковзною контролю.

При незалежній виборці середня помилка ковзного контролю дає незміщену оцінку ймовірності помилки. Це вигідно відрізняє її від середньої помилки на навчальній вибірці, яка може виявитися зміщеною (оптимістично заниженою) оцінкою ймовірності помилки, що пов'язано з явищем перенавчання.

Розглядається задача навчання з учителем:

1. Задана кінцева вибірка прецедентів:

$$X^L = (x_i, y_i)_i^L = 1 \subset X^* Y, \tag{2.15}$$

де X — множина описів об’єктів, Y — множина допустимих відповідей

2. Заданий алгоритм навчання – відображення μ – яке довільній кінцевій виборці прецедентів X^m ставить у відповідність функцію (алгоритм) $a: X \to Y$.

3. Якість алгоритму a оцінюється по довільній вибірці прецедентів X^m за допомогою функціоналу якості $Q(a, X^m)$. Для процедури ковзного контролю не важливо, як саме обчислюється цей функціонал. Як правило, він адитивний по об'єктах вибірки

$$Q(a, X^m) = \frac{1}{m}\sum_{x_i \in X^m} L(a(x_i), y_i), \tag{2.16}$$

де $L(a(x_i), y_i)$ – невід'ємна функція втрат, що повертає величину помилки відповіді алгоритму $a(x_i)$ при правильній відповіді y_i.

4. Для оцінки ковзного контролю ініціюється процедура ковзного контролю: вибірка X^L розбивається N способами на дві непересічні підвибірки

$$X^L = X_n^m \cup X_n^k, \tag{2.17}$$

де X_n^m – навчальна підвибірка довжиною m, X_n^k – контрольна підвибірка довжини

$$k = L - m, \tag{2.18}$$

$n = 1, \ldots, N$ – номер розбиття.

Для кожного розбиття n будується алгоритм $a_n = \mu(x_n^m)$ і обраховується значення функціонала якості:

$$Q_n = Q(a_n, X_n^k). \tag{2.19}$$

Середнє арифметичне значень Q_n по всім розбиттям називається оцінкою ковзного контролю:

$$CV(\mu, X^L) = \frac{1}{N}\sum_n^v Q\left(\mu(X_n^m), X_n^k\right). \quad (2.20)$$

Існують декілька видів ковзного контролю.

1. *Повний ковзний контроль (complete CV).*

Оцінка ковзного контролю будується за всіма $N = C_L^k$ розбиттями. Залежно від k (довжини навчальної вибірки) розрізняють:

– окремий випадок при $k = 1$ – контроль за окремими об'єктами (leave-one-out CV);

– загальний випадок при $k > 2$. Тут кількість N стає занадто великим навіть при порівняно малих значеннях k, через що ускладнюється практичне застосування. У цьому випадку повний ковзний контроль використовується або в теоретичних дослідженнях [23], або в тих рідкісних ситуаціях, коли для нього вдається вивести ефективну обчислювальну формулу. Наприклад, така формула відома для методу k–найближчих сусідів, що дозволяє ефективно вибирати параметр k. На практиці частіше застосовуються інші різновиди змінного контролю.

2. *Випадкові розбиття.*

Розбиття $n = 1, \dots, N$ обираються випадково та незалежно з множини усіх C_L^k розбиттів. Саме для цього випадку справедливі наведені вище оцінки довірчих інтервалів. В практичних задачах ці оцінки переносяться без змін і на інші способи розбиття вибірки.

3. Контроль на відкладених даних (hold-out CV).

Оцінка ковзного контролю будується по одному випадковому розбиттю, N = 1.

Цей підхід має очевидні недоліки:

– У контрольній підвибірці необхідно залишати занадто багато об’єктів. Зменшення довжини навчальної підвибірки призводить до необ’єктивної оцінки ймовірності помилки (песимістично завищеної);

– Оцінювання значною мірою залежить від помилок, але переважно лише характеризує алгоритм навчання;

– Оцінка має високу дисперсію, яку можна зменшити шляхом усереднення розподілу розбиття.

4. Контроль за окремими об'єктами (leave-one-out CV).

Є окремим випадком повного ковзного контролю при k = 1, відповідно, $N = L$. Це найпоширеніший варіант ковзного контролю.

Перевагою LOO є те, що кожен об'єкт лише один раз бере участь в контролі, а довжина навчальної підвибірки лише на одиницю менше довжини повної вибірки.

Проте суттєвим недоліком LOO є велика ресурсомісткість, оскільки навчатися доводиться L раз. Деякі методи навчання дозволяють достатньо швидко перенастроювати внутрішні параметри алгоритму при заміні одного навчального об'єкта іншим. У цих випадках обчислення LOO вдається помітно прискорити.

5. Контроль за q блокам (q-fold CV).

Вибірка випадковим чином розбивається на q непересічних блоків однакової (або майже однакової) довжини $k_1, \ldots, k_q$:

$$X^L = X_1^{k_1} \cup \ldots \cup X_q^{k_q} a, \qquad (2.21)$$
$$k_1 + \cdots + k_q = L.$$

Кожен блок по черзі стає контрольною підвибіркою, при цьому навчання проводиться за іншим q-1 блокам. Критерій визначається як середня помилка на контрольній підвибірці:

$$CV\left(\mu, X^L\right) = \frac{1}{q}\sum_{n=1}^{q} Q\left(\mu\left(X^L \setminus X_n^{k_n}\right), X_n^{k_n}\right). \quad (2.22)$$

Це компроміс між LOO, hold-out і випадковим розбиттям. З одного боку, навчання проводиться тільки q раз замість L. З іншого боку, довжина навчальних підвибірок, рівна $L\frac{q-1}{q}$ з точністю до округлення, не сильно відрізняється від довжини повної вибірки L. Зазвичай вибірку розбивають випадковим чином на 10 або 20 блоків.

