Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Факультет інформаційних технологій
(факультет)

Кафедра системного аналізу і управління
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню магістра
(назва освітньо-кваліфікаційного рівня)

студента Подвисоцького Артема Сергійовича
(ПІБ)

академічної групи 124м-22-1
(шифр)

спеціальності 124 - Системний аналіз
(код і назва спеціальності)

на тему «Застосування інтелектуального аналізу даних для підвищення точності класифікації споживачів кредитних запозичень»

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	к.т.н., доцент Алексєєв О.М.			
розділів:				
Інформаційно-аналітичний	к.т.н., доцент Алексєєв О.М.			
Спеціальний	к.т.н., доцент Алексєєв О.М.			

Рецензент				

Нормоконтролер	к.ф.-м.н., доц Хом'як Т.В.			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри
Системного аналізу і управління
(повна назва)

_____ к.т.н., доцент Т.А. Желдак
(підпис) (прізвище, ініціали)

«_____»_____2020 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра

студенту Подвисоцькому Артему Сергійовичу **академічної групи** 124м-19-1
(прізвище та ініціали) (шифр)

напряму підготовки 124 Системний аналіз
(код і назва спеціальності)

спеціалізації_____

на тему Застосування інтелектуального аналізу даних для підвищення точності

класифікації споживачів кредитних запозичень ,
(назва за наказом ректора)

затверджену наказом ректора НТУ «Дніпровська політехніка» від 09 жовтня 2023 р. №1227-с

Розділ	Зміст	Термін виконання
1. Інформаційно-аналітичний	Розгляд поняття кредитного скорингу, методів інтелектуального аналізу та класифікації даних, дослідження переваг алгоритму машинного навчання XGBoost.	
2. Спеціальний	Препроцесінг вихідних даних, побудова моделі алгоритму XGBoost та покращення розв'язку за рахунок оптимізації параметрів моделі.	

Завдання видано _____ к.т.н., доц. О.М. Алексєєв
(підпис керівника) (прізвище, ініціали)

Дата видачі _____

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Подвисоцький А.С.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 96 с., 40 рис., 1 табл., 3 додатки, 34 джерела.

Скорингові системи – популярний в банківській сфері інструмент оцінки клієнтів, в основі якого закладені статистичні методи. Такі системи дозволяють знизити витрати і мінімізувати операційний ризик за рахунок автоматизації прийняття рішення про надання кредиту.

Об'єкт дослідження: анкетні дані споживачів кредитів для задачі кредитного скорингу.

Предмет дослідження: алгоритм машинного навчання XGBoost, заснований на дереві пошуку рішень, який використовує фреймворк градієнтного бустинга.

Мета дослідження: дослідження якості класифікації позичальників в задачі кредитного скорингу за допомогою моделі алгоритму машинного навчання XGBoost.

Методи дослідження та апаратура: методи інтелектуального аналізу та класифікації даних, мова програмування Python.

Економічний ефект від реалізації результатів дослідження очікується позитивним за рахунок використання суб'єктами господарювання алгоритму препроцесінгу даних та побудови релевантної моделі алгоритму машинного навчання XGBoost для оптимізації їх діяльності.

В інформаційно-аналітичному розділі наведено визначення кредитного скорингу, описано основні алгоритми для вирішення задачі кредитного скорингу, проведено аналіз об'єкту дослідження, розглянуто послідовність препроцесінгу даних та методів вибору оптимального набору інформативних ознак. Описано можливості мови програмування Python для інтелектуального аналізу даних.

У спеціальному розділі виконано аналіз та підготовку вихідних даних, створено модель алгоритму XGBoost та відібрано найбільш інформативні ознаки, проведено аналіз параметрів моделі та відібрано ті, що максимально покращують

результати прогнозу; представлено результати розв’язку задачі кредитного скорингу.

Практична цінність роботи полягає у розробці детального алгоритму препроцесінгу даних, побудові релевантної моделі алгоритму машинного навчання XGBoost та дослідженні її на адекватність для ефективного прогнозу кредитоспроможності позичальників з метою зниження ризиків для банківської установи при видачі кредиту.

Ключові слова: КРЕДИТНИЙ СКОРИНГ, КРЕДИТОСПРОМОЖНІСТЬ, ДЕРЕВО КЛАСИФІКАЦІЇ, ПРЕПРОЦЕСІНГ, КАТЕГОРІАЛЬНІ ДАНІ, МАШИННЕ НАВЧАННЯ, PYTHON, XGBOOST.

THE ABSTRACT

Explanatory note: 96pages, 40 pic., 1 table, 3 applications, 39 sources.

Scoring systems are a popular customer assessment tool in the banking sector, based on statistical methods. Such systems can reduce costs and minimize operational risk by automating the decision to grant a loan.

Object of research: personal data of credit consumers for the task of credit scoring.

Subject of research: XGBoost machine learning algorithm, based on a solution search tree, that uses a gradient boosting framework.

Objective: researching of the quality of classification of borrowers in the problem credit scoring models using machine learning algorithm XGBoost.

Research methods: methods of data mining and classification, programming language – Python.

The *economic effect* of the implementation of the results of the study is expected to be positive due to the fact that businesses use the created algorithm for data preprocessing and build a relevant model of the machine learning algorithm XGBoost to optimize their activities.*The practical significance of the results:* is that they can be successfully used in the real management of complex modern systems.

The *information and analytical section* defines credit scoring, describes the main algorithms for solving the problem of credit scoring, analyzes the object of study, considers the sequence of data preprocessing and methods of selecting the optimal set of informative features. The capabilities of the Python programming language for data mining are described.

In a *special section* the analysis and preparation of initial data is performed, the model of the XGBoost algorithm is created and the most informative features are selected, the analysis of model parameters is carried out and the ones that improve the forecast results are selected; the results of solving the problem of credit scoring are presented.

The *practical value* of the work is to develop a detailed algorithm for data preprocessing, build a relevant model of machine learning algorithm XGBoost and study its adequacy for effective forecasting of solvency of borrowers to reduce risks for the bank when issuing a loan.

Keywords: CREDIT SCORING, SOLVENCY, CLASSIFICATION TREE, PREPROCESSING, CATEGORY DATA, MACHINE LEARNING, PYTHON, XGBOOST.

ЗМІСТ

ВСТУП

Сучасний світ неможливо уявити без банків. Вони є основним фінансовим посередником в економіці. Діяльність банків представляє собою той канал, за допомогою якого зміни на грошовому ринку трансформуються в зміни на товарному ринку.

Банки є фінансовими посередниками, оскільки, з одного боку, вони приймають вклади (депозити), залучаючи гроші вкладників, тобто акумулюють тимчасово вільні грошові кошти, а з іншого, надають їх під певний відсоток різним економічним агентам (фірмам, домогосподарствам та ін.), тобто видають кредити. Таким чином, банки – це посередники в кредиті, а банківська система є частиною кредитної системи. Кредитування, будучи найважливішою діяльністю банків, робить істотний вплив на розвиток національної економіки.

Величезні масштаби кредитування економіки вимагають і формування відповідних обсягів ресурсів для цього. Кредитна політика банків орієнтується на циклічні коливання в економіці, зайнятість населення, стійкість платіжного балансу, темп інфляції і економічне зростання країни. Спираючись на вартість фінансових ресурсів, банки встановлюють ставку відсотка по кредиту. Через нестабільну економіку сьогодення, ставка кредиту плаваюча, наслідком чого є зростання проблемної заборгованості за кредитами. Зрозуміло, що банки хочуть мати якомога точніший інструмент для оцінки кредитоспроможності позичальника.

Для оцінки кредитоспроможності позичальника банки використовують різні методики і схеми. Але в цілому аналіз кредитоспроможності клієнта часто здійснюється, виходячи з оцінки рівня доходів позичальника і їх зіставлення з майбутніми платіжками за кредитом, вивчення кредитної історії або ж скорингової оцінки.

Аналіз кредитної історії має на увазі під собою перевірку даних про якість платіжної дисципліни потенційного позичальника за вже раніше отриманими кредитами в цьому банку (у разі чинного клієнта) або ж в інших банках. Для

перевірки кредитної історії аналітики використовують внутрішні і зовнішні «чорні списки», «білі списки», існуючі бази кредитних історій. В Україні найбільш відома і велика база на даний час – Українське Бюро Кредитних Історій. Не існує однієї загальної бази з даними, що обумовлено в першу чергу небажанням банків розкривати конкурентам інформацію по позичальниках. Тому не завжди, навіть в разі наявності проблем з погашенням попередніх кредитів у клієнта або шахрайства, банк зможе про це дізнатися.

Скоринг – це евристичний спосіб побудови рейтингів і класифікації різних об'єктів на групи. Він ґрунтується на припущенні про те, що люди зі схожими соціальними показниками поводяться однаково. Він застосовується в банківській сфері, маркетингу, страховій справі. Основною метою традиційного скорингу є класифікація клієнтів банку на "хороших" і "поганих" позичальників, виходячи з якої кредитор може вибирати відповідні дії по відношенню до даного клієнта, тобто приймати рішення про те, видавати чи не видавати кредит. Для досягнення максимальної точності прогнозу алгоритми класифікації потребують якомога більшу кількість вихідних даних.

Потребу в обробці великих обсягів даних (Big Data) народжує інформація, що росте в колосальних обсягах. В цьому напрямку важливе місце відведено інтелектуальному аналізу даних (Data Mining). Цей напрямок включає в себе методи, відмінні від класичного аналізу, що вирішують завдання узагальнення, асоціювання і відшукання закономірностей. У великій мірі розвитку цієї дисципліни сприяло проникнення в сферу аналізу даних ідей, що виникли в теорії штучного інтелекту.

Найчастіше дані, з якими доводиться працювати алгоритмам, обумовлені такими особливостями як висока розмірність і надвеликий обсяг даних (мільйони записів в тисячах полів), а також включення у себе великої кількості числових (numerical), що здатні упорядковуватися в просторі, і категорійних (categorical), що не мають упорядкування, ознак. Підготовка таких даних повинна вестися окремо для коректної роботи алгоритму.

Найбільш популярні методи побудови скорингових алгоритмів на сьогоднішній будуються на основі логістичної регресії, дерева класифікації та нейронної мережі.

В даній роботі запропоновано розглянути алгоритм машинного навчання, заснований на дереві пошуку рішень, котрий використовує фреймворк градієнтного бустинга – XGBoost. Він є надійним, дуже потужним і складним алгоритмом, який дає сучасні результати для багатьох складних проблем, він також є дуже популярним і улюбленим алгоритмом у системі організації конкурсів з дослідження даних під назвою Kaggle, що також є соціальною мережею фахівців з обробки даних та машинного навчання.

1. ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

1.1 Роль банків у сучасній економіці

Банк – це основна ланка кредитної системи будь-якої держави. Банк – це автономна, незалежна кредитна установа, що має статус юридичної особи та здійснює діяльність в сфері надання фінансово-кредитних послуг. Банк є самостійним господарюючим суб'єктом, що діє на принципах господарського розрахунку. Специфіка полягає в тому, що банки працюють в сфері обміну, а не в сфері виробництва. Головним в їх діяльності є організація кредитно-грошового процесу і монетарна політика.

На сьогоднішній день банківський сектор виступає однією із найважливіших і невід'ємних структур ринкової економіки. Розвиток банків, товарного виробництва і обігу історично йшли паралельно і тісно переплітались. При цьому банки, виступаючи посередниками в перерозподілі капіталів, істотно підвищують загальну ефективність виробництва [1].

У сучасному світі значення банків вийшло за рамки власне грошових і кредитних відносин. Банки виступають в ролі інституту, що стоїть на рівні з державою і ринком. Без них немислима нормальна, раціональна організація господарської діяльності в суспільному масштабі.

Поняття «банк» найчастіше асоціюється зі сховищем грошей, однак більш широкий підхід передбачає вивчення різноманітних точок зору для цього поняття. Зокрема, банки розглядаються як посередницька організація, торгове підприємство, господарський суб'єкт, установа, організаційна структура, власність, володіння, суб'єкт фінансового ринку, об'єкт нагляду і регулювання. І нарешті, банки є носіями комплексу специфічних функцій, таких як перерозподільна, контрольна, економія витрат обігу, акумуляція коштів, посередницька функція регулювання грошових оборотів [2].

Залежно від виду і форми банківських об'єднань, характеру власності та інших показників банк може виконувати різні функції:

1) Збирання або акумулювання тимчасово вільних грошових коштів і перетворення їх в капітал. Виконуючи цю функцію, банки акумулюють грошові доходи і заощадження у формі вкладів. Вкладник отримує винагороду у вигляді відсотка або наданих банком послуг. Сконцентровані у внесках заощадження перетворюються в позиковий капітал, який використовується банками для надання кредитів підприємствам та підприємцям. Тільки за допомогою банків, заощадження перетворюються на капітал. Акумуляція коштів стає одним з основних видів діяльності банків. Для здійснення цієї функції потрібен спеціальний дозвіл – ліцензія.

2) Кредитування підприємств, держави і населення. У сучасних умовах розвитку підприємництва, малого і середнього бізнесу, це дуже важлива і актуальна функція. Так як для розвитку приватного бізнесу потрібні значні фінансові витрати, їх можна отримати або через банк, або шляхом отримання іноземних інвестицій. Банк виступає в якості фінансового посередника, отримуючи грошові кошти у кінцевих кредиторів і даючи їх кінцевим позичальникам. За рахунок банківських кредитів здійснюється фінансування промисловості, сільського господарства, торгівлі, а також забезпечується розширення виробництва. Банки надають позики споживачам на придбання товарів тривалого споживання, сприяючи, тим самим, зростання їх рівня життя. І, нарешті, так як державні витрати не завжди покриваються доходами, банки кредитують фінансову діяльність уряду.

3) Регулювання грошового обороту. Банки виступають центрами, через які проходить платіжний оборот різних господарюючих суб'єктів. Завдяки системі розрахунків, банки створюють для своїх клієнтів можливість здійснювати обмін, обіг грошових коштів і капіталу. Через банки проходить оборот як окремо взятої людини, так і економіки країни в цілому. Через них здійснюється перелив грошових коштів і капіталів від одного суб'єкта до іншого, від однієї галузі народного господарства до іншої.

4) Посередницька функція, відповідно до якої діяльність банків розуміється як посередник в платежах. Через банки проходять платежі

підприємств, організацій і населення. Здійснюючи за їх дорученням платежі, цим самим банк виконує посередницьку місію. В руках банків ця функція стає значно ширше, ніж елементарна посередницька діяльність. Банк може акумулювати невеликі розміри тимчасово вільних грошових коштів багатьох клієнтів і підсумувавши їх, направити величезні грошові ресурси тільки одному суб'єкту. Також банк може брати гроші у клієнтів на короткий термін, а видавати їх на тривалий час. Він може акумулювати ресурси в одному секторі економіки будь-якого регіону, або перерозподілити їх в інші галузі і зовсім інші регіони. Так як банки знаходяться в центрі економічного життя, вони отримують можливість змінювати розмір, терміни і напрямки капіталів відповідно до виникаючих потреб господарства.

5) Банки мають право здійснювати випуск, купівлю, продаж, облік, зберігання та інші операції з цінними паперами, що підтверджують залучення коштів у внески і на банківські рахунки. З іншими цінними паперами, операції з якими не вимагають спеціальної ліцензії, банки мають право також здійснювати довірче управління за договором з фізичними та юридичними особами.

6) Виконують інформаційно-консультативну функцію. Вони надають консультаційні послуги своїм клієнтам із приводу випуску і обігу цінних паперів, котирування валют і акцій. Банки інформують клієнтів і населення про зміни фінансового стану в економіці країни, зміни процентних ставок і проблем на валютному ринку. Найчастіше інформацією такого роду володіють тільки банки і дізнатися про це можливо тільки тому, що банки виконують інформаційну функцію.

Банк може розміщувати свої ресурси в цінні папери від свого імені, тоді всі ризики, пов'язані з таким розміщенням, усі доходи і збитки від зміни ринкової оцінки придбаних цінних паперів відносяться за рахунок акціонерів банку.

Окремими функціями наділений Центральний банк. Центробанк – це емісійний банк, тобто, він наділений правом емісії грошових знаків в обіг. Характерними для Центрального банку є наступні функції: емісія і контроль грошового обігу; функція резервного центру банків; управління державним

боргом; поповнення держбюджету; проведення наукових досліджень; контроль і вплив на комерційні банки.

Таким чином, банк – фінансова організація, установа, яка провадить різноманітні види операцій з грошима і цінними паперами і що надає фінансові послуги уряду, підприємствам, громадянам і іншим банкам. Банки випускають, зберігають, надають у кредит, купують і продають, обмінюють гроші і цінні папери, контролюють рух грошових коштів, обіг грошей і цінних паперів, надають послуги щодо платежів і розрахунків. Без банків немислимо сучасне грошове господарство. Їм немає альтернативи в майбутньому, оскільки вони є головною і сполучною ланкою всього економічного життя [3].

1.2 Роль кредитних відносин у суспільстві

Кредит (від лат. creditum – позика, борг) є однією з найскладніших економічних категорій. Передумовою його історичного генезису було майнове розшарування суспільства в період розкладу первіснообщинного ладу. Але характеру об'єктивної необхідності він набув лише за умов становлення і розвитку товарно-грошових відносин.

Ця необхідність була зумовлена особливими взаємовідносинами між товаровиробниками: коли продавцю потрібно було продавати товар, а в покупця не було грошей, щоб його купити (тому що він ще не виготовив або не продав свій), виникала потреба у передачі продавцем покупцеві товару з відстрочкою платежу, в кредит. Ще більшою мірою, чим функціонуючим виробникам, кредит був необхідним тим, хто тільки прагнув організувати виробництво, але не мав для цього власних коштів.

З часом розвиваються різні форми (товарна та грошова) і види (комерційний, банківський, споживчий, іпотечний та ін.) кредиту, виникли відповідні економічні інститути, які акумулюють і перерозподіляють кредитні ресурси-банки, постійно розширювалась сфера функціонування, яка відіграє важливу роль у забезпеченні неперервності процесу відтворення, а конкретніше-неперервності продукту в процесі його руху.

У молодих людей потреби часто перевищують їх прибутки. І навпаки, у людей похилого віку прибутки, як правило, перевищують видатки. Це також, з одного боку, викликає потребу в кредиті, а з іншого – створює умови для його надання. Можна зробити висновок, що необхідність кредиту викликана існуванням товарно-грошових відносин. Його передумовою є наявність поточних або майбутніх прибутків у позичальника, а конкретними причинами, що обумовлюють необхідність кредиту – коливання потреби в коштах та джерелах їх формування, як у юридичних, так і фізичних осіб [4].

1.3 Сутність кредиту, його види і форми

Даний термін можна визначити наступним чином: "Кредит – це економічні відносини між юридичними та фізичними особами і державами з приводу перерозподілу вартості на засадах повернення і, як правило, з виплатою процента" [5].

Економічні відносини між сторонами кредитної угоди виникають під час одержання позики, користування нею та її повернення. В цих відносинах завжди беруть участь не менш ніж дві сторони: кредитор – сторона, що передає вартість у грошовій чи натуральній формі в кредит і позичальник – сторона, що зацікавлена в одержані позики для досягнення своєї певної мети. Ці сторони називаються суб’єктами кредитної угоди, а ті грошові чи матеріальні цінності, затрати чи проекти, відносно яких укладається угода позики, – є об’єктом кредиту.

Основними суб’єктами кредитної угоди виступають держава, банківські установи, підприємства і організації різних форм власності та громадяни [4].

Перехід до конкретного банківського співтовариства, що орієнтується на ринок та на комерційний успіх, обумовив різноманіття форм і видів кредитування.

Залежно від форми руху вартості виділяють три основні форми кредиту:

- товарний, що виникає між продавцями і покупцями, коли покупці одержують товари чи послуги з відстрочкою платежу;
- грошовий;
- кредит у вигляді гарантій.

Банки в своїй діяльності використовують другу і третю форму, при цьому остання виражається в зобов’язанні банку гарантувати платіж клієнтові у випадку, коли той не зможе оплатити свої рахунки.

Що стосується товарної форми кредиту, то банк може обслуговувати або проводити операції з інструментом комерційного кредиту – векселем, де банк не являється прямим учасником кредиту у товарній формі, а також може бути учасником лізингу.

Найбільш розповсюджена форма кредиту на Україні – грошова, при організації міжнародних операції та розрахунків – гарантії.

Що стосується видів банківського кредиту, то їх кількість обумовлюється кількістю критеріїв для класифікації (детальніше у таблиці 1.1) [6].

Таблиця 1.1 – Види банківських кредитів

	Критерій (ознака)	Вид кредиту
1	Роль банка (кредитор чи позичальник)	активний
		пасивний
2	Строки користування	до запитання (онкольний)
		строковий: - короткостроковий (до 1 року) - довгостроковий (понад 1 рік)
3	Призначення	кредити торгівельним та промисловим підприємствам
		під нерухомість
		приватним особам
		фінансовим установам
4	Мета або сфера спрямування	на збільшення капіталу (в інвестиційну діяльність)
		на тимчасове поповнення обігових коштів (у сферу обігу)
		на споживчі потреби (споживчий)
5	Наявність та характер забезпечення	бланковий (незабезпечений)
		забезпечений: - заставою (в тому числі ломбардний) - гарантійним зобов’язанням чи порукою - страхуванням
6	Метод кредитування	цільовий кредит
		кредит по поточному рахунку

Кінець таблиці 1.1

	Критерій (ознака)	Вид кредиту
7	Спосіб надання	у разовому порядку
		відповідно до відкритої кредитної лінії
		гарантійні (із заздалегідь обумовленої датою надання)
8	Спосіб погашення	у разовому порядку
		достроково
		з регресією платежів
		після закінчення обумовленого періоду
9	Строк повернення кредиту	строковий
		до запитання
		прострочений
		відстрочений (пролонгований)
10	Кількість кредиторів	двосторонні
		консорціумні
		паралельні
11	За рівнем ризику	стандартні
		під контролем
		субстандартні
		сумнівні
		безнадійні

1.4 Оцінювання кредитоспроможності позичальника

Дотепер серед економістів немає єдиної думки щодо визначення суті такої категорії як кредитоспроможність. Так, автори однієї з методик розуміють під кредитоспроможністю позичальника – «його здатність вчасно й повно розрахуватися за своїми зобов’язаннями», що часто звужує поняття кредитоспроможності до поняття платоспроможності. Автори іншої методики вважають, що «...кредитоспроможність являє собою оцінку банком позичальника з погляду можливості і доцільності надання йому кредиту і визначає ймовірність повернення позик та виплати відсотків за ними в майбутньому».

Отже, кредитоспроможність – дійсний фінансово-господарський стан позичальника який визначає передумови для одержання ним кредитів та його спроможність за конкретних умов кредитування в повному обсязі і у визначений в кредитному договорі строк розрахуватися за своїми борговими зобов’язаннями

лише грошовими коштами, що одержані ним в ході звичайної діяльності. Застосування даного визначення має не лише важливе теоретичне, а й практичне значення, оскільки визначає спрямованість і зміст процесу оцінки кредитоспроможності позичальника, а отже, і його результати [6].

Банк надає кредити на здійснення заходів, передбачених статутом позичальника, на підставі індивідуальної кредитної угоди з урахуванням власного кредитного ризику. Діяльність банків у галузі кредитування має бути спрямована на проведення єдиної грошово-кредитної політики в країні, на зміцнення та стабілізацію національної валюти.

Кредити надаються позичальникам для здійснення заходів, пов'язаних із:

- розвитком поточної виробничої діяльності та товарообігу;
- експортно-імпортними операціями позичальника;
- задоволенням споживчих потреб;
- іншими напрямами функціонування господарської діяльності;

Забороняється використання позичальниками кредитів:

- на покриття збитків;
- придбання цінних паперів будь-яких підприємств. Кредити, які отримані комерційними банками за рахунок централізованих ресурсів Національного банку України, не можуть бути спрямовані:
- на конвертування національної валюти у валюту інших держав;
- викуп державного майна;
- використання в цілях, які не обумовлені кредитною угодою. У разі отримання централізованих ресурсів Національного банку України комерційні банки не мають права здійснювати операції з продажу власних кредитних ресурсів іншим банкам [7].

Банкам в умовах зростання проблемної заборгованості фізичних осіб з метою залучення потенційних позичальників, що мають вибір рівнозначних пропозицій від інших банків, необхідно використовувати більш технологічні способи оцінки кредитоспроможності клієнта, що дозволяють прискорити процес прийняття рішень без шкоди якості кредитного портфеля банку. Все це вимагає

нового підходу до систем управління заявками на кредит, впровадження технологій автоматичного і напівавтоматичного прийняття рішень в залежності від виду продукту, розміру кредиту та типу позичальника, що дозволяють мінімізувати кредитний ризик.

Аналіз практики діяльності банків показує, що єдиної методики для оцінки кредитоспроможності кредитоотримувачів – фізичних осіб немає. Кожен банк самостійно визначає набір використовуваних показників (чинників), а також процедуру оцінки кредитоспроможності клієнта. При кредитуванні населення оцінка надійності позичальників проводиться на основі аналізу таких показників, як доходи і витрати позичальника, ставлення щомісячної суми платежів по кредиту до середньомісячної суми чистих доходів клієнта. Іноді розглядається розмір бюджету прожиткового мінімуму. На основі даних показників розраховується коефіцієнт кредитоспроможності, від значення якого залежить максимально можлива сума кредиту. Інші показники позичальника, як правило, не аналізуються.

Одним із сучасних підходів до мінімізації кредитного ризику, в рамках якого використовуються автоматизовані системи прийняття рішень щодо видачі банківських кредитів і умов кредитування, є кредитний скоринг (credit scoring). Кредитний скоринг здійснюється за допомогою комп'ютерних скорингових систем, що забезпечують класифікацію потенційних позичальників комерційного банку за ступенем кредитоспроможності на основі доступної інформації. У скорингових системах на підставі бази даних, зазначених у заявці позичальника, обчислюється спеціальне число, або кількість балів (score), яке потім використовується при прийнятті рішення щодо приналежності позичальника до того чи іншого класу кредитоспроможності [8].

1.5 Основні поняття задачі кредитного скорингу

Спочатку кредитний скоринг ґрунтувався на спеціальних таблицях, так званих скорингових картах, які конкретним значенням кожного з показників, що характеризують позичальника, ставлять у відповідність певний скоринговий бал.

При цьому граничне значення сумарного скорингового балу, що розділяє заявки клієнтів на прийняті і ті, кому відмовлено у видачі кредиту, називається балом відсікання (cut-offscore). Всі необхідні характеристики в цьому випадку задавалися експертним шляхом. Надалі, у міру розвитку математико-статистичного інструментарію, що використовується для класифікації багатовимірних неоднорідних даних в різних сферах людської життєдіяльності, все нові моделі даних і методи класифікації знаходили застосування в задачах кредитного скорингу. Так, на різних етапах формування даного підходу для класифікації позичальників були запропоновані різноманітні статистичні методи і алгоритми, що розрізняються модельними припущеннями і рівнем апріорної інформації щодо моделей даних, в тому числі: лінійний дискримінантний аналіз, модель множинної лінійної регресії, логіт- і пробіт-моделі бінарного вибору, метод лінійного програмування, метод найближчих сусідів, дерева рішень (класифікаційні дерева), нейронні мережі, генетичні алгоритми [8].

Вибір алгоритмів для включення в скоринговую систему істотно залежить від типу використовуваних статистичних даних про позичальників. Для побудови скорингової системи фізичних осіб в зарубіжній практиці використовуються наступні типи даних:

- статистичні дані щодо соціально-економічного розвитку для тих регіонів, в яких є відділення (представництва, філії) банку або в яких банк планує їх відкрити;
- статистичні дані, що стосуються підприємств регіонів з метою включення в модель інформації про приналежність позичальника до певного сектору економіки;
- анкетні дані по всіх наявних позичальників банку в розрізі повернень і неповернень боргу, а також за простроченими виплатами відсотків і основної суми боргу;
- експертні знання банківського менеджменту по кожному з типів кредитних продуктів банку.

В цілому, можливе поєднання будь-яких комбінацій з перерахованих типів даних. З точки зору побудови або вибору статистичних алгоритмів кредитного скорингу істотним є те, що перераховані вище дані описуються як кількісними (безперервними), так і якісними (дискретними номінальними або ординальними) ознаками. У зв'язку з цим виникає проблема класифікації позичальників в просторі різнотипних ознак, модель яких не може бути описана деяким відомим законом розподілу ймовірностей. Зокрема, наявність дискретних ознак робить свідомо неадекватним припущення про спільний нормальний розподіл класифікаційних ознак, яке є суттєвим для популярного в системах кредитного скорингу лінійного дискримінантного аналізу.

Невисока ефективність систем кредитного скорингу при практичному застосуванні може бути наслідком ряду проблем, що виникають на етапі розробки скорингової моделі, формування навчальної вибірки і навчання (оцінювання параметрів) використовуваних статистичних алгоритмів. До числа таких проблем можна віднести:

1) неоптимальний вибір складу класифікаційних ознак (наявність великої кількості малоінформативних ознак і відсутність важливих класифікаційних ознак);

2) не репрезентативність навчальної вибірки (недостатній обсяг, погано формалізовані класи позичальників);

3) велика різниця в обсягах вибірок, що відповідають різним класам позичальників, наприклад, значне перевищення числа надійних позичальників над числом ненадійних позичальників в навчальній вибірці;

4) наявність аномальних (тих, що різко виділяються) спостережень в навчальній вибірці, а також інших типів неоднорідності, обумовлених не приналежністю позичальників до досліджуваних класів надійності, а іншими факторами неоднорідності (наприклад, типом і терміном кредиту);

5) неточності в використовуваних даних, що носять технічний або навмисний характер і т. д.

Багато банків, незважаючи на пропозиції "готових рішень" в даній області з боку ряду IT-компаній, розробляють власні скорингові системи. У той же час вирішення виникаючих при цьому проблем (включаючи підготовку даних) вимагає від розробників високої кваліфікації в області методів багатовимірного статистичного аналізу даних. Фахівців з даної кваліфікацією поки що недостатньо в аналітичних підрозділах банків [8].

1.6 Математична модель задачі кредитного скорингу

Скоринг являє собою математичну або статистичну модель, за допомогою якої на основі кредитної історії «минулих» клієнтів банк намагається визначити, наскільки велика ймовірність, що конкретний потенційний позичальник поверне кредит в строк.

Нехай потенційний позичальник банку ω характеризується набором індивідуальних показників, з яких утворений N-мірний вектор ознак $x = (x_1, \dots, x_N)' = x(\omega) \in R^N$. Ціллю кредитного скорингу є віднесення потенційного позичальника банку ω до одного з $L \geq 2$ класів $\{\Omega_\alpha\}(\alpha \in S = \{0, \dots L-1\})$, що відрізняються ступенем надійності, на основі аналізу вектору ознак $x(\omega), \omega \in \Omega_0 \cup \dots \cup \Omega_{L-1}$.

Для простоти викладення будемо надалі розглядати випадок $L = 2$ і вважати, що: Ω_0 – клас ненадійних позичальників, що мають низьку ступінь платоспроможності. Відносно позичальника ω із класу Ω_0 очікується повне виконання кредитних зобов'язань. В видачі кредиту позичальнику ω із класу Ω_1 може бути відмовлено, так як відносно нього очікується невиконання у повній мірі кредитних зобов'язань.

Істинний номер класу $d^0 = d^0(\omega) \in S = \{0,1\}$, до якого відноситься позичальник ω, є дискретною випадковою величиною з розподілом ймовірностей $P\{d^0(\omega) = \alpha\} = \pi_\alpha > 0, \alpha \in S;\ \pi_0 + \pi_1 = 1$, де $\pi_0, \pi_1 = 1 - \pi_0$ – апріорні ймовірності класів.

В загальному випадку передбачається, що випадковий вектор ознак $x = x(\omega)$ для об'єктів із фіксованого класу $\Omega_\alpha \in S(d^0 = \alpha)$ описується деякою умовною щільністю розподілу $p_\alpha(x)$.

Задача кредитного скорингу полягає у віднесенні позичальника $\omega \in \Omega$ до одного з класів $\{\Omega_\alpha\}_{\alpha \in S}$ за сукупністю його ознак $x = x(\omega)$, тобто в оцінюванні невідомого (неспостережуваного) номера класу $d^0 = d^0(\omega) \in S$ для позичальника ω по відомому значенню його ознак $x = x(\omega) \in R^N$.

На практиці ймовірнісні характеристики $\{\pi_\alpha, p_\alpha(x)\}_{\alpha \in S}$ класів $\{\Omega_\alpha\}_{\alpha \in S}$ частково або повністю невідомі. Однак банк може мати у своєму розпорядженні кредитну базу даних, що включає інформацію про позичальників, для яких раніше видавалися кредити і класифікація яких на класи $\{\Omega_0, \Omega_1\}$ до теперішнього моменту точно відома. Нехай $X = X_0 \cup X_1 = \{x_1, \dots, x_n\}$ – вибірка значень контрольованих ознак з класів $\Omega_0 \cup \Omega_1$ для позичальників з відомою кредитною історією. Тут $X_\alpha = \{x_{\alpha 1}, \dots, x_{\alpha n_\alpha}\}$ – випадкова вибірка об'єму n_α із розподілу зі щільністю $p_\alpha(x)$, що відповідає класу позичальників $\alpha \in S$. Вибірку X будемо називати класифікованою навчальною вибіркою об'єму $n = n_0 + n_1$. Оскільки навчальна вибірка є класифікованою, то для кожного спостереження x_i з навчальної вибірки $X = \{x_1, \dots, x_n\}$ точно відомий номер класу $d_i^0 = d^0(x_i) \in S$.

Навчальна вибірка X використовується на «етапі навчання» алгоритму кредитного скорингу для статистичного оцінювання імовірнісних характеристик $\{\pi_\alpha, p_\alpha(x)\}_{\alpha \in S}$ і побудови вирішального правила класифікації, яке потім застосовується на "етапі іспиту" для класифікації нових позичальників за відповідними їм спостереженнями $x_{n+1}, x_{n+2}, \dots$.

Методи побудови та конкретний вид вирішального правила класифікації залежать від додаткових модельних припущень щодо ймовірнісної моделі спостережень, які, в свою чергу, обумовлені особливостями реально спостережуваних показників [8].

1.7 Огляд алгоритмів для вирішення задачі кредитного скорингу

Найбільш популярними сьогодні є три основні методи побудови скорингових алгоритмів:

- на основі логістичної регресії;
- на основі дерева класифікації;
- на основі нейронної мережі.

Основна відмінність між цими трьома методами полягає в підходах до способів сегментації прецедентів навчальної вибірки. Сама сегментація має на меті визначити значимі фактори, що впливають на ймовірності можливих результатів кредитних угод, що можливо, якщо між сконструйованими сегментами можна виявити статистично значущу відмінність у співвідношенні позитивних і негативних прецедентів.

У методі логістичної регресії сегментація прецедентів здійснюється на основі розбиття факторного простору n-мірної сіткою, де n – кількість значущих чинників (рисунок 1.1).

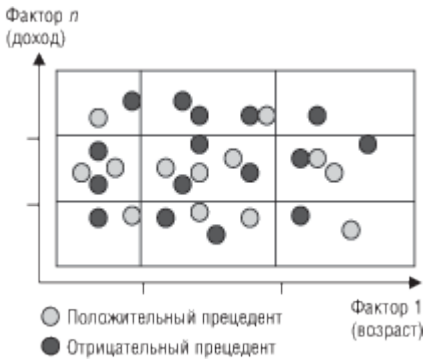

Рисунок 1.1 – Сегментація прецедентів у методі логістичної регресії

В якості вихідного припущення приймається, що кожна клітинка сітки (n-мірний прямокутник) об'єднує прецеденти з навчальної вибірки, що характеризуються однаковою ймовірністю результату. Координати вузлів цієї сітки розраховуються на підставі статистичних критеріїв, виходячи з принципу максимальності відмінності між вірогідністю результатів кредитних угод для суміжних сегментів прецедентів. Співвідношення позитивних і негативних

прецедентів в кожному сегменті використовується для розрахунку скоринг-балів в скоринговій карті, а координати вузлів сітки в факторному просторі якраз і задають інтервали значень ознак в скоринговій карті. Логістична регресія є, таким чином, адекватним математичним інструментом для розрахунку скорингових карт.

Дерево класифікацій (дерево рішень) є більш загальним алгоритмом сегментації навчальної вибірки прецедентів, ніж логістична регресія. На відміну від методу логістичної регресії, в методі дерева класифікації сегментація прецедентів задається не за допомогою n-мірної сітки, а шляхом послідовного дроблення факторного простору на вкладені прямокутні області (рисунок 1.2).

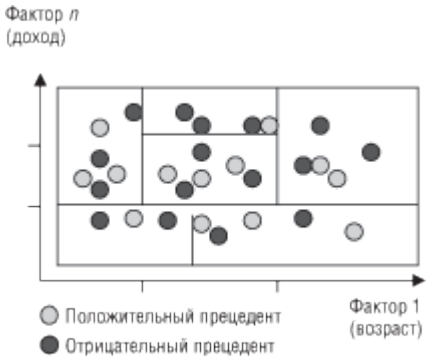

Рисунок 1.2 – Сегментація прецедентів в методі дерева класифікації

При цьому дотримується наступна послідовність кроків (рисунок 1.3).

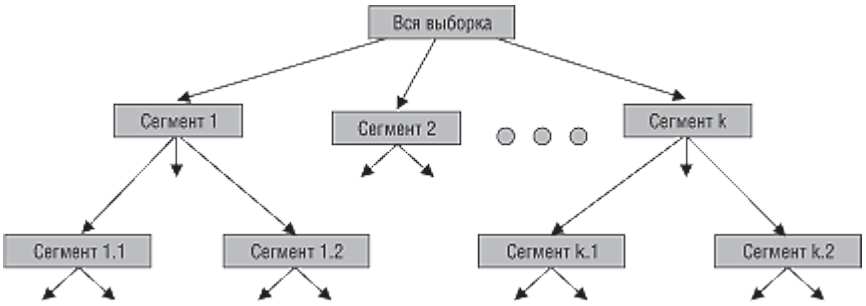

Рисунок 1.3 – Послідовність кроків в методі дерева класифікації

На першому кроці поділ вибірки прецедентів на сегменти проводиться по самому значимому чиннику. На другому і наступних кроках щодо кожного з

отриманих раніше сегментів процедура повторюється до тих пір, поки ніякий варіант подальшого дроблення не призводить до істотної відмінності між співвідношенням позитивних і негативних прецедентів в нових сегментах. Кількість розгалужень (сегментів) на кожному кроці процедури побудови дерева рішень вибирається автоматично.

Нейронна мережа дозволяє обробляти прецеденти навчальної вибірки з більш складним (ніж прямокутники) видом сегментів (рисунок 1.4). Геометрична форма сегментів буде істотно залежати від внутрішньої структури нейронної мережі, яка може бути налаштована з урахуванням характеру взаємозв'язків між враховуються факторами.

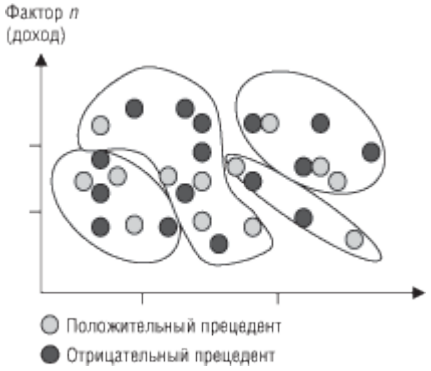

Рисунок 1.4 – Сегментація прецедентів нейронною мережею

Хоча ні дерево рішень, ні нейронна мережа не призводять до побудови скорингової карти в її класичному табличному вигляді, аналог скоринг-балів легко може бути отриманий і для цих методів. Як скоринг-бали можуть виступати, наприклад, емпірично розраховані частки позитивних прецедентів в сегменті. І тоді завдання розрахунку скоринг-бала апліканту рівносильна задачі віднесення апліканта до одного з побудованих сегментів, що і робиться в результаті застосування побудованих скорингових алгоритмів до нового апліканту.

З практики прогнозного моделювання відомо, що жоден з описаних методів не може бути визнаний «найкращим» в усіх випадках. І тільки зіставлення прогнозу і факту може дати оцінку ефективності скорингових моделей. Для

порівняння можна взяти всю або частину наявної емпіричної прецедентної вибірки [9].

1.8 Препроцесінг даних

Далеко не завжди вихідні дані мають чітку структуру. А, всупереч громадській думці, машинне навчання не працює автономно і самостійно. Для адекватного функціонування цього інструменту, як і будь-якого IT-засобу, необхідні чітко визначені вихідні дані і інструкції. Неможливо завантажити в алгоритм машинного навчання всі накопичені великі дані різних форматів і отримати на виході коректні результати. Крім того, вихідні дані часто спотворені і ненадійні: в них можуть бути присутніми значення, що виходять за межі допустимих діапазонів (шуми), аномальні значення (викиди), а також пропуски (відсутність значень).

У Data Mining не випадково виділяють підготовку даних у окрему фазу. Data Preparation – досить трудомісткий ітеративний процес, який займає до 80% всіх витрат ресурсів і часу в життєвому циклі Data Mining і включає такі задачі опрацювання вихідних («сирих») даних [10]:

- Вибірка даних – відбір ознак (features або предикторов) і об'єктів з урахуванням їх релевантності для цілей Data Mining, якості і технічних обмежень (обсягу і типу).
- Очищення даних – видалення помилок, неправильних значень (наприклад, число в строковому параметрі і ін.), відсутніх значень (Missing values або NA), виключення дублікатів і різних описів одного і того ж об'єкта, відновлення унікальності, цілісності і логічних зв’язків.
- Генерація ознак – створення похідних ознак і їх перетворення в вектори для моделі машинного навчання, а також трансформація для підвищення точності алгоритмів машинного навчання.
- Форматування – синтаксичні зміни, які не змінюють значення даних, але потрібні для інструментів моделювання, наприклад, сортування в певному

порядку або видалення непотрібних знаків пунктуації в текстових полях, обрізка «довгих» слів, округлення дійсних чисел до цілого і т.д [11].