6. Контроль при зростальній довжині навчання.

Саме цей метод використовується в задачах прогнозування. Навчальна підвибірка утворюється усіма *минулими* об'єктами $X_n^m = \{x_1, \ldots, x_n\}$. Контрольна підвибірка утворюється усіма *майбутніми* об'єктами $X_{n=}^k\{x_{n+\sigma+1}, \ldots, x_{n+\sigma+k}\}$, де $\sigma \geq 0$ – величина затримки прогнозування (зазвичай $\sigma = 0$). Момент «теперішнього часу» n «ковзає» по вибірці даних:

$$CV\left(\mu, X^L\right) = \frac{1}{T_2 - T_1 + 1}\sum_{n=T_2}^{T_2} Q\left(\mu\left(X_n^m\right), X_n^k\right), \quad (2.23)$$

де T_1 – мінімальна довжина навчальної вибірки, необхідна для нормальної роботи алгоритму навчання μ, $T_2 = L - \sigma - k$.

Оскільки довжина навчання m = n збільшується з часом, точність прогнозів може поступово поліпшуватися. Цей побічний ефект є небажаним, якщо ковзний контроль застосовується для оцінювання якості алгоритму навчання.

Недоліки ковзного контролю:

– задачу навчання необхідно вирішувати N разів. Це пов'язано зі значними обчислювальними витратами;

– спроби використати ковзний контроль для навчання як оптимізовани критерій, як правило, призводить до того, що він втрачає властивість незміщеності. Ця ситуація призводить до появи ризику перенавчання;

– ковзний контроль дає незміщену точкову, але не інтервальну оцінку ризику. В сьогоденні, на жаль, не існує методів побудови б точних довірчих інтервалів для ризику на основі ковзного контролю.

– оцінка ковзного контролю передбачає, що алгоритм навчання μ вже поставлено. У ньому нічого не сказано про те, якими властивостями повинні володіти «хороші» алгоритми навчання або як їх побудувати

На практиці, однак, ковзний контроль застосовується для оптимізації деяких ключових параметрів, які зазвичай визначають структуру або складність використовуваної моделі алгоритму, і кількість можливих значень відносно невелика. Наприклад:

– вибір моделі алгоритмів з невеликої множини альтернативних варіантів;

– оптимізація параметра регуляризації;

– оптимізація ширини вікна;

– вибір інформативного набору ознак;

– редукція вирішального дерева;

– структурна мінімізація ризику.

У результаті проведеного аналізу, було прийнято рішення використовувати процедуру крос-валідації для збільшення ефективності роботи моделі.

2.4 Пошук та підготовка набору даних

Для побудови моделі був обраний набір даних Optical Interconnection Network Data Set [24]. Усі симуляції виконані за допомогою програмного забезпечення під назвою OPNET Modeler. Передача повідомлень

використовується як механізм зв'язку, у якому будь-який процесор може надіслати в мережу повідомлення «точка-точка», призначене будь-якому іншому процесору. Черга M/M/1 враховується в обчисленнях, які складаються з буфера "першим прийшов – першим вийшов" із випадковим надходженням пакетів згідно з процесом надходження Пуассона та процесором, який отримує пакети з буфера із заданою швидкістю обслуговування. У всіх симуляціях передбачається, що процесор на кожному вузлі витягує пакет із вхідної черги, обробляє його протягом певного періоду часу, а коли цей період закінчується, він генерує повідомлення вихідних даних. Розмір кожної вхідної черги вважається нескінченним. Процесор перестає працювати тільки тоді, коли всі його черги введення порожні.

Ця база даних містить 11 атрибутів:

1. Номер вузла: кількість вузлів у мережі. (8x8 або 4x4).

2. Номер потоку: кількість потоків у кожному вузлі на початку симуляції.

3. Просторовий розподіл: продуктивність мережі оцінюється за допомогою синтетичного навантаження трафіку. Включено моделі трафіку Uniform (UN), Hot Region (HR), Bit Reverse (BR) і Perfect Shuffle (PS).

4. Тимчасовий розподіл: Тимчасовий розподіл генерації пакетів здійснюється незалежними джерелами трафіку. В наших симуляціях ми використовували трафік сервера клієнтів (тобто вузол сервера посилає пакети для відповіді на прийом пакетів від клієнтів) і асинхронний трафік (тобто спочатку всі вузли генерують трафік незалежно від інших; з плином часу, генерація трафіку в вузлах джерела/призначення залежний.

5. Про надходження повідомлень від вузлів призначення/джерела.

6. T/R: Час передачі повідомлень (T) рівномірно розподілений із середнім значенням в діапазоні від 20 до 100 тактових циклів. Час виконання потоків (R) експоненціально розподілений із середнім значенням 100 тактових циклів.

7. Утилізація процесора: Середня утилізація процесора вимірює відсоток часу, який працюють потоки в процесорі.

8. Час очікування каналу: Середній час очікування пакета в черзі вихідного каналу, поки він не обслуговується каналом.

9. Час очікування вводу: Середній час очікування пакета, поки він не буде обслуговуватися процесором.

10. Час відповіді мережі: Час між запитом повідомлення відображається на вихідному каналі, а відповідне повідомлення про дані надходить у чергу введення

11. Завантаження каналу: Відсоток часу, який канал зайнятий передачею пакетів в мережу.