1.8.1 Вибірка даних

Результатом проходження етапу підготовки даних є оброблений набір очищених даних, придатних для обробки алгоритмами машинного навчання. Така вибірка, що називається датасетом (dataset), потрібна для тренування моделі Machine Learning, щоб навчити систему і потім використовувати її для вирішення реальних завдань. Однак, оскільки в процесі навчання необхідно оцінювати якість моделі, розрізняють кілька типів вибірок.

Dataset для машинного навчання – це оброблена і структурована інформація в табличному вигляді. Рядки такої таблиці називаються об'єктами, а стовпці – ознаками. Розрізняють 2 види ознак:

- незалежні змінні – предиктори;
- залежні змінні – цільові ознаки, які обчислюються на основі одного або декількох предикторов.

Опис ознак більш характерний для задач класифікації, коли є вибірка – кінцева безліч об'єктів, для яких відомо, до яких класів вони належать. Класова приналежність інших об'єктів невідома. В процесі машинного навчання будується модель, здатна класифікувати довільний об'єкт з початкової множини. Практичний сенс завдань класифікації полягає в передбаченні можливих результатів на основі сукупності вхідних змінних, наприклад, діагностика захворювань, попередня оцінка ефективності родовищ корисних копалин, кредитний скоринг, розпізнавання мови, прогнозування відтоку клієнтів і т.д.

В залежності від варіанту завдання класифікації, цільова ознака може виглядати по-різному:

- один стовпець з двійковими значеннями (1/0, TRUE / FALSE і ін.): двох-класова класифікація (binary classification), коли кожен об'єкт належить тільки одному класу;

- декілька стовпців з двійковими значеннями: завдання класифікації з пересічними класами (multi-label classification), коли один об'єкт може належати кільком класам;
- один стовпець з дійсними значеннями: регресійний аналіз, коли прогнозується одна величина;
- декілька стовпців з дійсними значеннями: завдання множинної регресії, коли прогнозується кілька величин.

Для кожного етапу машинного навчання необхідний свій набір даних:

- для безпосереднього навчання моделі потрібна навчальна вибірка (training sample), по якій проводиться настройка (оптимізація параметрів) алгоритму;
- для оцінки якості моделі використовується тестова (контрольна) вибірка (test sample), яка, в ідеальному випадку, не повинна залежати від навчальної;
- для вибору найкращої моделі машинного навчання знадобиться перевірочна (валідаційна) вибірка (validation sample), яка також не повинна перетинатися з навчальною [15].

1.8.2 Очищення даних

Очищення даних – процес обробки вибірки для інтелектуального аналізу інформації (Data Mining) за допомогою алгоритмів машинного навчання. Цей етап, на якому виконується виявлення і видалення помилок і невідповідностей в даних з метою поліпшення якості датасета, також називається data cleaning, data cleansing або scrubbing. Некоректна, втрачена інформація або та, що дублюється, може стати причиною неадекватної статистики [13] і невірних висновків в контексті бізнесу. Тому очищення даних є обов'язковою процедурою Data Preparation.

Існують наступні 2 види проблем з даними, від яких позбавляє процедура їх очищення [12]:

- проблеми з ознаками – значеннями змінних, стовпцями в табличному поданні датасета;
- проблеми із записами – об'єктами, які є рядками датасета і описуються значеннями ознак.

На рівні ознак виділяють 6 основних проблем:

1) неприпустимі значення, які лежать поза потрібного діапазону, наприклад, цифра 7 в полі для шкільних оцінок за п'ятибальною шкалою;

2) відсутні значення, що не введені, безглузді або не визначені, наприклад, число 000-0000-0000 як телефонного номера;

3) орфографічні помилки – неправильне написання слів: «водітл» замість «водій» або «Омськ» замість «Томськ», що спотворює первинний сенс змінної, підставляючи замість одного міста інший;

4) багатозначність: використання різних слів для опису одного і того ж за змістом значення, наприклад, «водій» і «шофер» або застосування однієї абревіатури для різних за змістом значень;

5) перестановка слів, зазвичай зустрічається в текстових полях вільного формату;

6) вкладені значення – кілька значень в одній ознаці, наприклад, в полі вільного формату.

На рівні записів виділяють 4 основні проблеми:

1) порушення унікальності, наприклад, паспортного номера або іншого ідентифікатора;

2) дублювання записів, коли один і той же об'єкт описаний двічі;

3) суперечливість записів, коли один і той же об'єкт описаний різними значеннями ознак;

4) невірні посилання – порушення логічних зв’язків між ознаками [12].

1.8.3 Генерація ознак

Генерація ознак – мабуть, самий творчий етап підготовки даних (Data Preparation) для машинного навчання. Цей етап ще називають Feature Engineering. Він настає після того, як вибірка сформована і очищення даних завершено.

Всі ознаки у датасеті можуть бути наступних видів:

1) Бінарні, які приймають два значення, наприклад, {true, false}, {0,1}, {1,1}, { «так», «ні»} і т.д ..

2) Категоріальні (номінальні, факторні), які мають кінцеве кількість рівнів, наприклад, категорія «день тижня» має іменованих 7 рівнів: понеділок, вівторок і т. д. Категорії можуть бути впорядкованими та неупорядкованими. Наприклад, категорія «час доби» має 24 рівня і він впорядкований. Категорія «район міста» з 32 рівнями не впорядкована, оскільки всі рівні мають рівну значущість. Якщо категорія впорядкована, це варто явно вказати при його оголошенні.

3) Кількісні (числові) значення в діапазоні від мінус нескінченності до плюс нескінченності. Ознаки можуть вилучатись з даних будь-якого типу, в т.ч. з тексту, зображень та геоданих.

Генерація ознак включає в себе 3 взаємопов'язані завдання:

- витяг ознак (feature extraction) – перетворення даних, специфічних для предметної області, в зрозумілі для моделі числові вектори. Зокрема, саме тут виконується токенізація і лематизації текстів, обробки зображень і геоданих;
- перетворення ознак (feature transformation) – зміна даних для підвищення точності алгоритму, наприклад, нормалізація або зміна імовірнісного розподілу, видалення мультиколінеарності і т.д. ;
- відбір ознак (feature selection) – відсікання непотрібних ознак за допомогою алгоритмів машинного навчання, які дозволяють оцінити важливість предиктору, наприклад, жадібний алгоритм, логістична регресія, випадковий ліс, градієнтний бустинг [14].

1.8.3.1 Підготовка категоріальних даних

В останні роки з'явилися завдання, в яких майже всі або навіть всі ознаки категоріальні. Як було сказано у пункті 1.5, різнотипні ознаки призводять до неточності прогнозу моделі у задачі класифікації, тому потрібно приділити особливу увагу перетворенню категоріальних даних у зрозумілі моделі дані.

Самі значення категоріальних ознак для алгоритмів аналізу даних марні: на практиці різні категорії кодують різними цілими числами, але використовувати на таких даних класичні методи машинного навчання, орієнтовані на речові ознаки (і операції над ними), не можна. Для категоріальних ознак має сенс лише операція порівняння (збігаються чи ні категорії).

У прикладних задачах переважають речові ознаки. Якщо в задачі є категоріальні ознаки, то їх, як правило, небагато, тому вдається підібрати підходяще кодування в ознаки, над якими вже можна виконувати різні арифметичні операції [16].

Найбільш популярними способами кодування категоріальних ознак є:

1) Label Encoding – кодувальник даних; метод fit цього класу знаходить все унікальні значення і будує таблицю для відповідності кожної категорії деякому числу, а метод transform безпосередньо перетворює значення в числа. При використанні цього підходу аналітики завжди повинні бути впевнені, що ознака не може приймати невідомих раніше значень.

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	ca
0	26	student	single	high.school	no	no	no	telephone	jun	mon	901	1
1	46	admin.	married	university.degree	no	yes	no	cellular	aug	tue	208	2
2	49	blue-collar	married	basic.4y	unknown	yes	yes	telephone	jun	tue	131	5
3	31	technician	married	university.degree	no	no	no	cellular	jul	tue	404	1
4	42	housemaid	married	university.degree	no	yes	no	telephone	nov	mon	85	1

Рисунок 1.5 – Приклад роботи Label Encoding: вихідні дані

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign
0	26	8	2	3	0	0	0	1	4	1	901	1
1	46	0	1	6	0	2	0	0	1	3	208	2
2	49	1	1	0	1	2	2	1	4	3	131	5
3	31	9	1	6	0	0	0	0	3	3	404	1
4	42	3	1	6	0	2	0	1	7	1	85	1

Рисунок 1.6 – Приклад роботи Label Encoding: перетворені дані

Основна проблема такого перетворення полягає в тому, що числовий код створив евклідове уявлення для даних. Наприклад, неявним чином була введена алгебра над значеннями «job» – ми можемо відняти роботу клієнта 1 з роботи клієнта 2. Звичайно ж, ця операція не має ніякого сенсу. Але саме на цьому засновані метрики близькості об'єктів, що робить безглуздим застосування методу найближчого сусіда на даних в такому вигляді. Аналогічним чином, ніякого сенсу не матиме застосування лінійних моделей.

Для того, щоб застосовувати лінійні моделі на таких даних, нам необхідний інший метод, який називається One-Hot Encoding.

2) One-Hot Encoding – кодувальник даних для перетворення категоріальних або текстових даних в числа, які алгоритми машинного навчання розуміють краще.

Припустимо, що деяка ознака може приймати 10 різних значень. В цьому випадку One-Hot Encoding має на увазі створення 10 ознак, всі з яких дорівнюють нулю за винятком одного.

	0	1	2	3	4	5	6	7	8	9	...	43	44	45	46	47	48	49	50	51	52
0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
4	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0

Рисунок 1.7 – Приклад роботи One-Hot Encoding: перетворені дані

3) Хешування ознак (Hashing trick)

Реальні дані можуть виявитися набагато більш динамічними, і не завжди можна розраховувати, що категоріальні ознаки не матимуть нових значень. Все це сильно ускладнює використання вже навчених моделей на нових даних. Крім того, LabelEncoder має на увазі попередній аналіз всієї вибірки і зберігання побудованих відображень в пам'яті, що ускладнює роботу в режимі великих даних.

Для вирішення цих проблем існує більш простий підхід до векторизації категоріальних ознак, заснований на хешуванні, відомий як hashing trick [17].

Hashing trick дозволяє використовувати вектори функцій змінної величини зі стандартними алгоритмами навчання (регресія, випадкові ліси, нейронні мережі з прямим зворотним зв'язком, SVM, факторизація матриць тощо). Існує безліч різних типів хеш-функцій, але всі вони роблять одне і те ж: зіставляють дані довільних розмірів із даними фіксованого розміру. Єдиним реальним недоліком є той факт, що зворотний пошук (вихід на вхід) неможливий, але для багатьох проблем це не є вимогою [18].

1.8.3.2 Підготовка числових даних

Після роботи з пропусками та NaN даними наступними важливими кроками є нормалізація даних, стандартизація та усунення мультиколінеарності.

У розрізі машинного навчання, нормалізація – це процедура попередньої обробки вхідної інформації (навчальних, тестових і валідаційних вибірок, а також реальних даних), при якій значення ознак у вхідному векторі приводяться до деякого заданому діапазону, наприклад, [0 ... 1] або [-1 ... 1] [19].

Необхідність нормалізації вибірок даних зумовлена природою використовуваних алгоритмів і моделей машинного навчання. Вихідні значення ознак можуть змінюватися в дуже великому діапазоні і відрізнятися один від одного на кілька порядків. Припустимо, датасет містить відомості про концентрацію діючої речовини, яка вимірюється в десятих або сотих частках

відсотків, і показники тиску в сотнях тисяч атмосфер. Або, наприклад, в одному вхідному векторі присутня інформація про вік і доходи клієнта. Будучи різними за фізичним змістом, дані сильно розрізняються між собою за абсолютними величинами. Робота аналітичних моделей машинного навчання (нейронних мереж, карт Кохонена і т.д.) з такими показниками виявиться некоректною: дисбаланс між значеннями ознак може викликати нестійкість роботи моделі, погіршити результати навчання і уповільнити процес моделювання. Зокрема, параметричні методи машинного навчання зазвичай вимагають симетричного і унімодального розподілу даних. Популярний метод найближчих сусідів, часто використовуваний в задачах класифікації та іноді в регресійному аналізі, також чутливий до діапазону змін вхідних змінних.

Після нормалізації всі числові значення вхідних ознак будуть приведені до однакової області їх зміни – деякого вузького діапазону. Це дозволить звести їх разом в одній моделі машинного навчання і забезпечить коректну роботу обчислювальних алгоритмів [20].

Основними формулами, що проводять нормування і стандартизацію, є:

1) Область значень – [0,1]. Рекомендується використовувати, якщо значення початкових даних рівномірно заповнюють область дослідження. Для деяких методів прогнозування формула неефективна у випадку рівності значень нулю або їх зосередження біля кінців відрізка [0,1].

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{1.1}$$

2) Аналогічне першому, але дозволяє зворотно пропорційно розвернути шкалу, що зручно у випадках, коли більшість характеристик мають максимізуватися, а дана характеристика – мінімізуватися. Недоліки ті ж самі.

$$x_i' = \frac{\max(x_i) - x_i}{\max(x_i) - \min(x_i)} \tag{1.2}$$

3) Дане перетворення відрізняється тим, що отримані в результаті його застосування значення є безрозмірними, знаходяться з дисперсією та СКВ,

рівними 1, на відрізку $\left[\frac{x_{\min}-\bar{x}}{\sigma_x};\frac{x_{\max}-\bar{x}}{\sigma_x}\right]$ переважно в околі нуля. У перетвореній використовується $\bar{x}_i$ – вибіркове середнє значення $i-$тої змінної, σ_{x_i} – її ж вибіркове середньоквадратичне відхилення.

$$x_i' = \frac{x_i - \overline{x_i}}{\sigma_{x_i}} \tag{1.3}$$

4) Область значень $[-1;1]$. Формула зручна для використання при прогнозуванні із застосуванням нейронних мереж, в яких активаційною функцією є гіперболічний тангенс. Має всі ті ж переваги й недоліки, що і перетворення (1.1) та (1.2).

$$x_i' = \frac{2(x_i - \min(x_i))}{\max(x_i) - \min(x_i)} - 1 \tag{1.4}$$

5) Область значень $(0;1)$. Використовується рідко, здебільшого для значного підсилення реакції на зміни значень в околі нуля. Функція є допоміжною, вона не позбавляє розмірності значення факторів.

$$x_i' = \frac{1}{1+e^{-x_i}}. \tag{1.5}$$

Мультиколінеарність – кореляція незалежних змінних, яка ускладнює оцінку і аналіз загального результату. Коли незалежні змінні корелюють один з одним, говорять про виникнення мультиколінеарності.

У машинному навчанні мультиколінеарність може стати причиною перенавчання моделі, що призведе до невірного результату. Крім того, надлишкові коефіцієнти збільшують складність моделі машинного навчання, а значить, час її тренування зростає. Також мультиколінеарність факторів погана тим, що математична модель регресії містить надлишкові змінні, а це значить:

- інтерпретація параметрів множинної регресії як величин дії факторів стає складнішою, параметри регресії втрачають сенс і слід розглядати інші змінні;

- оцінки параметрів ненадійні – виходять великі стандартні помилки, які змінюються зі зміною обсягу спостережень, що робить модель регресії непридатною для прогнозування.

Для оцінки мультиколінеарності використовується матриця парних коефіцієнтів кореляції, у якій необхідно обчислити визначник. При цьому можливі наступні ситуації:

- у зовсім не корельованих факторів матриця парних коефіцієнтів кореляції одинична, у якій всі елементи поза її головною діагоналлю дорівнюють нулю;
- якщо між факторами визначилася абсолютно лінійна залежність і всі коефіцієнти кореляції дорівнюють одиниці, то визначник такої матриці дорівнює нулю;
- чим визначник менше (ближче до нуля), тим сильніше мультиколінеарність факторів і ненадійніші результати множинної регресії;
- чим визначник ближче до одиниці, тим менше мультиколінеарність факторів.

Дві змінні колінеарні, коли вони знаходяться між собою в лінійній залежності, якщо коефіцієнт кореляції більше 0,7. Щоб позбутися від мультиколінеарності, необхідно виключити з моделі один з чинників [21].

1.9 Методи вибору оптимального набору інформативних ознак

З розвитком науки і технологій машинного навчання, а також з ростом обсягів досліджуваних масивів даних стає все більш актуальною задача зниження розмірності простору ознак. Рішення задачі зниження розмірності залежить від цілей дослідження, і єдино вірного підходу до вирішення даного завдання, мабуть, не існує.

Традиційно при вирішенні прикладних задач машинного навчання, таких як розпізнавання зображень, прогнозування, задачі скорингу та ін., об'єкти з навчальної колекції характеризуються великою кількістю ознак, числових або категоріальних. Наприклад, в завданнях регресії визначення статистичної

взаємозв'язку між ознаками (предикторами) і цільовою змінною дозволяє зробити висновки про значимість тієї чи іншої ознаки для передбачення цільової змінної. З іншого боку, в задачах класифікації зображень розмірність простору, що описує об'єкти-зображення, може досягати декількох десятків тисяч. В такому випадку побудова статистичних висновків за даними стає не тільки трудомісткою, але і менш точною в силу можливої наявності шумових ознак.

Крім зменшення обчислювальної складності та підвищення точності алгоритмів машинного навчання, актуальність завдання зниження розмірності продиктована ще й необхідністю візуалізації даних. Так, при вирішенні задач кластеризації виявляється вельми наочним відображення об'єктів на дво- або тривимірних діаграмах, особливо якщо в обраних змінних кластери виявляються візуально помітні [22].

Задача вибору оптимального набору ознак полягає в тому, щоб вибрати таку підмножину ознак з вихідного набору ознак, що точність класифікатора, навченого на цьому підмножині ознак, буде максимальною (по всьому підмножини вихідного безлічі ознак).

Важливо відзначити, що задача вибору оптимального набору ознак відрізняється від задачі конструювання ознак.

Конструювання ознак (feature construction) вирішує подібну задачу зменшення розмірності вихідних даних, однак різниця полягає в тому, що методи конструювання ознак формують нові ознаки на основі вже наявних. До методів конструювання ознак відносяться такі методи, як:

- метод головних компонент (PCA, principal component analysis) і його різновиди;
- кластеризація ознак (як нові ознаки обираються центри кластерів);
- автокодувальник (спеціальний вид нейронної мережі);
- регуляризований випадковий ліс (RRF, regularized random forest);
- спектральні і хвильові перетворення ознак [23].

Існує кілька підходів до вибору оптимальних інформаційних признаків. За поширеною класифікацією [24], існує три основних категорії методів вибору

оптимальних інформаційних ознак: методи-обгортки (Wrappers), методи-фільтри (Filters) і вбудовані методи (Embedded).

1.9.1 Методи-обгортки (Wrappers)

Методи-обгортки використовують алгоритм навчання як "чорний ящик" для оцінки конкретних підмножин ознак і вибирають підмножини інформативних ознак на основі їх інформативності для алгоритму навчання.

У цьому розділі будуть прийняті наступні позначення:

X – вихідна підмножина ознак, $|X| = n$

X' – підмножина вихідної множини ознак, $X' \subseteq X$

$\bar{X}$ – множина всіх підмножин вихідної множини ознак, $\bar{X} = \bigcup_{X' \subseteq X} X'$

$J(X)$ – функція оцінки підмножини ознак, яка повинна бути максимізована: $J: X' \subseteq X \to R$, при цьому $J(X)$ називається монотонною, якщо для підмножин S_1 та S_2 виконано: $S_1 \subseteq S_2 \Rightarrow J(S_1) \geq J(S_2)$

Принцип методів обгортки полягає в наступному:

- виконується пошук по простору підмножин вихідного безлічі ознак;
- для кожного кроку пошуку використовується інформація про якість навчання на поточному підмножині ознак.

Отже, для застосування методів-обгорток необхідно, по-перше, вибрати стратегію пошуку по простору підмножин вихідної безлічі ознак, і, по-друге, вибрати функцію оцінки якості навчання на поточній підмножині ознак [23].

Існує три основних категорії алгоритмів пошуку в застосуванні до задачі вибору оптимального набору інформаційних ознак [25]. Це експоненціальні, послідовні і рандомізовані алгоритми пошуку.

1) Експоненціальні алгоритми пошуку

В цю категорію відносять алгоритми, чия складність за кількістю операцій вибору наступного елемента безлічі є $O(2^n)$. Це означає, що на кожному кроці вони вибирають більше одного наступного стану. До цієї категорії відносяться, зокрема, такі алгоритми:

1.1) Повний перебір (exhaustive search)

При exhaustive search ефективність алгоритму машинного навчання оцінюється за всіма можливими комбінаціями функцій у наборі даних. Цей алгоритм пошуку є найбільш жадібним алгоритмом з усіх методів обгортки, оскільки він випробовує всі комбінації функцій і вибирає найкращі, що забезпечують найкращу продуктивність. Недоліком повного перебору функцій є те, що він може бути повільнішим порівняно з методами «крок вперед» і «назад», оскільки він оцінює всі комбінації функцій [26].

1.2) Алгоритм FOCUS

Виконує перебір спочатку за всіма підмножинами потужності 1, після цього за всіма підмножинами потужності 2 і так далі (або у зворотньому порядку, починаючи з підмножин потужності n).

1.3) Алгоритм гілок і меж (branch and bound)

Застосовується в разі монотонної функції $J(X)$, що гарантує находження оптимальної підмножини X', що максимізує $J(X)$, але при цьому потребує набагато меншу кількість операцій оцінки підмножини.

Цей алгоритм заснований на наступній ідеї: нехай потрібно відібрати множину X_{m_0}, що складається з m_0 ознак, и нехай $\bar{m} = n - m_0,\ Z = \{Z_1, \dots, Z_{\bar{m}}\} = X \backslash X_{m_0}, \bar{J}(Z) = J(X_{m_0})$.

В процесі пошуку будується дерево станів наступним чином:

- кореневої вузол знаходиться на рівні 0, має рівень 0 і відповідає $X' = X$;
- кожен вузол на рівні i має деякий номер j, який відповідає номеру ознаки, що виключається з підмножини, відповідному батьківському вузлу;
- листя дерева відповідають підмножинам X': $|X'| = m_0$;
- глибина дерева становить $\bar{m} + 1$.

2) Послідовні алгоритми пошуку

У цю категорію відносять алгоритми, які на кожному кроці вибирають рівно один наступний стан. Такі алгоритми не можуть йти «назад», повертаючись до вже пройдених станів. Складність таких алгоритмів є $O(n)$. Якщо на кожному

кроці допустити вибір не більше k наступних станів, то складність буде оцінюватися як $O(n^k)$ [23]. До цієї категорії відносяться, зокрема, такі алгоритми:

2.1) Прямий жадібний алгоритм (forward greedy selection, Step Forward Feature Selection)

На першому етапі алгоритму оцінюється ефективність класифікатора щодо кожної ознаки. Функція, яка працює найкраще, вибирається з усіх функцій.

На другому етапі перша функція пробується у поєднанні з усіма іншими функціями та вибирається поєднання двох функцій, які забезпечують найкращу продуктивність алгоритму. Процес триває доти, доки не буде вибрано вказану кількість функцій [26].

2.2) Зворотний жадібний алгоритм (backward greedy elimination, Step Backwards Feature Selection)

Зворотний жадібний алгоритм, як випливає з назви, є прямо протилежним прямому жадібному алгоритму.

На першому кроці агоритму одна функція циклічно видаляється із набору ознак і оцінюється ефективність класифікатора. Зберігається набір функцій, що забезпечує найкращу продуктивність. На другому кроці знову видаляється одна характеристика і оцінюється ефективність усіх комбінацій ознак, крім 2-х. Цей процес триває, доки вказана кількість функцій не залишиться в наборі даних.

Емпірично зворотний жадібний алгоритм дає зазвичай кращі результати в порівнянні з прямим жадібним алгоритмом. Це пов'язано з тим, що зворотний жадібний алгоритм враховує ознаки, інформативні в сукупності, але неінформативні, якщо розглядати їх окремо [26].

2.3) Алгоритм сходження на вершину (greedy hill climbing)

Алгоритм є комбінацією двох попередніх методів, починаючи з деякої стартової підмножини X', на кожному кроці переходить в сусідній стан, в якому або з поточної підмножини видаляється один елемент, або в нього додається один елемент. Стартову підмножину X' можна вибрати випадковим чином, задати $X' = X$ або використовувати інші евристики.

Послідовні алгоритми пошуку мають істотний недолік: вони можуть зупинитися в локальному максимумі функції $J(X)$, так і не знайшовши її глобальний максимум.

3) Рандомізовані алгоритми пошуку

У цю категорію відносять алгоритми, що використовують рандомізацію для переходу в наступний стан. Це дозволяє уникнути зупинки в локальних максимумах. Однак час роботи таких алгоритмів заздалегідь невідомо, так як найкраще рішення вони можуть знаходити в загальному випадку як завгодно довго.

До цієї категорії відносяться, зокрема, такі відомі алгоритми:

3.1) Фільтр Лас-Вегас (Las Vegas Filter)

Крок цього алгоритму працює наступним чином:

- розглядає підмножину ознак X', вибрану випадковим чином;
- якщо його функція оцінки $J(X')$ не менше допустимого заздалегідь заданого порогу J_0: $J(X') \geq J$), то, якщо розмір цієї підмножини менше, ніж розмір найкращої знайденої підмножини: $|X'| \leq |Best|$, то нова підмножина вважається найкращою: $Best = X'$;
- якщо ж розмір нової підмножини збігається з розміром найкращої знайденої підмножини: $|X'| = |Best|$, то X' додається до списку найкращих підмножин.

3.2) Алгоритм симуляції відпалу (simulated annealing)

Алгоритм симуляції відпалу моделює фізичний процес, що відбувається при кристалізації речовини. Передбачається, що атоми речовини вже вишикувалися в кристалічну решітку, проте допустимі переходи окремих атомів з одного осередку в іншу. Цей перехід відбувається з певною ймовірністю, яка зменшується з пониженням температури.

Початковий стан X_0 вибирається випадковим чином. Потім, якщо система знаходиться в стані температурного рівноваги (тобто в точці локального максимуму), то ймовірність того, що вона знаходиться в деякому стані X', обчислюється так:

$$P(X') = \frac{\exp(\frac{-E(X')}{kT})}{\sum_{X' \in X} \exp(\frac{-E(X')}{kT})} \quad (1.6)$$

Тут $E(X)$ означає енергію системи, в даній задачі вибирають $E(X) = J(X), T$ – поточна температура системи, k – постійна Больцмана.

Тоді нехай X' – це поточний стан системи, тоді ймовірність її переходу в деякий стан X'' обчислюється таким чином:

$$P(X' \to X'') = \exp(-\frac{J(X'') - J(X')}{kT}) \quad (1.7)$$

При досить низькому T ймовірність переходу системи в будь-який стан стає дуже малою, і система вважається «холодної». Алгоритм на цьому завершується. Таким чином, розумно комбінувати запропонований алгоритм з алгоритмом сходження на вершину: поки система не знаходиться в точці температурного рівноваги, виконувати алгоритм сходження на вершину, а потім виконувати перехід згідно з алгоритмом симуляції відпалу. Це дозволяє уникнути зупинки в локальних максимумах.

3.3) Генетичні алгоритми (genetic approach)

Генетичний алгоритм розглядає елементи множини, за якою здійснюється пошук, як особин з певними генотипами. Генотип для даного завдання є бінарний вектор довжини n, в якому 1 означає наявність ознаки в підмножині ознак, і 0 – його відсутність.

Виконання генетичного алгоритму починається з того, що створюється безліч генотипів початкової популяції. Це зазвичай робиться випадковим чином. Вони оцінюються з використанням фітнес-функції («функції пристосування», fitness function), яка в даній задачі збігається з функцією оцінки $J(X)$.

Потім з отриманої безлічі рішень, з урахуванням значення фітнес-функції, вибираються особини, до яких застосовуються «схрещування» (crossover) і «мутація» (mutation), націлені на рандомізовану зміну особин. Для них також

обчислюється значення фітнес-функції, і потім проводиться «селекція» (selection) – відбір кращих рішень в наступне покоління.

Цей набір дій повторюється ітеративно, так моделюється «еволюційний процес», що триває кілька життєвих циклів (поколінь), поки не буде виконано критерій зупинки алгоритму.

4) Зупинка пошуку

Вище були викладені найбільш використовувані алгоритми пошуку для задачі вибору оптимальних ознак. Тепер розглянемо, яким чином для них можна вибирати критерії зупинки і функції оцінки якості навчання.

Критерій зупинки для алгоритмів пошуку можна вибирати в такий спосіб:

- поточний крок не привів до поліпшення: якщо на k-му кроці була вибрана підмножина X', на $k+1$-му кроці була вибрана підмножина X'', і $J(X'') \leq J(X')$, то алгоритм зупиняється;
- на поточному кроці досягнуті заздалегідь задані умови на розмір підмножини і величину функції оцінки: $|X'| = n', J(X') \geq J_0$, де n' і J_0 задані заздалегідь.

5) Функції оцінки якості навчання

Як правило, методи-обгортки для оцінки якості навчання використовують точність класифікатора. Для оцінки точності класифікатора можна використовувати такі методи:

- Утримання (holdout) – виділення спеціальної підмножини з навчальної множини, що буде використовуватися для тестування класифікаторів.

Одна з проблем цього методу полягає в тому, що значна частина даних не використовується для навчання класифікатора, що знижує ефективність навчання. Інша проблема – невдалий вибір підмножини, що використовується для тестування, може спричинити дуже неточні оцінки.

- Крос-валідація (cross-validation) і leave-one-out як її окремий випадок.

Метод крос-валідації полягає в тому, що навчальна вибірка розбивається K різними способами на дві підмножини, що не перетинаються: навчальна вибірка $T_a^i, i = \overline{1,K}$ і контрольна вибірка $T_b^i, i = \overline{1,K}$. Для кожного із розбиттів i

навчається класифікатор на навальній вибірці T_a^i і тестується на контрольній вибірці T_b^i, таким чином виходить значення точності q_i. Тоді оцінка точності класифікатора, що отримана за допомогою крос-валідації, буде дорівнювати $Q = \frac{1}{K}\sum_{i=1}^{K} q_i$.

Недолік крос-валідації полягає в тому, що вона може привести до перенавчання класифікатора. Крім того, крос-валідація веде до неповного використання навчальної вибірки для навчання. Нарешті, крос-валідація є обчислювально неефективною процедурою [23].

1.9.2 Методи-фільтри (Filters)

Методи-фільтри не взаємодіють з алгоритмом навчання і вибирають оптимальну підмножину ознак, використовуючи тільки інформацію, отриману із навчальної вибірки.

Методи-фільтри виконуються на етапі перед-обробки, до виконання алгоритму навчання. Методи-фільтри можуть як незалежно оцінювати інформативність ознак для навчання, так і оцінювати підмножину ознак в сукупності. У першому випадку знадобиться визначити значення порогової константи (потрібної для того, щоб відкинути ті ознаки, інформативність яких для алгоритму навчання нижче значення порога). У другому випадку знадобиться проводити пошук по простору підмножини ознак.

Даний підхід має найменшу обчислювальну складність серед розглянутих підходів, а також масштабність і простоту застосування. Крім того, подібні методи показують досить хороші результати на практиці.

Якщо для моделювання статистичної кореляції між ознаками не вистачає обсягу даних, тоді фільтри можуть давати результати гірше, ніж обгортки. На відміну від обгорток, такі методи менш схильні до перенавчання. Вони широко використовуються для роботи з даними високої розмірності, коли методи обгортки вимагають занадто великих обчислювальних потужностей.

1) Взаємна інформація (Mutual information) – статистична функція двох випадкових величин, що описує кількість інформації, що міститься в одній випадковій величині відносно іншої. Взаємна інформація визначається через ентропію і умовну ентропію двох випадкових величин x та y як:

$$I(x, y) = H(x) - H(x \mid y) \tag{1.8}$$

2) Приріст інформації (Information gain)

В аналізі даних і машинному навчанні критерій приросту інформації – це критерій, який використовується для вибору кращого розбиття підмножин в вузлах дерев рішень в алгоритмах навчання.

В процесі навчання дерев рішень проводиться рекурсивне розбиття вузлів на вузли-нащадки, які повинні бути більш однорідними за класовим складом прикладів, що в них потрапили, ніж батьківський вузол. Отже, ентропія дочірніх вузлів повинна бути менше, ніж батьківських, а внутрішня інформація – більше.

Розбиття в кожному вузлі дерева проводиться за певним атрибуту з навчальної множини. Оскільки атрибутів кілька, при кожному розбитті доводиться вирішувати завдання вибору найкращого атрибута. Найкращим атрибутом буде вважатися той, який забезпечить максимально можливе збільшення однорідності класів в дочірньому вузлі щодо батьківського або, що одне і те ж, максимальне зниження ентропії (приріст інформації).

У більшості випадків застосування критерію приросту інформації для визначення значущості атрибутів показує хороші результати. Проблеми виникають, коли атрибут має велику різноманітність унікальних значень. У цьому випадку дерево рішень виявляється схильним до перенавчання [29].

3) Алгоритм Relief

Цей метод випадковим чином вибирає з датасету зразки і оновлює значимість кожної ознаки на основі різниці між обраним екземпляром і двома найближчими до нього об'єктами того ж і протилежного класів. Якщо спостерігається різниця в значеннях ознаки для двох найближчих сусідів одного класу, його важливість знижується, а якщо, навпаки, спостерігається відмінність

між значеннями ознаки для об'єктів різних класів, важливість, відповідно, підвищується.

$$W_i = W_i - (x_i - nearHit_i)^2 + (x_i - nearMiss_i)^2 \quad (1.9)$$

Вага ознаки зменшується, якщо його значення відрізняється для найближчих об'єктів одного і того ж класу більше, ніж для найближчих об'єктів з різних класів; в іншому випадку вага збільшується.

Розширений алгоритм ReliefF використовує зважування ознак і шукає по більшій кількості найближчих сусідів.

4) Критерій хі-квадрат (Chi-squared score)

Перевіряє, чи є суттєва різниця між частотами, що спостерігаються і очікуються, для двох категоріальних змінних. Таким чином, перевіряється нульова гіпотеза про відсутність зв'язку між двома змінними.

$$X^2 = \frac{(Observed\_frequency - Expected\_frequency)^2}{Expected\_frequency} \quad (1.10)$$

Щоб коректно застосовувати критерій хі-квадрат для перевірки зв'язку між різними ознаками з датасету і цільової змінної, необхідно дотримати умови: змінні повинні бути категоріальним, незалежними і повинні мати очікувану частоту більше 5. Остання умова гарантує, що CDF (cumulative density function) статистичного критерію (test statistic) може бути апроксимований за допомогою розподілу хі-квадрат [28].

1.9.3 Вбудовані методи (Embedded)

До цієї групи належать алгоритми, які одночасно навчають модель і відбирають ознаки. Зазвичай це реалізують за допомогою l1-регуляризатора (sparsity regularizer) або умови, що обмежує деякі ознаки.

Переваги цих методів полягають в наступному:

- найкращим чином пристосовані до конкретної моделі;

- немає необхідності виділяти спеціальну тестову підмножину, на якій тестується поріг функції рангу для методів-фільтрів або виконується пошук найкращої підмножини для методів-обгорток;
- як наслідок з попереднього пункту, при використанні цього методу менше ризик перенавчання класифікатора [23].

Вбудовані методи вибору оптимального набору ознак є специфічними для кожної конкретної задачі. Коротке перерахування методів класифікації, які можуть включати в себе етап відбору ознак:

- SVM-RFE (Recursive Feature Elimination, рекурсивна оцінка ознак) – для методу опорних векторів (SVM)

Усунення рекурсивної функції (або RFE) працює шляхом рекурсивного видалення атрибутів та побудови моделі на тих атрибутах, які залишились. Алгоритм використовує точність моделі, щоб визначити, які атрибути (та їх комбінація) найбільше сприяють прогнозуванню цільового атрибута [27].

- RMNL (Random Multinomial logit) – для поліноміальної логістичної регресії.
- SMLR (Sparse Multinomial Logistic Regression, розріджена мультіномінальная логістична регресія)

Цей алгоритм реалізує $l1$-регуляризацію за допомогою ARD (Automatic relevance determination, автоматичне визначення релевантності) в рамках класичної мультиноміальної логістичної регресії. Регуляризація визначає важливість кожної ознаки і обнуляє ті, які не приносять користі для прогнозування.

- ARD (Automatic Relevance Determination Regression, регресія автоматичного визначення релевантності)

Модель використовує Байєсову гребневую регресію (Bayesian Ridge Regression). Вона сильніше зміщує ваги коефіцієнтів в сторону нуля в порівнянні, наприклад, з методом найменших квадратів.

Інші приклади алгоритмів регуляризації: Lasso (реалізує $l1$-регуляризацію), гребнева регресія (реалізує $l2$-регуляризацію), Elastic Net (реалізує $l1$ – і $l2$-регуляризацію) [28].

1.10 Огляд мов програмування для роботи з даними

Існує безліч мов програмування, проте не всі вони підходять для машинного навчання (МН). Спеціаліст з обчислювальної техніки Стенфордського університету Ендрю Нгом дав МН наступне визначення: «наука, яка працює над тим, як навчити комп'ютери функціонувати без явного програмування». Передумови до народження науки з'явилися в 1950-х, проте аж до початку 2000-х вони носили лише теоретичний характер. Справжній прорив стався десятиліття назад, коли МН стало каталізатором розвитку кількох проривних технологій, особливо це стосується штучного інтелекту.

МН в сучасній ітерації затребуване в багатьох галузях промисловості, зростає також попит на продукти і послуги, основою яких є машинні алгоритми. Підприємства застосовують прогностичні можливості МН, прагнучи розробити розпорядчі (prescriptive) методи для прийняття обґрунтованих рішень. Технологія передбачає кілька методів розробки ПЗ на базі машинних алгоритмів, проте найпопулярнішим з них є залучення мов програмування [30].

1) R

Є прямим нащадком старшої мови програмування S, був випущений в далекому 1995 році і з тих пір стає дедалі досконалішим. Написаний на таких мовах як C і Fortran, даний проект сьогодні підтримується Фондом мови R для статистичних обчислень (R Foundation for Statistical Computing).

Переваги:

- Відмінний набір високоякісних предметно-орієнтованих пакетів з відкритим вихідним кодом. R має в своєму розпорядженні пакети практично для будь-якого кількісного і статистичного застосування, яке можна тільки собі уявити. Сюди входять нейронні мережі, нелінійна регресія, філогенетика, побудова складних діаграм, графіків і багато іншого.

- Разом з базовою установкою на додачу можна установити велику кількість вбудованих функцій і методів. Крім того, R прекрасно обробляє дані матричної алгебри.

- Можливість візуалізації даних є важливою перевагою поряд з можливістю використання різних бібліотек, наприклад ggplot2.

Недоліки:

- Низька продуктивність (R не є швидкою мовою програмування).

- Специфічність. R прекрасно підходить для статистичних досліджень і науки про дані, але він не такий гарний, коли справа доходить до програмування для загальних цілей.

- Інші особливості. R має кілька незвичайних особливостей, які можуть збити з пантелику програмістів, які звикли працювати з іншими мовами програмування: індексування починається з 1, використання декількох операторів присвоювання, нетрадиційні структури даних.

R – потужна мова, яка відрізняється наявністю величезного вибору додатків для збору статистичних даних і візуалізації даних, а той факт, що він є мовою програмування з відкритим вихідним кодом, дозволяє йому зібрати велику кількість шанувальників серед розробників. Саме завдяки своїй ефективності для початкових цілей цій мові програмування вдалося досягти широкої популярності.

2) Python

У 1991 році Гвідо ван Россум представив мову програмування Python. З тих пір ця мова стала надзвичайно популярною для загального призначення і широко використовується в співтоваристві фахівців за даними.

Переваги:

- Python – це дуже популярний, це широко використовувана мова програмування загального призначення. Він має великий набір спеціально розроблених модулів і широко використовується розробниками. Багато онлайн-сервісів надають API для Python.

- Python дуже простий у вивченні. Низький поріг входження робить його ідеальною першою мовою для тих, хто займається програмуванням.

- Такі програмні пакети як pandas, scikit-learn і Tensorflow, роблять Python надійним варіантом для сучасних додатків в області машинного навчання.

Недоліки:

- Типобезпека. Python – це динамічна типізована мова, а це значить, що ви повинні бути обережними при роботі з ним. Помилки невідповідності типів (наприклад, передача рядка (string) в якості аргументу методу, який очікує ціле число (integer)) можуть час від часу траплятися.

- Наприклад, в разі якщо є конкретні цілі статистичного аналізу і аналізу даних, то великий набір пакетів мови R дає йому перевагу перед Python. Крім того, існують більш швидкі та безпечні альтернативи Python серед мов програмування.

Python є хорошим варіантом для цілей науки про дані (data science), і це твердження справедливе як для початкового, так і для просунутого рівнів роботи в даній області. Велика частина науки про дані зосереджена навколо процесу ETL (витяг-перетворення-завантаження). Ця особливість робить Python ідеально відповідною для таких цілей мовою програмування. Бібліотеки, такі як Tensorflow від Google, роблять Python дуже цікавою мовою для роботи в області машинного навчання.