Node_Number	Thread_Number	T_R	Processor_Utilization	Channel_Waiting_Time	Input_Waiting_Time	Network_Response_Time	Channel_Utilization
640.000000	640.000000	640.000000	640.000000	640.000000	640.000000	640.000000	640.000000
40.000000	7.000000	0.550000	0.649013	377.459157	333.247102	1504.247529	26.347886
24.018772	2.237817	0.287453	0.194737	381.974899	233.721860	1202.606968	223.782214
16.000000	4.000000	0.100000	0.202377	0.950721	33.036130	0.529210	0.138979
16.000000	5.500000	0.300000	0.492530	29.247560	137.730986	580.676198	0.587539
40.000000	7.000000	0.550000	0.624787	265.614624	261.855556	1232.150369	0.773611
64.000000	8.500000	0.800000	0.833106	664.965408	485.943680	2115.326618	0.905573
64.000000	10.000000	1.000000	0.986516	1627.330246	892.852416	6065.736672	2895.323131

Рис. 2.4. Зовнішній вигляд набору даних.

Набір даних Optical Interconnection Network Data Set містить 640 записів про використання мережі. Він був розбитий на дві частини: тренувальний (512) та тестовий (128). Тренувальний набір використовувався для знаходження оптимальних параметрів моделі, а тестовий набір даних був створений для оцінки кінцевої моделі.

Висновки до розділу 2

У підрозділі 2.1 було розглянуто прикладні алгоритми методів машинного навчання, які будуть використовуватись для побудови моделей.

Було прийнято рішення будувати моделі на основі методів: випадкогово лісу, градієнтного бустингу дерев рішень, методів лінійної регресії.

У підрозділі 2.2 було розглянуто існуючи метрики оцінювання ефективності моделей і методів прогнозування. Окремо розглянуті основні метрики для оцінювання роботи моделі для задач регресії.

В останньому розділі 2.3 було розглянуто методику оцінювання алгоритмів ковзний контроль(крос-валідація). Була розглянута задача навчання з учителем, а також була визначена процедура оцінювання завдяки ковзному контролю. Після цього були визначено основні типи даної методики, а також наведені недоліки та приклади використання.

В останньому розділі 2.4 було знайдено та проаналізовано набір даних, необхідний для тренування моделі випадкових лісів. Даним набором став Optical Interconnection Network Data Set [24]. Ця база даних містить 11 атрибутів, та містить інформацію про симуляцію комп'ютерного трафіку. На основі даного набору було сформовано відповідні тренувальні та тестові набори для підвищення ефективності тренування моделі.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ПРОГНОЗУВАННЯ ТРАФІКУ КОМП'ЮТЕРНИХ МЕРЕЖ НА ОСНОВІ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

3.1 Вибір програмного забезпечення

Вибір якісного ПО для розробки програмної реалізації методу прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання є досить важливим етапом. Були розглянуті наступні мови програмування для розробки моделі:

1) C++;
2) Matlab;
3) Python.

C++ — мова програмування високого рівня з хорошою продуктивністю та досить багатим синтаксисом. Недоліком цієї мови є те, що її загалом легше використовувати, ніж Python.

Matlab — дослідницька мова програмування, яка активно використовується в університетській сфері. До переваг Matlab можна віднести простоту навчання та простоту використання. Однак Matlab має високу ціну, низьку продуктивність і багато функцій.

Python став найпопулярнішою мовою машинного навчання нашого часу не лише для дослідницьких цілей, але й для створення комерційних систем. Велика кількість фреймворків використовує Python як мову інтерфейсу, що полегшує їх використання. Однак його недоліки включають низьку продуктивність і складність перенесення моделей, створених на Python, в інше середовище програмування. Серед фреймворків, які є визнаними та активно розвиваються, слід виділити наступні:

1) Tensorflow разом з Keras.
2) Google Colab

Tensorflow — це платформа машинного навчання, розроблена Google, яка також включає інтерфейс на основі Python. Перевагою цього фреймворку є велика кількість функцій і операторів, але, з іншого боку, це може призвести до надскладного його використання. Щоб вирішити цю проблему, був створений плагін Keras, який зосереджений лише на глибокому навчанні.

Google Colab — це безкоштовний хмарний сервіс на базі Jupyter Notebook. Google Colab надає все, що вам потрібно для машинного навчання, безпосередньо у вашому веб-переглядачі, надаючи вам безкоштовний доступ до блискавично швидких GPU та TPU. Colab дозволяє використовувати всі можливості популярних бібліотек Python для аналізу та візуалізації даних. У Colab ви можете імпортувати набір даних зображення, націлити класифікатор зображень і оцінити модель за допомогою лише кількох рядків коду. Colab активно використовується в області машинного навчання, зокрема:

- знайомства з TensorFlow;
- розробки та навчання нейронних мереж;
- експериментів з TPU;
- поширення досліджень в галузі ШІ.

Для розробки моделей прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання було обрано мову програмування Python та фреймворк Google Colab, через їхню простоту використання, малу затрату ресурсів та гарну ефективність, яка не поступається усім іншим засобам.

3.2 Результати імплементації програмного продукту

В результаті виконання кваліфікаційної роботи було створено програмний продукт для прогнозування трафіку комп'ютерних мереж на

основі алгоритмів машинного навчання. Даний продукт був розроблений за допомогою мови програмування Python, фреймворка Google Colab і відкритих бібліотек sklearn та Pandas. Програмний код був розбитий на частини. Це було зроблено для можливого підвищення легкості використання та внесення змін в майбутньому.

Таким чином, метод прогнозування складається з таких фрагментів, лістинг яких наведений у Додатку Г:

– у файлі «load_data_rf.ry» наведено код, який використовується для завантаження набору даних з бібліотеки openml, а також його валідації.

– у файлі «training_rf.ry» наведено код, який використовується для тренування моделей та оцінок результатів навчання

– у файлі «cross_rf.ry» наведено код, який використовується проведення процедури крос-валідації даних та її оцінки.

3.3 Аналіз результатів

Спочатку дані в тестовій виборці були нормалізовані завдяки методам із бібліотеки Sklearn. Після цього була створена матриця кореляції для наглядного перегляду залежності цільової змінної від атрибутів.

Після тренування моделей на тестовому наборі даних були отримані результати, які наведені у табл. 3.1.

Згідно з таблицями результатів, найкращі результати показали моделі, побудована на основі методів випадкових лісів та градієнтного бустингу. Після проведення процедури крос-валідації, результати покращились, що ще раз виправдовує її використання.

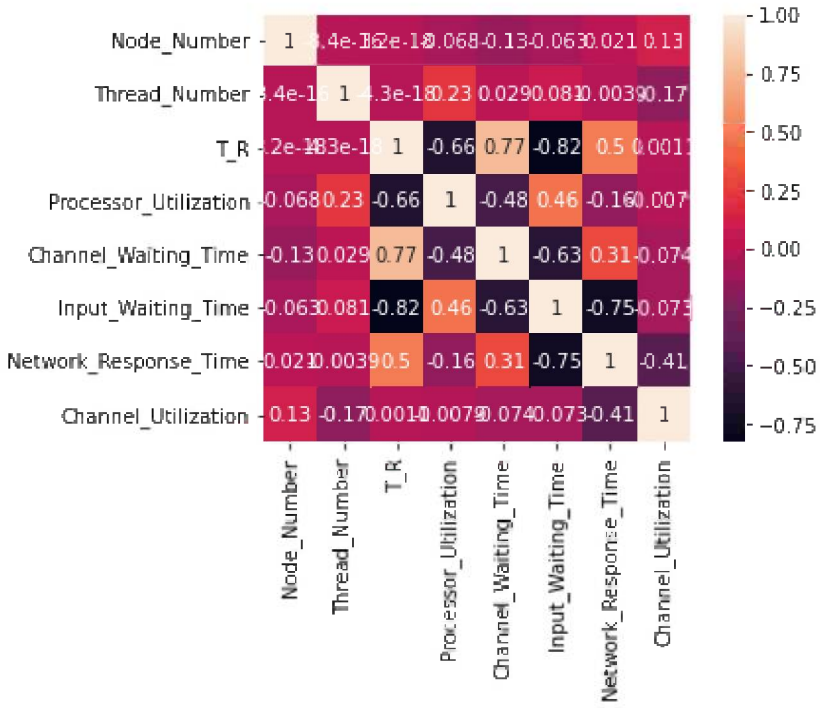

Рис. 3.1 – Матриця кореляції.

Таблиця 3.1

Результати на тренувальному наборі даних

Метод	Score для навч.	Score для тест.
Випадковий ліс	1.000	0.645
Градієнтний бустинг	1.000	0.619
Лінійна Регресія	0.764	0.012
Ridge	0.735	0.011
Lasso	0.000	0.000
Ridge CV	0.794	0.015
Випадковий ліс CV	1.000	0.859
Градієнтний бустинг CV	1.000	0.895

Значення метрики Score для моделей на основі випадкових лісів та градієнтного бустингу знаходиться на рівні 0.859 та 0.895 відповідно, що

позитивно оцінює роботу моделей. Висока оцінка крос-валідації виправдовує використання цієї процедури. Таким чином, модель повністю задовольнила поставлені вимоги.

Висновки до розділу 3

У підрозділі 3.1 були розглянуто можливі програмні забезпечення для створення методу та моделі прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання. Мовою програмування було обрано Python, а фреймворком для розробки моделі став хмарний сервіс Google Colab.

У наступному підрозділі 3.2 було описано програмний код методу та моделі прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання. Було зазначено, що програма складається з набору Python модулів, що полегшує легкість внесення потрібних модифікацій в майбутньому. У кінці підрозділу приведено опис усіх створених модулів.

Результати роботи моделі було приведено у підрозділі 3.3. Було проведено аналіз отриманих результатів та приведені показники метрик оцінювання ефективності. Метрика Score показала високий показник точності моделі, оцінка крос-валідації виправдала проведення даної процедури, тому був зроблений висновок, що технічні вимоги були задовільнені.

ВИСНОВКИ

Результатом даної кваліфікаційної роботи є модель, яка здатна робити прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання. Під час процесу розробки було розглянуто різноманітні методи машинного навчання для розв'язання задачі прогнозування стану систем, і у результаті аналізу було прийнято рішення використовувати саме регресійні методи, а саме лінійну регресію, випадковий ліс, градієнтний бустинг дерев рішень. Крім того, результати досліджень були представлені на Міжнародній науково-технічній конференції «Комп'ютерне моделювання у наукоємних технологіях (КМНТ – 2022)» [25].

Отримана модель була навчена, а потім практично використана на тестовому наборі даних. Результати роботи були задовільні, значення метрики Score а також оцінка крос-валідації показали високі показники ефективності роботи моделі.

У вступі до кваліфікаційної роботи було розглянуто актуальність даної проблеми, а також сформовано предмет, об'єкт та мету дослідження.

Перша частина містить сформовані задачі регресії та прогнозування, аналіз методів машинного навчання для розв'язання задачі прогнозування стану систем, що існують і окремо методів випадкових лісів, лінійної регресії та градієнтного бустингу. Були приведені інформативні ознаки та декілька прикладів випадкових лісів.

Друга частина містить алгоритми побудови моделей випадкового лісу та градієнтного бустингу дерев рішень, пояснення до імплементації даних моделей, а саме: опис метрик, за якими відбувалась оцінка ефективності моделей, обгрунтування вибору набору даних та застосування процедури крос-валідації.

В третьому розділі було обгрунтовано вибір програмного забезпечення, що використовувалось у ході розробки програмного застосунку, описано модулі, з яких складається програма. Також були розглянуті результати роботи моделі та показники метрик оцінювання ефективності.