3) Java

Java – це надзвичайно популярна мова загального призначення, яка працює на віртуальній машині Java Virtual Machine (JVM). Це абстрактна обчислювальна система, яка забезпечує плавну переносимість між платформами. Вона з самого початку задумувалася як високорівнева і об'єктно-орієнтована мова програмування, який багато в чому нагадує за структурою C ++. В даний час підтримується корпорацією Oracle. Переваги:

- Універсальність. Багато сучасних систем і програм розроблено за допомогою мови Java. Величезною перевагою такої мови є здатність інтегрувати методи науки про дані безпосередньо в існуючу кодову базу.

- Сувора типізація. Забезпечення типобезпеки – не порожній звук для Java, і в разі розробки критично важливих додатків для роботи з великими даними ця особливість як ніколи важлива.

- Java – це високопродуктивна, скомпільована мову загального призначення. Це робить її придатньою для написання ефективного виробничого коду ETL, а також алгоритмів машинного навчання з використанням обчислювальних засобів.

Недоліки:

- «Багатослівність» мови Java робить його не кращим варіантом для проведення спеціальних аналізів і розробки більш спеціалізованих статистичних додатків.

- Java не має великої кількості бібліотек для передових статистичних методів в порівнянні з деякими предметно-орієнтованими мовами, наприклад R.

Багато чого можна сказати на користь вивчення Java як мови для роботи в області науки про дані. Багато компаній оцінять можливість безперешкодної інтеграції готового коду програмного продукту в власну кодову базу, а продуктивність і типобезпека Java є її незаперечною перевагою. Проте, до недоліків такої мови можна віднести той факт, що у нього відсутні набори специфічних пакетів, які доступні для інших мов. Незважаючи на такий недолік, Java є мовою програмування, до якої обов'язково варто приділити увагу, особливо якщо ви вже знаєте R або Python.

4) MATLAB

MATLAB – це визнана мова для чисельних розрахунків, що використовується як в наукових цілях, так і в індустрії. Вона була розроблена і ліцензована MathWorks, компанією, створеною в 1984 році, основною метою якої було комерціалізація програмного забезпечення.

Переваги:

- MATLAB, призначений для чисельних обчислень, добре підходить для використання кількісного аналізу зі складними математичними вимогами,

такими як обробка сигналів, перетворення Фур'є, матрична алгебра і обробка зображень.

- Візуалізація даних. MATLAB має ряд вбудованих можливостей побудови графіків і діаграм.

- MATLAB часто можна зустріти в багатьох курсах бакалаврату з точних наук, таких як фізика, інженерія та прикладна математика. Таким чином, він широко використовується в цих областях.

Недоліки:

- Платна ліцензія. Незалежно від обраного вами варіанту (для наукових, особистих цілей або цілей компанії) вам доведеться розщедритися на дорогу ліцензію.

- MATLAB – це не краща мова програмування для загального призначення.

Завдяки своєму широкому використанню в різних кількісних обчисленнях як для наукових цілей, так і для цілей індустрії, MATLAB став гідним варіантом для застосування в області науки про дані. Він є дуже доречним, якщо для щоденних цілей необхідна інтенсивна, просунута математична функціональність, власне, для чого MATLAB і був розроблений.

5) Julia

Випущена трохи більше 5 років тому, Julia справила враження на світ обчислювальних методів. Мова досягла такої популярності завдяки тому, що кілька великих організацій, включаючи деякі у фінансовій галузі, майже відразу почали використовувати її для своїх цілей.

Переваги:

- Julia – це скомпільована мова JIT («точно в строк»), завдяки якому вдається досягти хорошої продуктивності. Ця мова є досить простою, вона передбачає можливості динамічної типізації та сценаріїв мови, що інтерпретується, такого як Python.

- Julia була призначена для проведення чисельного аналізу, вона також може розглядатися в якості мови програмування загального призначення.

- Читаність. Багато програмістів, що працюють з даною мовою, вважають, що така особливість є її найбільшою перевагою.

Недоліки:

- Незрілість. Оскільки Julia є досить новою мовою, деякі розробники стикаються з нестабільністю під час роботи з його пакетами. Проте, базові засоби мови вважаються стабільними.

- Ще однією ознакою незрілості мови є обмежена кількість пакетів програм, а також невелике число шанувальників серед розробників. На відміну від усталених R і Python, мова програмування Julia не має велику кількість пакетів програм на даний час.

Так, головна проблема мови Julia – це її молодість, оскільки Julia була створена лише недавно, вона поки що не може конкурувати зі своїми основними конкурентами, Python і R [31].

1.11 Обгрунтування вибору мови програмування Python

Хоча не всі знають, що Python не замислювався творцями як мова для аналізу даних, на сьогодні це одна з найкращих мов для статистики, машинного навчання, прогнозної аналітики, а також стандартних завдань з обробки даних.

Найбільш корисними у Python для вирішення задачі класифікації є наступні бібліотеки:

1) NumPy

NumPy дозволяє дуже ефективно обробляти багатовимірні масиви. Багато інших бібліотек побудовані на NumPy, і без неї було б неможливо використовувати pandas, Matplotlib, SciPy або scikit-learn – саме тому вона займає перше місце в списку.

Також в ній є кілька добре реалізованих методів, наприклад, функція random, яка набагато якісніше модуля випадкових чисел зі стандартної бібліотеки. Функція polyfit відмінно підходить для простих завдань прогнозної аналітики, наприклад, для лінійної або поліноміальної регресії.

2) pandas

Аналітики даних зазвичай використовують плоскі таблиці, такі, як в SQL і Excel. Спочатку в Python такої можливості не було. Бібліотека pandas дозволяє працювати з двомірними таблицями на Python. Ця високорівнева бібліотека дозволяє будувати зведені таблиці, виділяти колонки, використовувати фільтри по параметрам, виконувати угруповання за параметрами, запускати функції (складання, знаходження медіани, середнього, мінімального, максимального значень), об'єднувати таблиці і багато іншого. У pandas можна створювати і багатовимірні таблиці.

3) Matplotlib

Візуалізація даних дозволяє представити їх в наочному вигляді, вивчити більш детально, ніж це можна зробити в звичайному форматі, і доступно викласти іншим людям. Matplotlib – найкраща і найпопулярніша Python-бібліотека для цієї мети. Вона не така проста у використанні, але за допомогою 4-5 найбільш поширених блоків коду для простих лінійних діаграм і точкових графіків можна навчитися створювати їх дуже швидко.

4) scikit-learn

Найцікавішими можливостями Python деякі вважають машинне навчання і прогнозну аналітику, а найбільш підходяща для цього бібліотека – scikit-learn. Вона містить ряд методів, що охоплюють все, що може знадобитися протягом перших кількох років в кар'єрі аналітика даних: алгоритми класифікації та регресії, кластеризації, валідацію і вибір моделей. Також її можна застосовувати для зменшення розмірності даних і виділення ознак.

Машинне навчання в scikit-learn полягає в тому, щоб імпортувати правильні модулі і запустити метод підбору моделі. Складніше вичистити, відформатувати і підготувати дані, а також підібрати оптимальні вхідні значення і моделі. Тому перш ніж взятися за scikit-learn, потрібно, по-перше, відпрацювати навички роботи з Python і pandas, щоб навчитися якісно готувати дані, а по-друге, освоїти теорію і математичну основу різних моделей прогнозування та класифікації, щоб розуміти, що відбувається з даними при їх застосуванні.

5) SciPy

Існує бібліотека SciPy і стек SciPy. Більшість вищеописаних бібліотек і пакетів входять в стек SciPy, призначений для наукових розрахунків на Python. Бібліотека SciPy – один з його компонентів, який включає засоби для обробки числових послідовностей, що лежать в основі моделей машинного навчання: інтеграції, екстраполяції, оптимізації та інших. Як і у випадку з NumPy, найчастіше використовується не сама SciPy, а згадана вище бібліотека scikit-learn, яка багато в чому спирається на неї. SciPy корисно знати тому, що вона містить ключові математичні методи для виконання складних процесів машинного навчання в scikit-learn [32].

Саме наявність значної кількості бібліотек і пакетів для аналізу даних робить Python зручним у використанні.

1.12 Вибір моделі для вирішення задачі класифікації

Описана вище бібліотека Scikit-Learn – це Python-бібліотека, вперше розроблена David Cournapeau в 2007 році. У цій бібліотеці знаходиться велика кількість алгоритмів для задач, пов'язаних з класифікацією і машинним навчанням в цілому.

Scikit-Learn базується на бібліотеці SciPy, яку потрібно встановити перед початком роботи.

Scikit-Learn спрощує процес створення класифікатора і допомагає більш чітко виділити концепції машинного навчання, реалізуючи їх за допомогою зрозумілої, добре документованої і надійної бібліотекою.

В контексті машинного навчання класифікація відноситься до навчання з учителем. Такий тип навчання передбачає, що дані, що подаються на входи системи, вже помічені, а важлива частина ознак вже розділена на окремі категорії або класи. Тому мережа вже знає, яка частина входів важлива, а яку частину можна самостійно перевірити. Приклад класифікації – сортування різних рослин на групи, наприклад «папороті» і «покритонасінні». Подібна задача може бути виконана за допомогою Дерева Рішень – одного з типів класифікатора в Scikit-

Learn. При навчанні без учителя в систему подаються непомічені дані, і вона повинна спробувати сама розділити ці дані на категорії.

Процес навчання моделі – це подача даних для нейромережі, яка в результаті повинна вивести певні шаблони для даних. В процесі навчання моделі з учителем на вхід подаються ознаки і масив даних, а при прогнозуванні на вхід класифікатора подаються тільки ознаки. Прийняті мережею дані діляться на дві групи: набір даних для навчання і набір для тестування. Не варто перевіряти мережу на тому ж наборі даних, на яких вона навчалася, так як модель вже буде «заточена» під цей набір.

1.12.1 Алгоритми класифікації бібліотеки Scikit-Learn

Scikit-Learn дає доступ до безлічі різних алгоритмів класифікації. Ось основні з них:

1) Метод k-найближчих сусідів (K-Nearest Neighbors)

Цей метод працює за допомогою пошуку найкоротшої дистанції між тестованим об'єктом і найближчими до нього класифікованими об'єктами з навчального набору. Об'єкт, що класифікується, буде відноситься до того класу, до якого належить найближчий об'єкт набору.

2) Класифікатор дерева рішень (Decision Tree Classifier)

Цей класифікатор розбиває дані на все менші і менші підмножини на основі різних критеріїв, тобто у кожної підмножини своя категорія для сортування. З кожним поділом кількість об'єктів певного критерію зменшується. Класифікація підійде до кінця, коли мережа дійде до підмножини тільки з одним об'єктом.

Якщо об'єднати кілька подібних дерев рішень, то вийде так званий Випадковий Ліс (Random Forest).

3) Наївний байесовский класифікатор (Naive Bayes)

Такий класифікатор обчислює ймовірність приналежності об'єкта до якогось класу. Ця ймовірність обчислюється з шансу, що якась подія відбудеться, з опорою на події, що вже відбулися. Кожен параметр об'єкта, що класифікується, вважається незалежним від інших параметрів.

4) Лінійний дискримінантний аналіз (Linear Discriminant Analysis)

Цей метод працює шляхом зменшення розмірності набору даних, проектуючи всі крапки даних на лінію. Потім він комбінує ці точки в класи, базуючись на їх відстані від центральної точки. Цей метод, як можна вже здогадатися, відноситься до лінійних алгоритмів класифікації, тобто він добре підходить для даних з лінійною залежністю.

5) Метод опорних векторів (Support Vector Machines)

Робота методу опорних векторів полягає в малюванні лінії між різними кластерами точок, які потрібно згрупувати в класи. З одного боку лінії будуть точки, що належать одному класу, з іншого боку – до іншого класу. Класифікатор намагатиметься збільшити відстань між мальованої лініями і крапками на різних сторонах, щоб збільшити свою «впевненість» визначення класу. Коли всі крапки побудовані, сторона, на яку вони падають – це клас, якому ці точки належать.

6) Логістична регресія (Logistic Regression)

Логістична регресія виводить прогнози для точок в бінарному масштабі – нульовому або одиничному. Якщо значення чого-небудь одного або більше 0.5, то об'єкт класифікується в більшу сторону (до одиниці). Якщо значення менше 0.5 – в меншу (до нуля). У кожної ознаки є своя мітка, що дорівнює лише 0 або тільки 1. Логістична регресія є лінійним класифікатором і тому використовується, коли в даних простежується якась лінійна залежність [33].

Вищеописані класифікатори з бібліотеки Scikit-Learn мають достатню точність для побудови прогнозу, та коли ще 10 років назад моделі регресії вважалися безперечними для аналізу з прогнозуванням, зараз існують більш просунуті способи, такі як XGBoost або екстремальний градієнтний бустинг.

1.12.2 XGBoost

Методи, що підсилюють градієнт (Gradient Boosting Machines), входять до категорії машинного навчання, яка називається ансамблевим навчанням (Ensemble Learning), що є галуззю методів машинного навчання, які навчають і

передбачають відразу багато моделей для отримання єдиного чудового результату.

Ансамблеве навчання розбито на три основні підмножини:

- Упаковка (Bagging)

Агрегація Bootstrap або Bagging має дві різні особливості, які визначають її підготовку та прогнозування. Для навчання використовується процедура Bootstrap для розділення навчальних даних на різні випадкові підвибірки, на яких використовуються різні ітерації моделі. Для прогнозування пакувальні класифікатори використовуватимуть прогноз з найбільшою кількістю голосів від кожної моделі, щоб отримати свій результат, а пакувальна регресія займе середнє значення всіх моделей для отримання результату. Завантаження упаковки, як правило, застосовується до моделей з великою дисперсією, таких як дерева рішень, а алгоритм Random Forest – дуже близька варіація щодо упаковки.

- Стекування (Stacking)

Модель стекування – це «метамодель», яка використовує результати вибору з набору багатьох, як правило, суттєво різних моделей, як функції введення. Наприклад, це дозволяє вам навчити метод k-найближчих сусідів, лінійну регресію та дерево рішень з усіма навчальними даними, потім взяти ці результати та об’єднати їх з логістичною регресією. Ідея полягає в тому, що це може зменшити перенавчання і підвищити точність.

- Підсилення (Boosting**)**

Основне визначення підсилення – це метод, який перетворює слабких учнів на сильних, і зазвичай застосовується до дерев. Алгоритм підсилення послідовно додає ітерації моделі, регулюючи ваги тих, хто навчається слабко. Це зменшує зміщення моделі та, як правило, покращує точність. Популярними алгоритмами для підвищення якості є AdaBoost, Gradient Tree Boosting та XGBoost [34].

XGBoost – це скорочення від Extreme Gradient Boosting. Це алгоритм машинного навчання, заснований на дереві пошуку рішень і використовує фреймворк градієнтного бустинга. У завданнях передбачення, які використовують неструктуровані дані (наприклад, зображення або текст), штучна нейронна

мережа вигідно відрізняється від інших алгоритмів або фреймворків. Але коли справа доходить до структурованих або табличних даних, в першості виявляються алгоритми, засновані на дереві пошуку рішень. На рисунку 1.8 можна переглянути еволюцію таких алгоритмів [36].

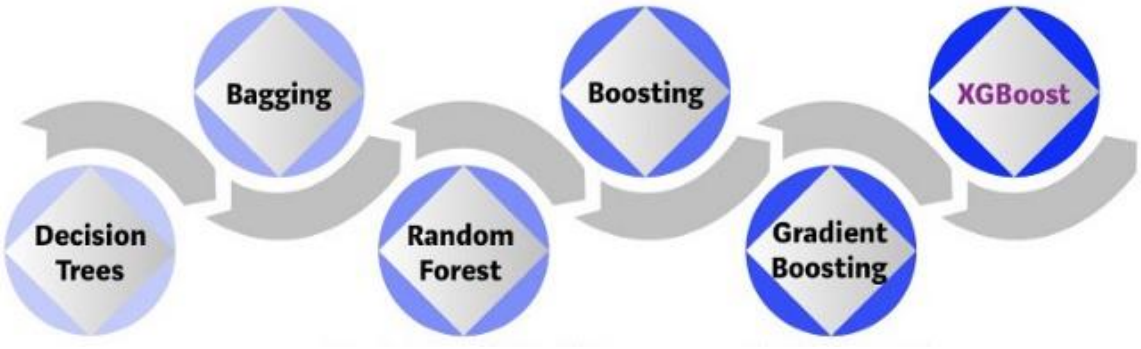

Рисунок 1.8 – Еволюція алгоритмів, заснованих на дереві пошуку рішень

XGBoost забезпечує високоефективну реалізацію алгоритму посиленням стохастичного градієнта (1.11) та доступом до набору гіперпараметрів моделі, призначених для забезпечення контролю за процесом навчання моделі. Це ефективна модель машинного навчання, навіть на наборах даних, де розподіл класів перекошений [35].

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{1.11}$$

Функція Objective вирішується за допомогою апроксимації поліномів Тейлора другого порядку, тобто XGBoost намагається знайти оптимальне вихідне значення для дерева f_t в ітерації t, яке додається для мінімізації вищезазначеної функції втрат у всіх точках даних [37].

XGBoost розроблявся як дослідницький проект Вашингтонського Університету. Tianqi Chen і Carlos Guestrin представили їх роботу на конференції SIGKDD в 2016 році і викликали фурор в світі машинного навчання. З моменту його введення цей алгоритм не тільки лідирував в змаганнях Kaggle, а й був основою декількох галузевих передових додатків. Особливості фреймворка:

- Широка область застосування: може бути використаний для вирішення завдань регресії, класифікації, впорядкування і призначених для користувача завдань на передбачення.
- Сумісність: Windows, Linux і OS X.
- Мови: підтримує більшість провідних мов програмування, наприклад, C ++, Python, R, Java, Scala і Julia.
- Хмарна інтеграція: підтримує кластери AWS, Azure і Yarn, добре працює з Flink, Spark.

XGBoost і Gradient Boosting Machines (GBM) – ансамблі методів дерев, які використовують принцип бустинга слабких учнів (найчастіше, алгоритм побудови бінарного дерева рішень) за допомогою архітектури градієнтного спуску. У свою чергу, XGBoost – поліпшення фреймворка GBM через системну оптимізацію та удосконалення алгоритму, тому він показує таку гарну продуктивність.

Системна оптимізація:

1) Паралелізація

В XGBoost побудова дерев заснована на паралелізації. Це можливо завдяки взаємозмінній природі циклів, що використовується для побудови бази для навчання: зовнішній цикл перераховує листя дерев, внутрішній цикл обчислює ознаки. Знаходження циклу всередині іншого заважає паралелізації алгоритму, так як зовнішній цикл не може почати своє виконання, якщо внутрішній ще не закінчив свою роботу. Тому для поліпшення часу роботи порядок циклів змінюється: ініціалізація проходить при зчитуванні даних, потім виконується сортування, що використовує паралельні потоки. Ця заміна покращує продуктивність алгоритму, розподіляючи обчислення по потокам.

2) Відсікання гілок дерева

У фреймворку GBM критерій зупинки для розбиття дерева залежить від критерію негативної втрати в точці розбиття. XGBoost використовує параметр максимальної глибини max_depth замість цього критерію і починає зворотне

відсікання. Цей "глибинний" підхід значно покращує обчислювальну продуктивність.

3) Апаратна оптимізація

Алгоритм був розроблений таким чином, щоб він оптимально використовував апаратні ресурси. Це досягається шляхом створення внутрішніх буферів в кожному потоці для зберігання статистики градієнта. Подальші поліпшення, як, наприклад, обчислення поза ядром, дозволяють працювати з великими наборами даних, які не поміщаються в пам'яті комп'ютера.

Покращення алгоритму:

1) Регуляризація

Він штрафує складні моделі, використовуючи як регуляризацію LASSO (L1), так і Ridge-регуляризацію (L2), для того, щоб уникнути перенавчання.

2) Робота з розрідженими даними

Алгоритм спрощує роботу з розрідженими даними, в процесі навчання заповнюючи пропущені значення в залежності від значення втрат. До того ж, він дозволяє працювати з різними візерунками розрідженості.

3) Метод зважених квантилів

XGBoost використовує його для того, щоб найбільш ефективно знаходити оптимальні точки поділу в разі роботи зі зваженим датасетом.

4) Крос-валідація

Алгоритм використовує свій власний метод крос-валідації на кожній ітерації. Тобто, нам не потрібно окремо програмувати цей пошук і визначати кількість ітерацій бустинга для кожного запуску [36].

1.13 Оцінка роботи обраного класифікатора

Перед переходом до самих метрик необхідно ввести важливу концепцію для опису цих метрик в термінах помилок класифікації – confusion matrix (матриця помилок).