Отже, на основі даної кваліфікаційної роботи та результатів, які були описані вище, був зроблений висновок, що використання регресійних моделей для прогнозування трафіку комп'ютерних мереж є виправданим, а також перспективним, тому може використовуватись для подальших досліджень. Модель є достатньо оптимізованою та придатною до модифікацій.

Таким чином, мета роботи та необхідні цілі були досягнуті, а поставлені завдання були вирішені.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Eremenko K. Data Science A-Z: Real-Life Data Science ExercisesIncluded. URL: https://www.udemy.com/course/datascience. (Last accessed: 05.12.2021).

2. Evgeniou T., Pontil M. Support Vector Machines: Theory and Applications. 2049. 249-257, 2001, DOI: 10.1007/3-540-44673-7_12 (Last accessed: 12.12.2021).

3. Приклад методу опорних векторів. URL: https:// upload.wikimedia.org/wikipedia/commons/thumb/7/72/SVM_margin.png/ 300px-SVM_margin.png (Дата звернення: 19.12.2021).

4. Ho, Tin Kam. Random decision forests // Proceedings of 3rd International Conference on Document Analysis and Recognition. – 1995. – vol.1. – P. 278–282. (Last accessed: 05.12.2021).

5. Breiman L., Friedman J. H., Olshen R. A, Stone C. J. Classification and Regression Trees. – 2001. – Vol. 45, no. 1. – P. 40 – 82. (Last accessed: 06.12.2021).

6. Sutton, C. D. Classification and regression trees, bagging, and boosting. – 2005. – P. 12 – 32. (Last accessed: 06.12.2021).

7. Breiman, Leo. Random Forests // Machine Learning: journal. – 2001. – Vol. 45, no. 1. – P. 5 – 32. (Last accessed: 06.12.2021).

8. Breiman L. Out-of-bag estimation // Technical report, Statistics Department University of California, Berkeley. 1996. P. 1– 13. (Last accessed: 15.12.2021).

9. Ruczinski I., Kooperberg C., LeBlanc M. Logic Regression methods and software / Eds: D. Denison et al. // Proceedings of the MSRI workshop on Nonlinear Estimation and Classification. New York: Springer, 2002. P. 333– 344. (Last accessed: 20.12.2021).

10. Prinzie A., Poel D. Random Multiclass Classification: Generalizing Random Forests to Random MNL and Random NB // Working paper, Department of Marketing, Ghent University, 2007 (Last accessed: 23.12.2021).

11. Genuer R., Poggi J.-M., Tuleau C. Random Forests: Some methodological insights // Research Report 6729, INRIA Saclay-Ile-de-France. 2008. P. 1–35. URL: http://hal.inria.fr/inria-00340725/fr (Last accessed: 19.01.2022).

12. Breiman L. Manual on setting up, using, and understanding random forests v3.1. 2002. URL: http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf. (Last accessed: 25.01.2022).

13. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Springer, 2001. 533 p. (Last accessed: 01.02.2022).

14. Shi T., Horvath S. Unsupervised Learning with Random Forest Predictors // Journal of Computational and Graphical Statistics. 2006. Vol. 15, N 1. P. 118–138. (Last accessed: 04.02.2022).

15. Buhlmann, P. Boosting for high-dimensional linear models. Ann. Stat. 34, 559–583. doi: 10.1214/009053606000000092. 2006. Vol. 15, N 1. P. 118–138. (Last accessed: 04.02.2022).

16. Hastie, T. Comment: boosting algorithms: regularization, prediction and model fitting. Stat. Sci. 22, 513–515. 2006. Vol. 17, N 1. P. 118–138. (Last accessed: 04.02.2022).

17. Michael Panik. Endocrine Manifestations of Systemic Autoimmune Diseases. 22, 513–515. 2006. Vol. 17, N 1. P. 118–138. (Last accessed: 04.02.2022).

18. Everitt, B. S.; Skrondal, A. (2010), The Cambridge Dictionary of Statistics, Cambridge University Press. Vol. 17, N 1. P. 118–138. (Last accessed: 04.02.2022).

19. R.J. Hyndman, A.B. Koehler, "Another look at measures of forecast accuracy", International Journal of Forecasting, 22 (2006). 118–138. (Last accessed: 04.02.2022).

20. Meinshausen N. Quantile Regression Forests // Journal of Machine Learning Research. 2006. Vol. 7. P. 983–999. (Last accessed: 16.02.2022).

21. Wolf B. J., Slate E. H., Hill E. G. Logic Forest: An ensemble classifier for discovering logical combinations of binary markers // Bioinformatics. 2010. Vol. 26, N 17. P. 2183–2189. (Last accessed: 17.02.2022).

22. Приклад ROC кривої. URL: https://habrastorage.org/web/299/157/fad/299157fad56a4ecca8f6b96b425bd38c.png (Дата звернення: 19.02.2022).

23. Воронцов К. В. Комбинаторный подход к оценке качества обучаемых алгоритмов. — Математические вопросы кибернетики / Под ред. О.Б. Лупанов. — М.: Физматлит, 2004. — Т. 13. — С. 5–36. (Дата звернення: 24.02.2022).

24. Optical Interconnection Network Data Set. (Last accessed: 27.02.2022).

25. Регресійні моделі прогнозування трафіку комп'ютерних мереж / В.Б. Головченко, В.Є. Стрілець // Збірник наукових праць Міжнародної науково-технічної конференції «Комп'ютерне моделювання у наукоємних технологіях». – Х. : ХНУ імені В.Н. Каразіна, 2022. http://www-csd.univer.kharkov.ua/science/konferentsiyi

ДОДАТКИ

Додаток А

Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Освітньо-кваліфікаційний рівень Магістр
Галузь знань 12 – Інформаційні технології
Спеціальність 123 – Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та
прикладної системотехніки
_____д.т.н., проф. Шматков С. І.