Матриця помилок – це таблиця або діаграма, що показує точність прогнозування класифікатора щодо двох і більше класів (рисунок 1.9). Прогнози

класифікатора знаходяться на осі X, а результат (точність) – на осі Y. Осередки таблиці заповнюються кількістю прогнозів класифікатора. Правильні прогнози йдуть по діагоналі від верхнього лівого кута в нижній правий [33].

Матриця помилок надає більше уявлення не тільки про ефективність моделі прогнозування, але також про те, які класи прогнозуються правильно, які неправильно та який тип помилок допускається. Найпростіша матриця помилок призначена для двокласової класифікаційної задачі з негативними (клас 0) та позитивними (клас 1) класами [38].

	Negative Prediction	Positive Prediction
Negative Class	True Negative (TN)	False Positive (FP)
Positive Class	False Negative (FN)	True Positive (TP)

Рисунок 1.9 – Приклад побудови матриці помилок

Коли справа доходить до оцінки точності класифікатора, використовуються наступні метрики:

1) Точність класифікації (Accuracy)

Це інтуїтивно зрозуміла метрика, яка показує число правильних прогнозів, поділене на число всіх прогнозів або, простіше кажучи, ставлення правильних прогнозів до всіх.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.12}$$

Хоч цей показник і може швидко дати вам явне уявлення про продуктивність класифікатора, його краще використовувати, коли кожен клас має хоча б приблизно однакову кількість прикладів. В завданнях з нерівними класами ця метрика може показувати некоректні результати [39].

2) Precision, recall и F-міра

Precision (точність) можна інтерпретувати як частку об'єктів, названих класифікатором позитивними і при цьому дійсно є позитивними (1.13), а recall

(повнота) показує, яку частку об'єктів позитивного класу з усіх об'єктів позитивного класу знайшов алгоритм (1.14).

$$precision = \frac{TP}{TP + FP} \tag{1.13}$$

$$recall = \frac{TP}{TP + FN} \tag{1.14}$$

Саме введення precision не дозволяє записувати всі об’єкти в один клас, так як в цьому випадку отримуємо ріст рівня False Positive. Recall демонструє здатність алгоритму знаходити даний клас взагалі, а precision – здатність розрізняти цей клас від других класів.

Precision та recall не залежать, на відміну від accuracy, від співвідношення класів и тому можуть використовуватися в умовах незбалансованих вибірок.

Зазвичай при оптимізації гіперпараметрів алгоритму (наприклад, у разі перебору по сітці GridSearchCV) використовується одна метрика, покращення якої ми і очікуємо побачити на тестовій вибірці. Та існує декілька різних способів поєднати precision та recall в в агрегований критерій якості. F-міра (в загальному випадку F_β) – середнє гармонійне між precision та recall:

$$F_\beta = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall} \tag{1.15}$$

β в даному випадку означає вагу точності в метриці, і при $\beta = 1$ це середнє гармонійне (з множником 2, щоб у випадку precision = 1 та recall = 1 мати $F_1 = 1$).

F-міра досягає максимуму при повноті та точності, що дорівнюють 1, і близька до 0, якщо один з аргументів близький до 0.

Необхідно відмітити, що у випадку задач з незбалансованими класами, які превалюють в реальній практиці, часто доводиться застосовувати техніки штучної модифікації датасету для вирівнювання співвідношення класів.

3) Площа під ROC-кривою (AUC-ROC и AUC-PR)

При конвертації відповіді алгоритму (як правило, ймовірності приналежності до класу) в бінарну метрику, потрібно обрати поріг, при якому 0

стає 1. Доречним і близьким здається поріг, рівний 0,5, але він не завжди є оптимальним, наприклад, при вищезгаданій відсутності балансу класів.

Одним із способів оцінити модель у цілому, не прив'язуючись до конкретного порогу, є AUC-ROC (або ROC AUC) – площа (*A*rea *U*nder *C*urve) під кривою помилок (*R*eceiver *O*perating *C*haracteristic curve). Дана крива представляє собою лінію від (0,0) до (1,1) в координатах True Positive Rate (TPR) та False Positive Rate (FPR):

$$TPR = \frac{TP}{TP + FN} \tag{1.16}$$

$$FPR = \frac{FP}{FP + TN} \tag{1.17}$$

TPR уже відома (повнота), а FPR показує, яку долю з об'єктів negative класу алгоритм спрогнозував невірно. В ідеальному випадку, коли класифікатор не робить помилок (FPR = 0, TPR = 1) отримуємо площу під кривою, що дорівнює одиниці; в іншому випадку, коли класифікатор випадково видає ймовірності класів, AUC-ROC буде прямувати до 0,5, так як класифікатор буде видавати однакову кількість TP та FP.

Кожна точка на графіку відповідає вибору деякого порогу. Площа під кривою в даному випадку показує якість алгоритму (більше – краще), крім того, важливою є крутизна самої кривої – ми хочемо максимізувати TPR, мінімізуючи FPR, а значить, крива в ідеалі повинна йти до точки (0,1).

Критерій AUC-ROC стійкий до незбалансованих класів і може бути інтерпретований к вірогідність того, що випадково обраний positive об'єкт буде проранжований класифікатором вище (буде мати більш високу вірогідність бути positive), чим випадково обраний negative об'єкт.

Важливо відмітити, що на маленьких датасетах площа під PR-кривою може бути занадто оптимістична, тому що розраховується по методу трепецій, але зазвичай в таких задачах даних достатньо [39].

4) Логарифмічні втрати

Значення логарифмічних втрат (англ. Logarithmic Loss) – або просто логлосс – показує, наскільки класифікатор «впевнений» в своєму прогнозі. Логлосс повертає ймовірність приналежності об'єкта до того чи іншого класу, підсумовуючи їх, щоб дати загальне уявлення про «впевненість» класифікатора. Цей показник лежить в проміжку від 0 до 1 – «зовсім не впевнений» і «повністю впевнений» відповідно [33].

Логарифмічні втрати розраховуються як:

$$\log loss = -\frac{1}{l} * \sum_{i=1}^{l} (y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)) \tag{1.18}$$

тут $\hat{y}$ – це відповідь алгоритму на i-му об’єкті, y – істинна мітка класу на i-му об’єкті, а l – розмір вибірки.

Інтуїтивно можна представити мінімізацію логлосс як задачу максимізації accuracy шляхом штрафу за неправильні передбачення. Та необхідно відмітити, що логлосс дуже сильно штрафує за впевненість класифікатора у неправильній відповіді.

Також необхідно відмітити, що похибка на одному об’єкті може давати суттєве погіршення загальної похибки на виборці. Такі об’єкти часто бувають викидами, які потрібно не забувати фільтрувати чи розглядати окремо [39].

Висновки

В даному розділі було розглянуто роль банків у сучасній економіці, наведено означення банку та описано його основні функції, також була розглянута роль кредитних відносин у суспільстві, сутність означення кредиту, його форми і види. Було з’ясовано, за якими чинниками банки оцінюють кредитоспроможність позичальника та способи отримання цих оцінок, розглянуто основні поняття задачі кредитного скорингу та типи даних, що використовуються для побудови скорингової системи, а також загальні проблеми, що призводять до невисокої ефективності таких систем. Було наведено математичну постановку задачі кредитного скорингу та основні алгоритми для вирішення цієї задачі. Було обґрунтовано важливість попередньої обробки даних, яка включає в себе вибірку

даних, їх очищення, генерацію ознак та їх форматування. Детально розглянуто процес генерації ознак, що поділяються на категоріальні та числові, та потребують окремої підготовки. Розглянуто основні методи вибору оптимального набору інформативних ознак (методи-обгортки, методи-фільтри та безпосередньо вбудовані в модель класифікації методи). Було розглянуто сучасні мови програмування для роботи з даними, їх переваги та недоліки, обґрунтовано вибір мови програмування Python та його можливості у сфері аналізу даних. Розглянуто моделі для вирішення задачі класифікації, наведена їх коротка характеристика. Наведено означення алгоритму машинного навчання XGBoost, що заснований на дереві пошуку рішень і використовує фреймворк градієнтного бустинга, описано його основні властивості та переваги над іншими методами класифікації. Розглянуто основні метрики, що використовуються для оцінки точності методу класифікації, їх можливості та обмеження.

СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Постановка задачі

Вхідним значенням для роботи виступає набір певних ознак та відповідні цим ознакам дані. Таким чином, маємо матрицю значень з 553 стовпців та 26824 рядків, частина якої завантажена у дистрибутив Python (Anaconda) та представлена на рисунку 2.1.

	num_1	num_2	num_3	num_4	num_5	num_6	num_7	cat_1	num_8	num_9	...	num_413	cat_132	cat_133	num_414	num_415	num_416	cat
0	1377.3	3712.9	NaN	1303.0	2409.0	1281.7	43.0	1	195.0	NaN	...	11327.5	1	1	28162496.65	0.0	0.07	
1	20.0	13.4	NaN	565.0	NaN	357.2	19.0	1	170.0	NaN	...	168.6	1	1	989383.82	0.0	0.56	
2	150.6	1.8	NaN	2294.0	15.0	107.6	76.0	1	196.0	2.3	...	209.1	1	1	87444.51	0.0	0.01	
3	11.4	178.0	NaN	1236.0	NaN	350.6	41.0	1	117.0	5.3	...	56.9	1	1	297608.00	0.0	0.08	
4	5372.0	3386.0	NaN	1340.0	NaN	14.0	45.0	1	9.0	108.0	...	31.0	1	1	6614247.89	0.0	0.20	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
26819	NaN	NaN	NaN	108.0	NaN	85.6	4.0	1	123.0	1.8	...	181.4	1	1	840801.82	0.0	0.09	
26820	4.2	97.2	NaN	2149.0	130.5	409.3	72.0	1	157.0	NaN	...	596.7	1	1	557554.94	0.0	0.26	
26821	4368.0	5004.0	NaN	2215.0	478.0	3176.0	74.0	1	381.0	140.0	...	0.0	1	2	17821295.19	0.0	0.37	
26822	NaN	NaN	NaN	135.0	NaN	25.7	4.0	1	151.0	NaN	...	1494.9	1	1	7228675.60	0.0	0.00	
26823	28.5	26.3	NaN	44.0	166.7	1706.6	1.0	1	104.0	2.2	...	652.7	1	1	5277923.92	0.0	0.10	

26824 rows × 553 columns

Рисунок 2.1 – Матриця вхідних даних

Задача полягає у розробці інструменту аналізу та прогнозування даних для вирішення задачі кредитного скорингу, який би включав у себе наступні проблеми:

1) аналіз отриманих даних та їх підготовка, а саме:

1.1) видалення або перетворення пропущених даних (NaN);

1.2) видалення «нуль-варіантних» ознак (числових і категоріальних), тобто ознак з єдиним унікальним значенням;

1.3) перетворення категоріальних даних в доступний для розуміння алгоритмом вид;

1.4) нормалізація та стандартизація значень числових ознак;

1.5) усунення мультиколінеарності даних;

1.6) розділення даних на тренувальну та тестову вибірки;

2) початкове навчання моделі з параметрами за умовчуванням та обчислення похибки на тестових даних;

3) навчання додаткових моделей з параметрами за умовчуванням та обчислення похибки на тестових даних;

4) відбір найбільш важливих та інформативних ознак у моделі бустингу для зменшення складності моделі та покращення результатів прогнозу;

5) огляд та перебір параметрів моделі для виявлення більш придатних для даного набору даних;

6) вибір найбільш відповідних параметрів для побудови розв'язку задачі скорингу та аналіз отриманих результатів.

2.2 Аналіз та підготовка отриманих даних

Як можна побачити з рисунку 2.2, дані включають в себе 416 числових та 135 категоріальних ознак. Також присутні стовпці «id» та «gb», де «id» – індивідуальний номер позичальника банку, «gb» – цільова функція, що складається з 0 та 1 (0 – позичальник не отримав кредит, 1 – позичальнику видали кредит).

num_415	num_416	cat_134	cat_135	id	gb
0.0	0.07	1	1	1	0
0.0	0.56	2	1	2	0
0.0	0.01	3	1	3	0
0.0	0.08	4	1	4	0
0.0	0.20	1	1	5	0
...	...	...	...	...	...
0.0	0.09	7	1	5242	0
0.0	0.26	5	1	4320	0
0.0	0.37	3	1	2516	0
0.0	0.00	4	1	4610	0
0.0	0.10	3	1	5243	0

Рисунок 2.2 – Останні стовпці матриці даних

2.2.1 Видалення та перетворення пропущених даних (NaN)

Перш за все, потрібно за наявності знайти та видалити стовпці, що не містять жодних даних. Завдяки можливостям Python та бібліотеці pandas це досить легко зробити всього за декілька рядків:

```
# Видалення та перетворення пропущених NaN даних
# перевіряємо, чи є стовпці, що не містять даних
list_null = data.columns[data.isnull().all()]
print(list_null)
# створюємо новий датасет без стовпців, що не містять даних
data_wt0 = pd.DataFrame(data.drop(list_null, axis = 1))
data_wt0.shape # кількість записів в навчальній виборці
```

```
Index(['num_23', 'num_66', 'num_150', 'num_152', 'num_242', 'num_250',
       'num_258', 'num_266', 'num_342', 'num_365', 'num_375', 'num_392'],
      dtype='object')

(26824, 541)
```

Рисунок 2.3 – Перевірка даних на стовпці, що не містять даних

З рисунку 2.3 видно, що дані мали 12 стовпців без даних, які були вилучені, таким чином скоротивши вибірку з 553 до 541.

```
# підраховуємо кількість NaN даних в кожному стовпці
data_wt0.isnull().sum()
```

```
num_1       3796
num_2       3871
num_3      26761
num_4          8
num_5       9615
           ...
num_416        8
cat_134        0
cat_135        0
id             0
gb             0
Length: 541, dtype: int64
```

Рисунок 2.4 – Кількість NaN даних у кожному стовпці

На рисунку 2.4 бачимо, що деякі стовпці мають багато NaN даних (наприклад, «num_3» має 26761 пусте значення з 26824). Це вказує на те, що при створенні таблиці даних могли виникнути технічні помилки, або на це вплинув людський фактор, наприклад, при заповненні анкети позичальник не вказав ці дані чи ці дані не були знайдені у системі при складанні таблиці. Такі стовпці не несуть інформативності і лише сповільнюють роботу моделі класифікації, тому повинні бути вилучені.

```
# видалимо стовпці, де NaN даних більше 30%
# (параметр thresh вказує на те, скільки потрібно не_NaN значень)
data_wt0 = data_wt0.dropna(axis = 1, thresh = 18777)
data_wt0.shape # кількість записів в навчальній виборці
```

```
(26824, 381)
```

Рисунок 2.5 – Видалення стовпців з NaN даними

Параметр thresh вказує на те, скільки потрібно значень, відмінних від NaN. Визначивши його як 70%, скоротили вибірку з 541 до 381 стовпців.

Надалі робота з підготовки даних буде йтися окремо для категоріальних та числових стовпців, тому потрібно розділити їх на два датасета (рисунок 2.6).

```
# відокремлюємо списки категоріальних та числових стовпців
list_cols = data_wt0.columns.tolist()
list_cat = []
list_num = []
for item in list_cols:
    if not(item.find('cat',0,3) == -1):
        list_cat.append(item)
    elif not(item.find('num',0,3) == -1):
        list_num.append(item)
```

```
# створюємо окремі датасети для категоріальних та числових ознак
data_num = pd.DataFrame(data_wt0[list_num])
data_cat = pd.DataFrame(data_wt0[list_cat])
data_cat.shape, data_num.shape
```

```
((26824, 135), (26824, 244))
```

Рисунок 2.6 – Створення окремих датасетів для категоріальних та числових ознак.

```
# NaN значення в числових даних змінюємо на нулі
data_num = data_num.fillna(0)
```

Рисунок 2.7 – Перетворення NaN даних, що залишились, на нулі

На цьому робота з NaN даними закінчена.

2.2.2 Видалення «нуль-варіантних» ознак

Видалення «нуль-варіативних» ознак (числових і категоріальних), тобто стовпців з єдиним унікальним значенням, є ще одним пунктом підготовки даних, так як такі стовпці не дають інформації та уповільнюють роботу моделі.

```
# Видалення «нуль-варіативних» ознак (числових і категоріальних),
# тобто стовпців з єдиним унікальним значенням,
# так як вони не дають інформації та уповільнюють роботу моделі

# спочатку подивимся на числові дані
for k, v in data_num.nunique().to_dict().items():
    if v == 1:
        print('{} = {}'.format(k,v))
```

```
# подивимся на категоріальні дані
for k, v in data_cat.nunique().to_dict().items():
    if v > 10 or v == 1:
        print('{} = {}'.format(k,v))
```

```
cat_19 = 630
cat_21 = 42
cat_38 = 428
cat_60 = 1313
cat_73 = 19
cat_77 = 22
cat_97 = 22
cat_98 = 1461
cat_124 = 58
```

Рисунок 2.8 – Пошук «нуль-варіативних» ознак

На рисунку 2.8 видно, що серед числових даних немає таких стовпців, що мають лише одне унікальне значення. Серед категоріальних даних потрібно знайти не тільки «нуль-варіативні», а ще й так звані «багато-варіативні» ознаки, так як при їх подальшому перетворенню ці стовпці збільшать вибірку у декілька разів.

Як бачимо, серед категоріальних даних немає ознак з одним унікальним значенням, та є ознаки, що мають більше 10 критеріїв («cat_98» має аж 1461 ознаку). Значна кількість критеріїв може призводити до того, що ознака не матиме важливості для побудови моделі, але вона суттєво її ускладнить. Такі ознаки потрібно видалити.

```
# Видаливо стовпці з кількістю категорій > 10
data_cat = pd.DataFrame(data_cat.drop(['cat_19', 'cat_21', 'cat_38', 'cat_60', 'cat_73', 'cat_77', 'cat_97',
                                       'cat_98', 'cat_124'], axis=1))
data_cat.shape
```

```
(26824, 126)
```

Рисунок 2.9 – Видалення ознак з великою кількістю категорій

2.2.3 Перетворення категоріальних даних

Як було сказано у пункті 1.8.3.1, перетворення категоріальних даних в доступний для розуміння алгоритмом вид є необхідним пунктом підготовки даних, так як різнотипні ознаки призводять до неточності прогнозу моделі у задачі класифікації. Для цього застосуємо один з популярних кодувальників – One-Hot Encoding.

```
# Категоріальні дані: розділення на бінарні стовпці
onehotencoder = OneHotEncoder(categories = 'auto')
onehotencoder.fit_transform(data_cat).toarray()
list_of_names = onehotencoder.get_feature_names(['cat_1','cat_2','cat_3','cat_4','cat_5','cat_6','cat_7','cat_8','cat_9','cat_
           'cat_12','cat_13','cat_14','cat_15','cat_16','cat_17','cat_18','cat_20','cat_22','cat_23','cat_24','cat_25','cat_2
           'cat_27','cat_28','cat_29','cat_30','cat_31','cat_32','cat_33','cat_34','cat_35','cat_36','cat_37','cat_39','cat_4
           'cat_41','cat_42','cat_43','cat_44','cat_45','cat_46','cat_47','cat_48','cat_49','cat_50','cat_51','cat_52',
           'cat_53','cat_54','cat_55','cat_56','cat_57','cat_58','cat_59','cat_61','cat_62','cat_63','cat_64','cat_65',
           'cat_66','cat_67','cat_68','cat_69','cat_70','cat_71','cat_72','cat_74','cat_75','cat_76','cat_78','cat_79',
           'cat_80','cat_81','cat_82','cat_83','cat_84','cat_85','cat_86','cat_87','cat_88','cat_89','cat_90','cat_91',
           'cat_92','cat_93','cat_94','cat_95','cat_96','cat_99','cat_100','cat_101','cat_102','cat_103','cat_104','cat_105',
           'cat_106','cat_107','cat_108','cat_109','cat_110','cat_111','cat_112','cat_113','cat_114','cat_115','cat_116',
           'cat_117','cat_118','cat_119','cat_120','cat_121','cat_122','cat_123','cat_125','cat_126','cat_127','cat_128',
           'cat_129','cat_130','cat_131','cat_132','cat_133','cat_134','cat_135'])
ohe_data = pd.DataFrame(onehotencoder.fit_transform(data_cat).toarray(), columns = list_of_names)
ohe_data.head(5)
```

	cat_1_1	cat_1_2	cat_1_3	cat_1_4	cat_1_5	cat_2_1	cat_2_2	cat_2_3	cat_3_1	cat_3_2	...	cat_134_3	cat_134_4	cat_134_5	cat_134_6	cat_134_7	c
0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
2	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	1.0	0.0	0.0	0.0	
4	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	

5 rows × 437 columns

Рисунок 2.10 – Кодування категоріальних ознак за допомогою One-Hot Encoding

Як можна побачити з рисунку 2.10, кодування виконується у декілька строк, та найбільш витратним було перелічення усіх імен ознак. Це потрібно для того, аби кожен стовпець мав нове ім’я не у вигляді простих чисел (рисунок 2.11), а у зрозумілому для людини вигляді. Таким чином, кожна ознака перетворилася на ту кількість нових ознак, скільки було у ній унікальних критеріїв. Без видалення ознак з великою кількістю категорій кількість нових ознак сягала б декількох тисяч, що б сильно ускладнювало подальшу роботу.