«_10_» __12__2021року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Головченка Віталія Борисовича
(прізвище, ім'я, по батькові)

1. Тема роботи: Регресійні моделі прогнозування трафіку комп'ютерних мереж
_____

керівник роботи: Стрілець Вікторія Євгенівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "___"_____20__року №___

2. Строк подання студентом роботи _____30 листопада 2022_____

3. Перелік питань, які потрібно розробити:
3.1. Порівняльний аналіз методів прогнозування стану систем.
3.2. Розробка моделей прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання.
3.3. Розробка та тестування програмного додатку, який реалізує регресійні моделі прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання.

4. План роботи

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи
4.1	Формулювання цілей роботи та постановка задачі.	31.10.2021 –01.12.2021
4.2	Аналіз існуючих методів класифікації та прогнозування трафіку комп'ютерних мереж.	02.12.2021 – 05.01.2022
4.3	Збір вибірок даних за трафіком комп'ютерних мереж та їх інформаційний аналіз.	06.01.2022 – 31.03.2022
4.4	Розробка моделі прогнозування трафіку комп'ютерних мереж на основі алгоритму випадкових лісів.	01.04.2022 – 01.05.2022
4.5	Розробка моделі прогнозування трафіку комп'ютерних мереж на основі лінійних алгоритмів регресії.	02.05.2022 – 02.06.2022
4.6	Розробка програмного додатку для прогнозування трафіку комп'ютерних мереж на основі алгоритму випадкових лісів	03.06.2022 – 15.07.2022
4.7	Розробка програмного додатку для прогнозування трафіку комп'ютерних мереж на основі лінійних алгоритмів регресії.	16.07.2022 – 31.08.2022
4.8	Тестування розроблених програмних додатків та порівняльний аналіз результатів.	01.09.2022 – 04.11.2022
4.9	Оформлення пояснювальної записки.	05.11.2022 – 01.12.2022

5. Дата видачі завдання_____10.12.21_____

Студент _____ Головченко В.Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стрілець В.Є.
(підпис) (прізвище та ініціали)

Додаток Б

Затверджую

_____

«_____»_____2022 р.

Технічне завдання
на розробку програмного виробу
«Регресійні моделі
прогнозування трафіку комп'ютерних мереж»

Назва розділу	Назва і зміст підрозділу
1. Введення	1) Назва: Програмний застосунок для регресійних моделей прогнозування трафіку комп'ютерних мереж 2) Область застосування: комп'ютерні мережі
2. Підстава для розробки	1) Навчальний план ФКН за спеціальністю 123 - Комп'ютерна інженерія. 2) Завдання на кваліфікаційну роботу, затверджене наказом № _ від _____ привести в Додатку А.
3. Призначення розробки	Програмний застосунок для методу прогнозування трафіку комп'ютерних мереж на основі алгоритмів машинного навчання призначений для збільшення ефективності та точності прогнозу стану мережевої системи.
4. Вхідні дані	В якості основи взяти набір даних Optical Interconnection Network Data Set.
4. Технічні вимоги до програмного виробу	4.1.Вимоги до функціональних характеристик: програмний виріб повинен надавати можливість натренувати нову модель, оцінити ефективніть її роботи

	за допомогою метрик score, провести прогнозування значень цільових змінних. 4.2. Вимоги до надійності: метрика точності score на тестовому наборі даних повинна мати значення не менше ніж 0.65. 4.3. Вимоги до умов експлуатації: для виконання програми потрібно, щоб на ПК було встановлено стабільне інтернет-з'єднання, остання версія Python та бібліотеки sklearn. 4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми потрібен комп'ютер, оснащений стабільним інтернет- з'єднанням, встановленою останньою версією Python та бібліотеки sklearn. 4.5. Вимоги до інформаційної та програмної сумісності: підтримка наступних ОС: Windows, Linux. 4.6. Вимоги до маркування та упаковки: відсутні. 4.7. Вимоги до транспортування і зберігання: відсутні. 4.8. Спеціальні вимоги: відсутні.
5. Вимоги до програмної документації.	Програмною документацією до виробу «Програмний застосунок для регресійних моделей прогнозування трафіку комп'ютерних мереж» вважати: 1) Справжнє Технічне завдання на розробку програмного виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Програму і методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи). 3) Опис програмного виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи).

<table>
<tr><td></td><td colspan="2">4) Лістинг програмного коду (представити у Додатку Г).</td></tr>
<tr><td>6. Техніко-економічні показники</td><td colspan="2">1) Визначення економічних переваг методу у порівнянні з вітчизняними та зарубіжними аналогами: виконати в розділі 1 пояснювальної записки до кваліфікаційної роботи
2) Оцінка економічної ефективності – непотрібна.</td></tr>
<tr><td rowspan="8">7. Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>31.10.2021
01.12.2021</td><td>Формулювання цілей роботи та постановка задачі.</td></tr>
<tr><td>02.12.2021
05.01.2022</td><td>Аналіз існуючих методів класифікації та прогнозування трафіку комп’ютерних мереж.</td></tr>
<tr><td>06.01.2022
31.03.2022</td><td>Збір вибірок даних за трафіком комп’ютерних мереж та їх інформаційний аналіз.</td></tr>
<tr><td>01.04.2022
01.05.2022</td><td>Розробка моделі прогнозування трафіку комп’ютерних мереж на основі алгоритму випадкових лісів.</td></tr>
<tr><td>02.05.2022
02.06.2022</td><td>Розробка моделі прогнозування трафіку комп’ютерних мереж на основі лінійних алгоритмів регресії.</td></tr>
<tr><td>03.06.2022
15.07.2022</td><td>Розробка програмного додатку для прогнозування трафіку комп’ютерних мереж на основі алгоритму випадкових лісів</td></tr>
</table>

	16.07.2022 31.08.2022	Розробка програмного додатку для прогнозування трафіку комп'ютерних мереж на основі лінійних алгоритмів регресії.
	01.09.2022 04.11.2022	Тестування розроблених програмних додатків та порівняльний аналіз результатів.
	05.11.2022 01.12.2022	Оформлення пояснювальної записки.
8. Порядок контролю і приймання	1) Перевірку ходу розробки програмного виробу Керівнику робіт виконувати 1 раз на тиждень. 2) Випробування програмного виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу. 3) Захист розробленого програмного виробу провести на засіданні атестаційної комісії. 4) Пояснювальну записку уявити на паперових носіях в одному примірнику, в електронному вигляді - на CD-диску в одному екземплярі.	

Виконавець
студент групи КІ-61
Головченко В. Б.

Замовник
к. т. н., доцент кафедри ТПС
Стрілець В. Є.

Додаток В

Програма і методика випробувань програмного виробу «Програмний застосунок для регресійних моделей прогнозування трафіку комп'ютерних мереж»

1. Об'єкт випробувань

Об'єктом випробовування є програмний застосунок для методу прогнозування стану систем на основі машинного навчання. (розділ 1 ТЗ).

2. Мета випробувань

Перевірка роботоспособності програмного застосунок для методу програмний застосунок для методу прогнозування стану систем на основі машинного навчання, оцінка ефективності роботи моделі за допомогою метрики score, перевірка необхідності крос-валідації даних (Додаток Б).

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ № _ від _____ про призначення атестаційної комісії та перевірку даного виробу.

3.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу або будь-якого приміщення в період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Програмний виріб повинен надавати такі можливості:

1) Натренувати нову модель.

2) Оцінити ефективніть роботи моделі за допомогою метрики score.

3) Провести процедуру крос-валідація для покращення роботи моделі та оцінити її точність.

Метрика точності score на тестовому наборі даних повинна мати значення не менше ніж 0.65.

Для експлуатації програми потрібно, щоб на ПК було встановлено стабільне інтернет-з'єднання, остання версія Python та бібліотеки sklearn

Програмний застосунок повинен підтримувати роботу на наступних операційних системах: Windows, Linux

5. Вимоги до програмної документації

Склад програмної документації, що подається на випробування, включає:

1) Технічне завдання на розробку програмного виробу (представлено в Додатку Б до пояснювальної записки до кваліфікаційної роботи).

2) Ця програма і методика випробувань розробленого програмного виробу (представлена в Додатку В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного виробу (представлено в розділі 3 пояснювальної записки до кваліфікаційної роботи).

4) Фрагмент програми (представлений в Додатку Г до пояснювальної записки до кваліфікаційної роботи)

6. Засоби і порядок випробувань

6.1. Засоби випробувань

Для виконання програми потрібно, щоб на ПК було стабільне інтернет-з'єднання, а також встановлено останню версію Python та бібліотека sklearn.

6.2. Порядок проведення випробувань

6.2.1. Перевірка програмної документації

1) Перевірка складу програмної документації. Перевірку здійснювати за критерієм наявності, представленої в ТЗ документації.

2) Перевірка якості програмної документації. Перевірку здійснювати за критерієм відповідності вимогам ЕСПД. «Програма і методика випробувань».

6.2.2. Перевірка працездатності методу

Тест 1

1. Перевірку працездатності здійснюємо шляхом запуску програми розміщеної у «load_data_rf.py». Отримуємо результат ініціалізації набору даних.

	Node_Number	Thread_Number	T_R	Processor_Utilization	Channel_Waiting_Time	Input_Waiting_Time	Network_Response_Time	Channel_Utilization
count	640.000000	640.000000	640.000000	640.000000	640.000000	640.000000	640.000000	640.000000
mean	40.000000	7.000000	0.550000	0.649013	377.459157	333.247102	1504.247529	26.347886
std	24.018772	2.237817	0.287453	0.194737	381.974899	233.721860	1202.606968	223.782214
min	16.000000	4.000000	0.100000	0.202377	0.950721	33.036130	0.529210	0.136979
25%	16.000000	5.500000	0.300000	0.492530	29.247560	137.730985	580.676198	0.587539
50%	40.000000	7.000000	0.550000	0.624787	265.614624	261.855556	1232.150369	0.773611
75%	64.000000	8.500000	0.800000	0.833106	664.965408	485.943680	2115.326618	0.905573
max	64.000000	10.000000	1.000000	0.986516	1627.330246	892.852416	6065.736672	2895.323131

Рис. В.1 – Перегляд деталей завантаженого набору даних.

Тест 2

2. Перевірку здійснюємо шляхом запуску програми розміщеної у «models_rf.py» із вказуванням всіх необхідних параметрів. Отримуємо показник метрики score - точність роботи моделі.

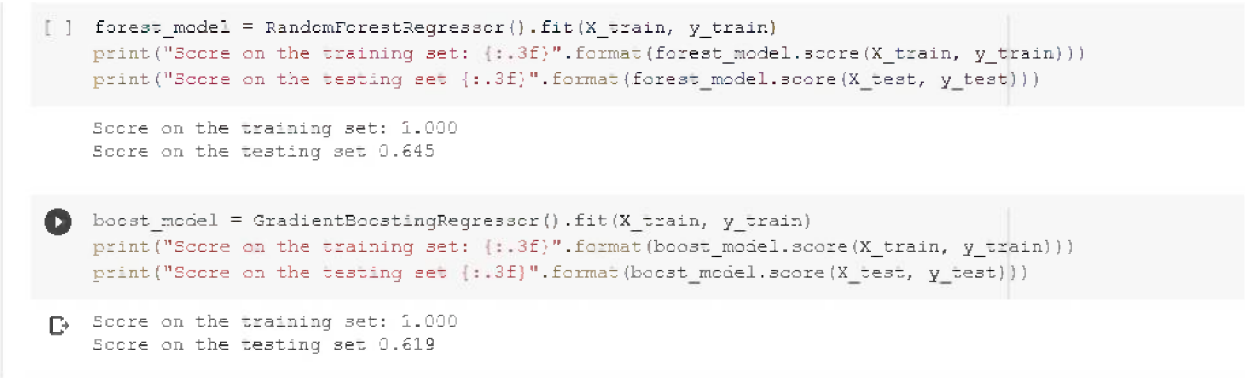

Рис. В.2 – Результати роботи моделі.