	0	1	2	3	4	5	6	7	8	9	...	427	428	429	430	431	432	433	434	435	436
0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

5 rows × 437 columns

Рисунок 2.11 – Кодування без указування імен ознак

На даному етапі потрібно поєднати стовпці ознак та відокремити частину рядків для тестового набору (рисунок 2.12).

```
# об'єднання числових і категоріальних даних
Data = data_num.merge(ohe_data, how ='left', left_index=True, right_index=True)
Data.shape
```

```
(26824, 681)
```

```
# розділення даних на тренувальну та тестову вибірки (у співвідношенні ~ 70/30)
data_train = pd.DataFrame(Data.iloc[:18800])
data_test = pd.DataFrame(Data.iloc[18800:])
```

Рисунок 2.12 – Поєднання стовпців ознак та розділення датасету

Далі ще раз розділяємо числові та категоріальні ознаки, в останніх видаляємо дублікати стовпців.

```
# прибираємо дублікати стовпців в категоріальних даних
data_1 = pd.DataFrame(data_train[list2_cat])
ohe_data_2 = data_1.T.drop_duplicates().T
ohe_data_2.head(5)
```

	cat_1_1	cat_1_2	cat_1_3	cat_1_4	cat_1_5	cat_2_1	cat_2_2	cat_3_1	cat_3_2	cat_4_1	...	cat_134_2	cat_134_3	cat_134_4	cat_134_5	cat_134_6	c
0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	
1	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	...	1.0	0.0	0.0	0.0	0.0	
2	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	...	0.0	1.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	...	0.0	0.0	1.0	0.0	0.0	
4	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	

5 rows × 309 columns

Рисунок 2.13 – Видалення дублікатів серед категоріальних ознак

Як бачимо з рисунку 2.13, кількість ознак зменшилась з 437 до 309, що суттєво облегшить роботу моделі.

2.2.4 Нормалізація та стандартизація значень числових ознак

Нормалізуємо та стандартизуємо одночасно значення за формулою 1.3.

```
# Нормалізація тренувальних даних
data_2 = pd.DataFrame(data_train[list2_num])
df_norm = (data_2-data_2.mean())/data_2.std()
df_norm.head(5)
```

	num_1	num_2	num_4	num_6	num_7	num_8	num_10	num_13	num_14	num_15	...	num_407	num_408	num_409	num_410
0	-0.056440	-0.021408	-0.041123	-0.035649	-0.058806	0.455999	-0.036211	2.574620	0.094394	-0.648245	...	0.010932	-0.042789	0.175652	-0.047642
1	-0.088253	-0.075323	-1.053676	-0.049674	-1.046782	0.252547	-0.043079	-0.301990	-0.070005	0.347548	...	-0.062011	-0.042789	-0.120292	-0.047642
2	-0.085192	-0.075492	1.318553	-0.053461	1.299661	0.464137	-0.041751	-0.348994	-0.070670	-0.150349	...	-0.061796	-0.029200	-0.116352	-0.034069
3	-0.088455	-0.072924	-0.133048	-0.049774	-0.141137	-0.178772	-0.036763	-0.340936	-0.070849	0.347548	...	-0.061486	-0.042789	-0.120292	-0.047642
4	0.037190	-0.026172	0.009642	-0.054881	0.023526	-1.057684	0.136384	0.145214	-0.040273	-0.648245	...	-0.052825	-0.042789	-0.111246	-0.047642

5 rows × 244 columns

```
# Нормалізація тестових даних
dftest_minmax = pd.DataFrame(data_test[list2_num])
dftest_norm = (dftest_minmax-dfminmax.mean())/dfminmax.std()
```

Рисунок 2.14 – Нормалізація тренувальних та тестових числових ознак

Далі поєднаємо числові та категоріальні ознаки тестового набору даних.

```
# поєднаємо тестувальні нормалізовані числові дані та категоріальні
Data_test = dftest_norm.merge(data_test[list2_cat], how ='left', left_index=True, right_index=True)
Data_test.head(5)
```

	num_1	num_2	num_4	num_6	num_7	num_8	num_10	num_13	num_14	num_15	...	cat_134_3	cat_134_4	cat_134_5	cat_
18800	-0.087721	-0.073820	0.875389	-0.054690	0.888005	-1.016994	-0.041876	-0.182467	-0.071206	0.347548	...	0.0	0.0	1.0	
18801	-0.087081	-0.072459	-0.297691	-0.027501	-0.305800	0.936146	-0.020320	-0.335564	-0.003371	-0.150349	...	0.0	0.0	0.0	
18802	-0.074265	-0.066714	1.557285	-0.039565	1.546655	-1.016994	-0.043079	-0.353022	-0.034050	-0.150349	...	0.0	0.0	0.0	
18803	-0.088722	-0.075431	-1.083861	-0.053923	-1.087947	-1.016994	-0.039119	-0.273788	-0.064205	-0.150349	...	1.0	0.0	0.0	
18804	-0.084754	-0.072195	0.864413	-0.054714	0.846839	-1.016994	-0.038016	-0.174409	-0.070125	2.837030	...	0.0	0.0	0.0	

5 rows × 681 columns

Рисунок 2.15 – Поєднання ознак для тестового набору даних

2.2.5 Усунення мультиколінеарності даних

Усунення мультиколінеарності між числовими ознаками дозволить позбутися певної кількості ознак для полегшення роботи алгоритму класифікації. Завдяки мові Python це можна зробити за декілька кроків:

1) Будується матриця коефіцієнтів кореляції.

```
# Усунення мультиколінеарності між числовими ознаками тренувального набору даних
# Дивимся, як ознаки корелюють між собою
# Для цього побудуємо матрицю з коефіцієнтами кореляції:
CorrKoef2 = df_norm.corr()
CorrKoef2
```

	num_1	num_2	num_4	num_6	num_7	num_8	num_10	num_13	num_14	num_15	...	num_407	num_408	num_409	nu
num_1	1.000000	0.844593	0.034521	0.092518	0.034599	0.031770	0.068729	0.141249	0.125753	-0.037994	...	0.171400	-0.003597	0.014466	0.
num_2	0.844593	1.000000	0.028063	0.149234	0.028293	0.029230	0.141019	0.140266	0.239759	-0.036124	...	0.249848	-0.006644	0.003053	0.
num_4	0.034521	0.028063	1.000000	-0.002316	0.999930	0.040748	0.021850	0.003109	0.026745	0.033793	...	0.031849	-0.036817	-0.005315	-0.
num_6	0.092518	0.149234	-0.002316	1.000000	-0.002369	-0.005732	0.338473	-0.005962	0.512549	-0.008694	...	0.400430	0.260851	0.007535	0.
num_7	0.034599	0.028293	0.999930	-0.002369	1.000000	0.040855	0.022208	0.003058	0.027105	0.033998	...	0.032113	-0.036870	-0.005432	-0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
num_412	0.510060	0.541590	0.035178	0.462299	0.035535	0.014222	0.392718	0.058144	0.905230	-0.024576	...	0.652926	-0.000457	0.037664	0.
num_413	0.040087	0.079203	0.019011	0.456529	0.019325	0.000098	0.219913	0.005457	0.704872	-0.008463	...	0.288967	0.000337	0.049928	0.
num_414	0.018838	0.013866	-0.100122	-0.002304	-0.100128	0.050544	-0.003728	0.177273	0.005704	-0.022755	...	-0.001706	0.000662	0.038432	0.
num_415	-0.002170	-0.002075	-0.010679	-0.001135	-0.010658	0.002870	-0.001047	-0.003593	-0.001438	0.018166	...	-0.001532	-0.001200	-0.002667	-0.
num_416	0.101568	0.079078	0.000730	-0.004083	0.000742	-0.001465	-0.002462	0.013721	0.015859	-0.017789	...	-0.003914	0.009826	0.086966	0.

244 rows × 244 columns

Рисунок 2.16 – Матриця коефіцієнтів кореляції

2) Виділяється список ознак, у яких коефіцієнт кореляції більший за 0,85 (прийнято брати значення від 0,7 до 0,95).

```
CorField2 = []
for i in CorrKoef2:
    for j in CorrKoef2.index[CorrKoef2[i] > 0.85]:
        if i != j and j not in CorField2 and i not in CorField2:
            CorField2.append(j)
            print ("%s --> %s: r^2 = %f" % (i,j, CorrKoef2[i][CorrKoef2.index==j].values[0]))
```

```
num_53 --> num_298: r^2 = 0.856810
num_57 --> num_321: r^2 = 1.000000
num_58 --> num_119: r^2 = 0.897474
num_68 --> num_69: r^2 = 1.000000
num_68 --> num_83: r^2 = 1.000000
num_68 --> num_117: r^2 = 1.000000
num_68 --> num_175: r^2 = 0.923777
num_68 --> num_344: r^2 = 1.000000
num_68 --> num_346: r^2 = 0.995523
num_70 --> num_155: r^2 = 0.946951
num_72 --> num_339: r^2 = 1.000000
num_77 --> num_173: r^2 = 0.984299
num_80 --> num_81: r^2 = 0.999177
num_87 --> num_282: r^2 = 0.857237
num_87 --> num_384: r^2 = 0.857237
num_92 --> num_124: r^2 = 0.891057
num_92 --> num_264: r^2 = 1.000000
num_92 --> num_268: r^2 = 0.983954
num_92 --> num_269: r^2 = 0.983954
```

Рисунок 2.17 – Список ознак з коефіцієнтом кореляції більше 0,85

3) Проглянувши список, видаляються ознаки з сильною кореляцією. Якщо коефіцієнт дорівнює одиниці, і для ознаки таких випадків декілька – видаляються всі ознаки окрім останньої (наприклад, у ознаки num_68 коефіцієнт кореляції дорівнює одиниці для ознак num_69, num_83, num_117, num_344, у такому випадку можна залишити лише num_344).

```
new_data_2 = pd.DataFrame(df_norm.drop(['num_1', 'num_2', 'num_4', 'num_6', 'num_8', 'num_13', 'num_14', 'num_15', 'num_28',
            'num_29', 'num_32', 'num_41', 'num_50', 'num_53', 'num_57', 'num_58', 'num_68', 'num_69', 'num_83', 'num_117',
            'num_70', 'num_72', 'num_77', 'num_80', 'num_87', 'num_92', 'num_264', 'num_270', 'num_380', 'num_94', 'num_95',
            'num_104', 'num_112', 'num_129', 'num_132', 'num_147', 'num_161', 'num_164', 'num_176', 'num_180', 'num_185',
            'num_220', 'num_244', 'num_246', 'num_338', 'num_406', 'num_296', 'num_319', 'num_355', 'num_408'], axis=1))
len(new_data_2.columns)
```

```
194
```

Рисунок 2.18 – Видалення ознак із сильною кореляцією

Таким чином, отримали новий датасет, для якого потрібно повторювати вищеописані кроки до тих пір, доки ознаки мають велику кореляцію.

Коли кореляції між ознаками більше немає, об’єднуємо числові та категоріальні дані в кінцевий датасет, готовий до навчання моделі.

```
# кореляції більше немає - об'єднуємо числові та категоріальні дані
# отримуємо кінцевий датасет, готовий для навчання моделі
data_fin = new7_data_2.merge(ohe_data_2, how ='left', left_index=True, right_index=True)
data_fin.head(5)
```

	num_10	num_26	num_27	num_30	num_38	num_40	num_43	num_52	num_55	num_59	...	cat_134_2	cat_134_3	cat_134_4	cat_134_
0	-0.036211	0.485763	-0.023972	0.107447	0.000912	1.729937	0.672018	-0.162585	-0.624201	-0.424417	...	0.0	0.0	0.0	0.
1	-0.043079	-0.813632	0.023486	-0.295476	0.015733	-0.402294	-0.358893	-0.300900	-0.712248	0.071042	...	1.0	0.0	0.0	0.
2	-0.041751	-0.732420	0.027624	-0.295476	0.021667	1.168823	-0.316658	0.031056	1.048696	-0.754723	...	0.0	1.0	0.0	0.
3	-0.036763	-0.028581	0.024641	-0.230363	0.017990	1.056601	-0.362844	1.331219	-0.712248	1.061960	...	0.0	0.0	1.0	0.
4	0.136384	0.594046	0.035131	0.397353	0.022084	-1.300076	-0.191864	-0.273237	1.709050	2.768542	...	0.0	0.0	0.0	0.

5 rows × 445 columns

Рисунок 2.19 – Кінцевий датасет, готовий до навчання моделі

Завершальним етапом цього пункту є відбір ознак з тестового датасету, які є лише в кінцевому тренувальному датасеті, та виділення двох стовпців зі значеннями цільової функції (для тренувального набору даних та для тестового, щоб порівняти з прогнозними значеннями та розрахувати похибку).

```
# Завершимо підготовку кінцевого датасету тестових даних
```

```
# відбираємо із тестових даних лише ті стовпці, які є в тренувальних даних
test_col = data_fin.columns.tolist()
test_fin = pd.DataFrame(Data_test[test_col])
test_fin.head(5)
```

	num_10	num_26	num_27	num_30	num_38	num_40	num_43	num_52	num_55	num_59	...	cat_134_2	cat_134_3	cat_134_4	cat_
18800	-0.041876	-0.597067	0.024661	-0.172048	0.021809	-1.300076	-0.368366	0.943936	-0.712248	-0.039060	...	0.0	0.0	0.0	
18801	-0.020320	0.431621	-0.023331	-0.275225	-0.009537	1.505491	0.493116	-0.300900	0.080177	-0.754723	...	1.0	0.0	0.0	
18802	-0.043079	-0.434642	0.016911	-0.233552	0.020695	0.158819	0.302561	-0.287069	-0.712248	-0.424417	...	0.0	0.0	0.0	
18803	-0.039119	-0.434642	0.019386	-0.208337	0.021809	0.046596	-0.314240	0.141708	-0.712248	-0.534519	...	0.0	1.0	0.0	
18804	-0.038016	-0.299289	0.023157	-0.121395	0.021809	1.168823	-0.356307	-0.300900	-0.580177	0.841756	...	0.0	0.0	0.0	

5 rows × 445 columns

```
# виділяємо стовпець цільової функції для тренування моделі
targ_train = pd.DataFrame(data['gb'].iloc[:18800])
# виділяємо стовпець цільової функції тестових даних для порівняння
# з прогнозними значеннями і визначенням похибки
targ_test = pd.DataFrame(data['gb'].iloc[18800:])
```

Рисунок 2.20 – Створення кінцевого тестового датасету та виділення цільових функцій

2.3 Порівняння XGBoost з іншими моделями з параметрами за умовчуванням

Для порівняння моделі XGBoost були обрані наступні моделі класифікації:

- LogisticRegression — це варіант реалізації логістичної регресії;
- DecisionTreeClassifier — реалізація алгоритму дерева ухвалення рішення;
- ExtraTreesClassifier — це посилений варіант алгоритму випадкових лісів;
- GradientBoostingClassifier — це варіант алгоритму бустингу (Boosting).

Не зважаючи на налаштування параметрів, поглянемо, як будуть працювати ці класифікатори. Засоби Python дозволяють реалізувати всі п’ять класифікаторів одночасно.

Перш ніж робити будь-які модифікації або налаштування алгоритму XGBoost, важливо протестувати модель XGBoost за замовчуванням і встановити базовий рівень продуктивності.

Хоча бібліотека XGBoost має власний API Python, можна використовувати моделі XGBoost з API scikit-learn через клас обгортки XGBClassifier.

```
# Побудуємо декілька моделей з параметрами за замовчуванням
models = []
models.append(LogisticRegression())
models.append(DecisionTreeClassifier())
models.append(ExtraTreesClassifier())
models.append(GradientBoostingClassifier())
models.append(xgb.XGBClassifier())
```

```
for model in models:
    model.fit(data_fin, targ_train.values.ravel())

# розрахуємо точність моделі
for model in models:
    print('Назва моделі:', str(model)[:str(model).find('(')])

    y_pred = model.predict(test_fin)

    print('Побудуємо confusion matrix: [TP, FP] x [FN, TN]')
    print(confusion_matrix(targ_test, y_pred))
    print('Log loss: ',log_loss(y_true = targ_test, y_pred = y_pred))
    print(precision_recall_fscore_support(targ_test, y_pred, average=None))
    print('---------------------------------------')
```

Рисунок 2.21 – Побудова п’яти моделей класифікації

```
Назва моделі: LogisticRegression
Побудуємо confusion matrix: [TN, FP] x [FN, TP]
[[7788   63]
 [ 132   41]]
Log loss:  0.8393708588790743
(array([0.98333333, 0.39423077]), array([0.99197554, 0.23699422]), array
([0.98763553, 0.29602888]), array([7851,  173], dtype=int64))
----------------------------------------
Назва моделі: DecisionTreeClassifier
Побудуємо confusion matrix: [TN, FP] x [FN, TP]
[[7693  158]
 [  98   75]]
Log loss:  1.1019507843333887
(array([0.98742138, 0.32188841]), array([0.97987518, 0.43352601]), array
([0.98363381, 0.36945813]), array([7851,  173], dtype=int64))
----------------------------------------
Назва моделі: ExtraTreesClassifier
Побудуємо confusion matrix: [TN, FP] x [FN, TP]
[[7846    5]
 [ 122   51]]
Log loss:  0.5466635842648074
(array([0.98468876, 0.91071429]), array([0.99936314, 0.29479769]), array
([0.99197168, 0.44541485]), array([7851,  173], dtype=int64))
----------------------------------------
Назва моделі: GradientBoostingClassifier
Побудуємо confusion matrix: [TN, FP] x [FN, TP]
[[7803   48]
 [ 124   49]]
Log loss:  0.7403673879114273
(array([0.98435726, 0.50515464]), array([0.99388613, 0.28323699]), array
([0.98909875, 0.36296296]), array([7851,  173], dtype=int64))
----------------------------------------
Назва моделі: XGBClassifier
Побудуємо confusion matrix: [TN, FP] x [FN, TP]
[[7841   10]
 [ 138   35]]
Log loss:  0.6370571912289493
(array([0.9827046 , 0.77777778]), array([0.99872628, 0.20231214]), array
([0.99065066, 0.32110092]), array([7851,  173], dtype=int64))
----------------------------------------
```

Рисунок 2.22 – Результати побудови п’яти моделей класифікації

З рисунку 2.22 можна зробити висновки, що серед усіх п’яти класифікаторів модель XGBoost виявилася не найгіршою, хоча вона і дає найменшу кількість TP значень (значень, які дорівнюють одиниці – схваленню видачі кредиту, та при прогнозуванні справді виявилися одиницею) – 35 схвалень.

Найбільшу кількість TP значень (75 схвалень) показує модель DecisionTreeClassifier, але вона також показує і найбільше значення FP (значень, що дорівнюють нулю – відмові у видачі кредиту, та при прогнозуванні виявилися одиницею) – 158 схвалень, що автоматично робить її найгіршою з усіх.

LogisticRegression показала 41 значення TP та 63 значення FP, тобто тих, хто не повинен був отримати кредит, але отримав, виявилося більше, ніж тих, хто повинен був отримати кредит і все ж отримав його. Це не є добрим результатом також.

Результатом роботи GradientBoostingClassifier є видача правильних кредитів 49 позичальникам, а також видача 48 кредитів тим, хто не повинен був його отримати. Цей результат є трохи кращим, ніж показала модель LogisticRegression.

ExtraTreesClassifier показав найкращі результати, а саме 51 правильну видачу кредитів та 5 видач кредитів тим, хто не повинен був його отримати. Логістична функція втрат для цього класифікатора є найменшою.

Подібні результати можна пояснити тим, що набір даних є незбалансованим, тому відбувається таке викривлення в сторону значення FP.