Тест 3

3. Перевірку здійснюємо шляхом запуску програми розміщеної у «cross_rf.py» із вказуванням всіх необхідних параметрів. Отримуємо оцінку крос-валідації.

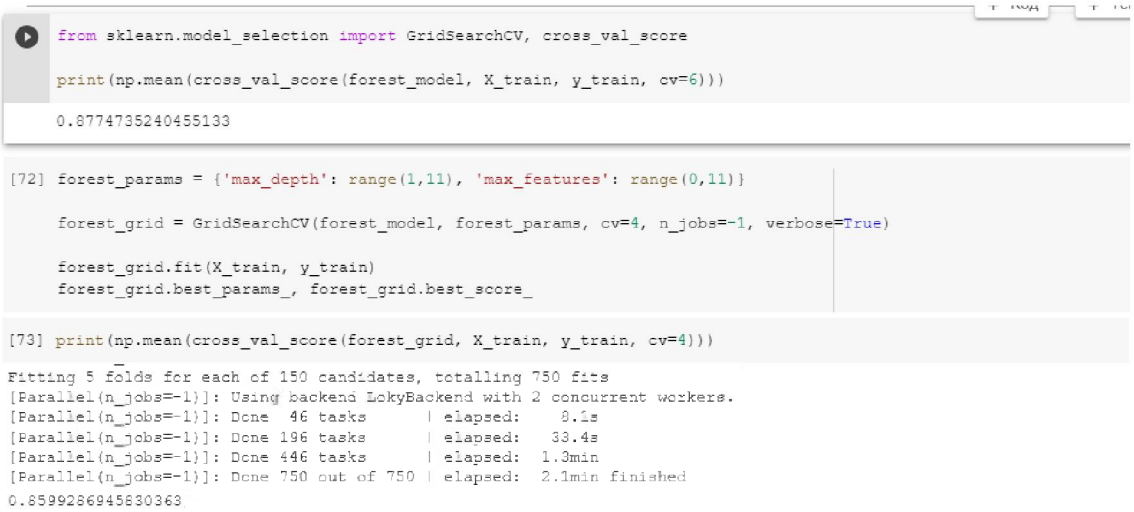

Рис. В.3 – Оцінка крос-валідації.

Висновок: якщо були пройдені всі тестування у пунктах 1-3, результати випробування вважати успішними.

Виконав
студент групи КІ-61 Головченко В.Б. 

Додаток Г

Фрагменти програмного коду

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

```
# dataset optical_interconnection_network from OpenML
# Description on https://www.openml.org/search?type=data&status=active&id=42365
from sklearn.datasets import fetch_openml

df = fetch_openml(data_id=42365, as_frame=True).frame
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 640 entries, 0 to 639
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Node_Number            640 non-null    float64
 1   Thread_Number          640 non-null    float64
 2   Spatial_Distribution   640 non-null    category
 3   Temporal_Distribution  640 non-null    category
 4   T_R                    640 non-null    float64
 5   Processor_Utilization  640 non-null    float64
 6   Channel_Waiting_Time   640 non-null    float64
 7   Input_Waiting_Time     640 non-null    float64
 8   Network_Response_Time  640 non-null    float64
 9   Channel_Utilization    640 non-null    float64
dtypes: category(2), float64(8)
memory usage: 41.7 KB
```

Рис. Г.1 – Фрагмент програмного коду «load_data_rf.py».

```
X_train, X_test, y_train, y_test = train_test_split(df_data, df_target, test_size=0.2, random_state=42)
X_train
```

```
forest_model = RandomForestRegressor().fit(X_train, y_train)
print("Score on the training set: {:.3f}".format(forest_model.score(X_train, y_train)))
print("Score on the testing set {:.3f}".format(forest_model.score(X_test, y_test)))
```

```
Score on the training set: 1.000
Score on the testing set 0.641
```

```
from sklearn.model_selection import GridSearchCV, cross_val_score

print(np.mean(cross_val_score(forest_model, X_train, y_train, cv=6)))
```

```
-0.2597710545344943
```

```
forest_params = {'max_depth': range(1,11), 'max_features': range(0,11)}

forest_grid = GridSearchCV(forest_model, forest_params, cv=4, n_jobs=-1, verbose=True)
```

Рис. Г.2 – Фрагмент програмного коду «models_rf.py».

```
print(np.mean(cross_val_score(forest_grid, X_train, y_train, cv=4)))
```

```
boost_model = GradientBoostingRegressor().fit(X_train, y_train)
print("Score on the training set: {:.3f}".format(boost_model.score(X_train, y_train)))
print("Score on the testing set {:.3f}".format(boost_model.score(X_test, y_test)))
```

```
Score on the training set: 1.000
Score on the testing set 0.619
```

```
print(np.mean(cross_val_score(boost_model, X_train, y_train, cv=6)))
```

```
0.8528801539201211
```

```
boost_params = {'max_depth': range(1,11), 'max_features': range(0,11)}

boost_grid = GridSearchCV(boost_model, boost_params, cv=4, n_jobs=-1, verbose=True)

boost_grid.fit(X_train, y_train)
boost_grid.best_params_, boost_grid.best_score_
```

Рис. Г.3 – Фрагмент програмного коду «cross_rf.py».