2.4 Відбір найбільш важливих та інформативних ознак у моделі XGBoost

Для зменшення розмірності датасету доречно здійснити відбір інформативних ознак. Для цього можна використати вбудовану у модель функцію – feature_importances_. Вона показує ознаки, які сама модель у ході побудови визнала важливими. Для зручності можна побудувати графік важливості ознак.

```
# відбір ознак за feature_importances_
# побудуємо графік важливостей ознак
pyplot.figure(figsize=(12,7))
pyplot.bar(range(len(xgb_model.feature_importances_)), xgb_model.feature_importances_)
pyplot.show()
```

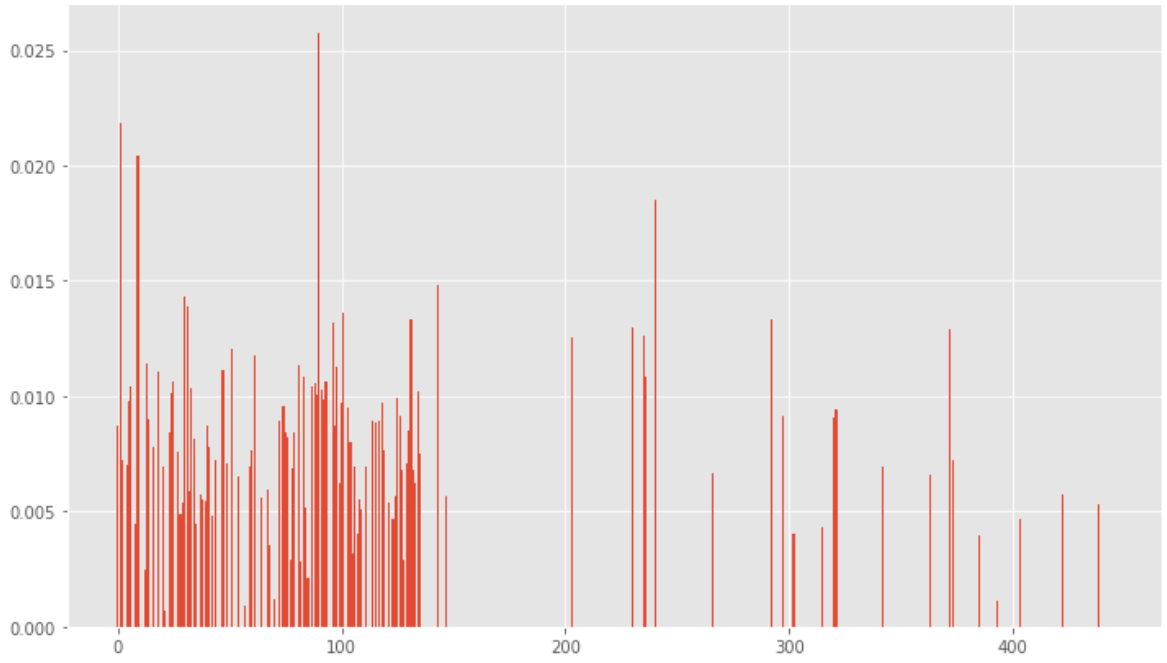

Рисунок 2.23 – Графік важності ознак

Як можна підмітити з рисунку 2.23, значна кількість ознак не була визнана важливими моделлю, вони лише сповільнюють роботу алгоритму. Тому їх відсутність не повинна вплинути на результати роботи.

Відібравши всі ознаки, чия важливість > 0, була натренована нова модель, результати якої зображені на рисунку 2.24.

```
Кількість відібраних ознак: 121
Confusion matrix: [TN, FP] x [FN, TP]
[[7841   10]
 [ 138   35]]
Log loss:  0.6370571912289493
(array([0.9827046 , 0.77777778]), array([0.99872628, 0.20231214]), array
([0.99065066, 0.32110092]), array([7851,  173], dtype=int64))
```

Рисунок 2.24 – Результати роботи моделі з відібраними 121 ознаками

Результат роботи не змінився, отже класифікатор може впевнено працювати з великою кількістю неінформативних ознак, просто не беручи їх до уваги.

Спробувавши відібрати різну кількість ознак, вказавши порогове значення їх важливості, та навчивши моделі на нових датасетах, отримали наступні результати:

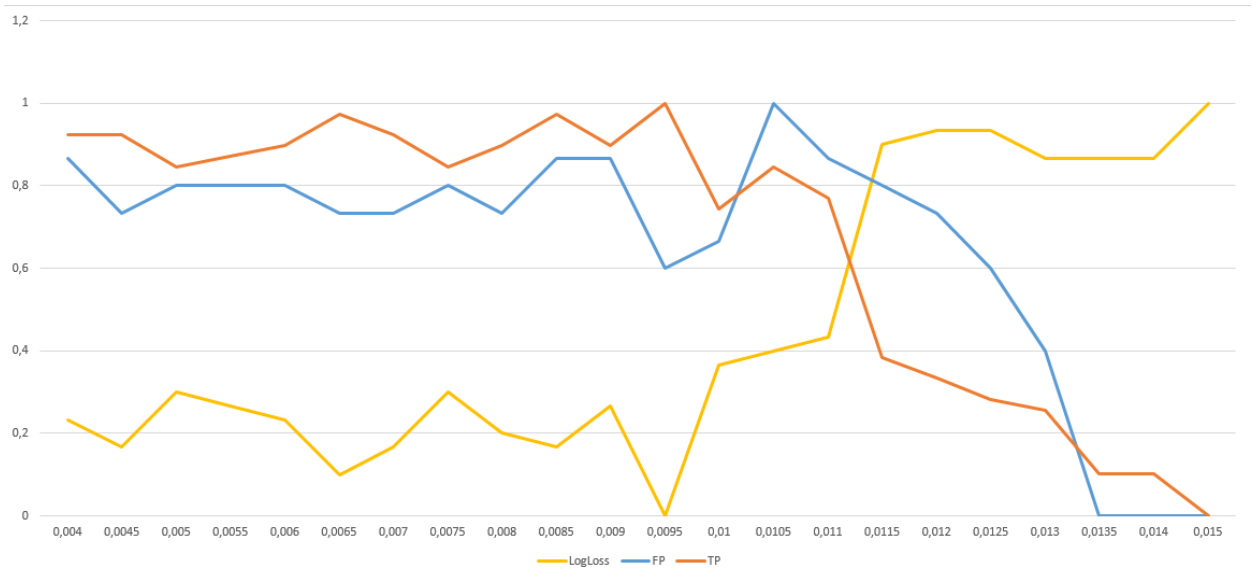

Рисунок 2.25 – Графік результатів роботи моделі з різним пороговим значенням важливості відбору ознак

Отримані дані були нормалізовані, що дало можливість їх об’єднати на одному графіку. Кількість FP повинна бути низькою, тоді як значення TP – високою. LogLoss, у свою чергу, повинен бути мінімальним.

На графіку видно, що підвищення порогового значення більш ніж 0,0095 веде до суттєвого погіршення прогнозу, а саме цей поріг дає найкращі результати. Поріг в 0,0095 перетинають лише 40 ознак, але підбір параметрів лише на 40 ознаках виключає можливі покращення при більшому наборі ознак, тому доречніше робити підбір при установленні меншого порогу відсікання ознак, наприклад, при 0,0045 (FP = 11, TP = 37, LogLoss = 0,632753).

2.5 Огляд параметрів моделі та аналіз результатів їх перебору

Алгоритм XGBoost ефективний для широкого кола проблем прогнозуючого моделювання регресії та класифікації, який пропонує ряд параметрів, багато з яких вимагають налаштування, щоб отримати максимальну віддачу від алгоритму на даному наборі даних.

Хоча XGBoost добре працює в цілому, навіть на незбалансованих наборах даних класифікації, він пропонує спосіб налаштувати навчальний алгоритм, щоб приділити більше уваги неправильній класифікації класу меншості для наборів даних з похилим розподілом класів. Для цього потрібно звернутися до параметру scale_pos_weight.

За замовчуванням параметр scale_pos_weight встановлюється на значення 1,0 і має ефект зважування залишку позитивних прикладів щодо негативних прикладів під час підсилення дерев рішень. Його точний розрахунок можна побачити на рисунку 2.26.

```
# для отримання значення параметру scale_pos_weight
# розрахуємо кількість відмов і згод для видачі кредиту
counter = Counter(targ_train['gb'])
# розрахуємо scale_pos_weight
estimate = counter[0] / counter[1]
print('Estimate: %.3f' % estimate)
```

```
Estimate: 43.762
```

Рисунок 2.26 – Розрахунок значення для параметру scale_pos_weight

За швидшу продуктивність навчання відповідає параметр tree_method. Його можна встановити на значення «hist» або «gpu_hist» для більшої швидкості обчислення.

Коли спостерігається висока точність тренувань, але низька точність тесту, ймовірно, це є проблемою перенавчання (overfitting). Є два способи контролю перенавчання в XGBoost:

- Перший спосіб – це безпосередній контроль складності моделі. Для вирішення проблеми можна налаштувати параметри max_depth, min_child_weight та gamma.
- Другий спосіб – додати випадковості, щоб зробити навчання стійким до шуму. Це може буто зроблено за допомогою налаштування параметрів subsample та colsample_bytree. Також можна зменшити розмір «eta», та при цьому потрібно збільшувати num_round.

Перебір параметрів власноруч часто затрачає багато часу, тому для його полегшення можна використовувати спеціальні функції, наприклад GridSearch з бібліотеки scikit-learn.

Використання даної функції прискорює пошук необхідних параметрів, але і потребує значної обчислювальної потужності комп'ютера для великих наборів даних. Так як після встановлення порогового значення відбору інформативних ознак до 0,0045 їх кількість була скорочена до 104 при 18800 строках тренувальних даних, послідовний перебір параметрів за допомогою GridSearch займає занадто багато часу, тому в цій роботі я обмежилась перебором власноруч.

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.5, gamma=0,
              learning_rate=0.2, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=500, n_jobs=
1,
              nthread=None, objective='binary:logistic', random_state=35,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=44, seed=None,
              silent=None, subsample=1, verbosity=1)
```

Рисунок 2.27 – Відібрані параметри в результаті ручного перебору

На тестових даних модель з такими параметрами показала наступний результат:

```
Кількість відібраних ознак: 104
Confusion matrix: [TN, FP] x [FN, TP]
[[7831   20]
 [  43  130]]
Log loss:  0.2711813191460598
(array([0.99453899, 0.86666667]), array([0.99745255, 0.75144509]), array
([0.99599364, 0.80495356]), array([7851,  173], dtype=int64))
```

Рисунок 2.28 – Результати прогнозу моделі з підібраними параметрами

Значення FP збільшилося з 11 до 20, що є незначним погіршенням, значення TP зросло з 37 до 130, що є суттєвим покращенням прогнозу моделі, логістична функція втрат LogLoss значно зменшилась з 0,632753 до 0,271181.

Для більшої впевненості побудуємо моделі для іншої кількості ознак, поступово підвищуючи поріг відбору.

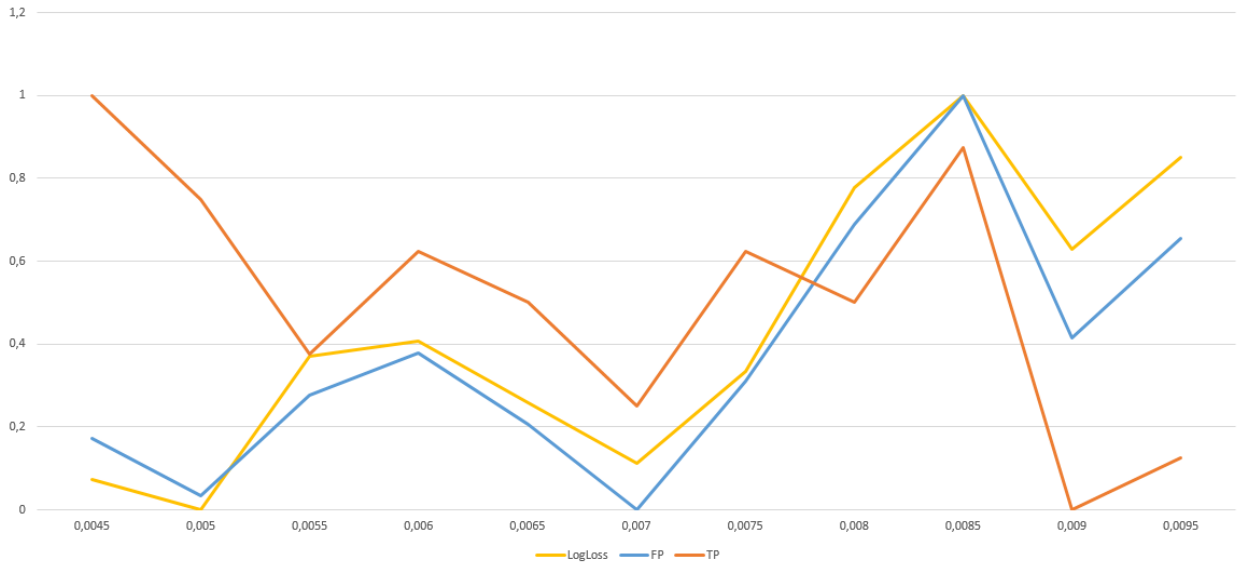

Рисунок 2.29 – Результати роботи моделі з налаштованими параметрами з різним пороговим значенням важливості відбору ознак

Як можна побачити з рисунку 2.29, логістична функція втрат LogLoss досягає своїх найменших значень при порогових значеннях, що дорівнюють 0,005 та 0,007, а також мають найнижче значення FP, але з огляду на те, що значення TP при порозі 0,005 набагато вище, можна зробити висновок, що модель з параметрами, продемонстрованими на рисунку 2.28, при пороговому значенні відсікання ознак, що дорівнює 0,005, дає свої найкращі результати.

```
Кількість відібраних ознак: 100
Confusion matrix: [TN, FP] x [FN, TP]
[[7835   16]
 [  45  128]]
Log loss:  0.2625720530469145
(array([0.99428934, 0.88888889]), array([0.99796204, 0.73988439]), array
([0.99612231, 0.80757098]), array([7851,  173], dtype=int64))
```

Рисунок 2.30 – Результати оцінки прогнозних значень за найкращою моделлю

2.6 Результати розв’язку задачі кредитного скорингу

Для отримання розв’язку задачі кредитного скорингу необхідно провести роботу з підготовки даних, а саме: перетворення NaN значень, нормалізації числових ознак, розділення на бінарні стовпці категоріальних даних, відокремлення числа ознак, що були підібрані при встановленні порогового значення відсікання ознак як 0,005.

Після цього був створений прогноз для нових даних, що не мають цільової функції. Відокремивши індивідуальні номери позичальників та поєднавши їх із стовпцем прогнозних значень, отримали шуканий результат розв’язку задачі. Для зручності були відокремлені прогнози всіх позичальників, кому було схвалено надання кредиту.

	id	pred
144	5388	1
270	5514	1
563	5807	1
607	5851	1
684	5928	1
...	...	...
9048	6435	1
9436	10675	1
9593	8559	1
9722	8663	1
9850	6537	1

107 rows × 2 columns

Рисунок 2.31 – Результат розв’язку задачі (позитивні передбачення)

З рисунку 2.31 видно, що кількість схвалених кредитів дорівнює 107 з 9922 можливих, тобто лише трохи більше одного відсотку (~1,07841) позичальників отримало кредит.

При більш ретельному переборі параметрів результати прогнозу можливо можуть дати кращий результат і підвищити кількість виданих кредитів, та в такому випадку треба орієнтуватися на пріоритети банку – цілком можливо, що помилкове схвалення кредиту може нанести набагато більший збиток, ніж недоотриманий прибуток від помилкової відмови.

Висновки

У даному розділі був проведений аналіз вихідних даних, в ході якого установлено, що дані включають в себе як числові, так і категоріальні дані, що потребує окремої підготовки даних.

Була проведена ретельна підготовка даних, а саме видалення та перетворення NaN даних, видалення ознак з єдиним унікальним значенням, перетворення категоріальних даних в доступний для розуміння алгоритмом вид за

допомогою кодувальника One-Hot Encoding, нормалізація та стандартизація значень числових ознак, усунення мультиколінеарності даних.

Було проведене навчання декількох додаткових моделей, окрім XGBoost, класифікації (DecisionTreeClassifier, LogisticRegression, ExtraTreesClassifier, GradientBoostingClassifier) з параметрами за умовчуванням та обчислення похибки їх прогнозу на тестових даних. Аналіз отриманих результатів показав, що серед усіх п’яти класифікаторів модель XGBoost виявилася не найгіршою, хоча вона і дає найменшу кількість значень схвалення видачі кредиту. Найкращі результати були отримані за допомогою класифікатора ExtraTreesClassifier. Якщо приділити більше уваги підбору параметрів для цього класифікатора, він може також давати задовільні результати прогнозу. Подібні результати пояснюються тим, що набір даних є незбалансованим.

Був проведений відбір найбільш важливих та інформативних ознак у моделі бустингу для зменшення складності моделі та покращення результатів прогнозу, також був проведений огляд та перебір параметрів моделі для виявлення більш придатних для даного набору даних.

На моделі зі зменшеною кількістю ознак та з найбільш відповідними параметрами був побудований розв’язок задачі кредитного скорингу, який дав у результаті схвалення видачі кредитів 107 позичальникам.

ВИСНОВКИ

Здатність передбачати майбутнє завжди цінувалася дорого. Зараз прогнозування поставлено на твердий математичний ґрунт. І банки користуються цим, впроваджуючи системи автоматизованого скорингу, призначені для оцінки кредитних ризиків фізичних осіб.

Основною метою традиційного скорингу є класифікація клієнтів банку на "хороших" і "поганих", виходячи з якої кредитор може вибирати відповідні дії по відношенню до даного клієнта, тобто приймати рішення про те, видавати чи не видавати кредит. Таким чином, точність побудованої моделі класифікації для вирішення задачі кредитного скорингу має прямий вплив на економіку банку, що її використовує.

У ході виконання даної дипломної роботи вирішувалося питання розв’язання задачі кредитного скорингу за допомогою алгоритму машинного навчання, заснованого на дереві пошуку рішень, котрий використовує фреймворк градієнтного бустинга (XGBoost).

У першому розділі були наведені теоретичні дані щодо сутності банку, означення кредиту, його форм і видів, розглянуто основні поняття задачі кредитного скорингу, її математичну постановку та основні алгоритми для вирішення цієї задачі. Було обґрунтовано важливість попередньої обробки даних, яка включає в себе вибірку даних, їх очищення, генерацію ознак та їх форматування. Розглянуто основні методи вибору оптимального набору інформативних ознак, обґрунтовано вибір мови програмування Python та його можливості у сфері аналізу даних. Розглянуто моделі для вирішення задачі класифікації, приділено окрему увагу алгоритму машинного навчання XGBoost, оглянуто основні метрики для оцінки точності методу класифікації і зроблено висновок, що вибір метрики потрібно робити з фокусом на предметну область, попередньо обробляючи дані і в разі нерівності класів потрібно підбирати баланс класів для навчання.

У другому розділі проведений аналіз вихідних даних, підготовка даних для побудови моделі, навчання декількох додаткових, окрім XGBoost, моделей класифікації (DecisionTreeClassifier, LogisticRegression, ExtraTreesClassifier, GradientBoostingClassifier) з параметрами за умовчуванням та проведений аналіз отриманих результатів. Проведений відбір найбільш важливих ознак, огляд та перебір параметрів моделі для виявлення більш придатних для даного набору даних, в ході якого стало зрозуміло, що оптимальні параметри моделі можуть залежати від багатьох сценаріїв, тому неможливо створити всебічний алгоритм для цього.

У результаті проведеної роботи був зроблений висновок про доцільність використання алгоритму XGBoost для розв’язання задачі кредитного скорингу.

Потрібно зауважити, що в області машинного навчання постійно ведуться нові дослідження. На даний момент вже існує кілька життєздатних альтернатив XGBoost. Нещодавно Microsoft Research випустила фреймворк LightGBM для градієнтного бустинга. Яндекс випустила CatBoost, що показує вражаючу продуктивність для даних, що не потребують бінарізації категоріальних даних. Створення фреймворка, що перевершує XGBoost в рамках продуктивності, гнучкості і практичності – всього лише питання часу. Але поки такого немає, XGBoost продовжить лідирувати у світі машинного навчання.

Результати дипломної роботи можуть бути застосовані для оптимізації діяльності роботи алгоритмів класифікації не тільки для розв’язання банківської задачі кредитного скорингу, а й в інших сферах, що потребують надійного інструменту передбачення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Банки, їх роль та функції. Банківська система в освіті та подоланню економічної кризи у країні: веб-сайт. URL: https://studfile.net/preview/2278651/ (дата звернення: 9.11.2020).

2. Роль банків у сучасній економіці: веб-сайт. URL: https://works.doklad.ru/view/wWsPmqDZlZs.html (дата звернення: 9.11.2020).

3. Роль кредитних відносин у суспільному відтворенні: веб-сайт. URL: https://osvita.ua/vnz/reports/bank/19816/ (дата звернення: 12.11.2020).

4. Бюлетень Національного банку України: веб-сайт. URL: https://bank.gov.ua/files/stat.pdf (дата звернення: 9.11.2020).

5. Кредитний менеджмент: веб-сайт. URL: https://banking.uabs.sumdu.edu.ua/images/department/banking/discip/kreditmanag/Lekcii_krmen.pdf (дата звернення: 12.11.2020).

6. Суть, основні риси та принципи банківського кредитування: веб-сайт. URL: https://ru.osvita.ua/vnz/reports/bank/20468/ (дата звернення: 12.11.2020).

7. Про ефективність статистичних алгоритмів кредитного скорингу // Банкаўскі веснік. 2010.

8. Впровадження системи кредитного скорингу в банку. Розрахунки і операційна робота в комерційному банку // Методический журнал. 2004.

9. CRISP-DM: перевірена методологія для Data Scientist-ів: веб-сайт. URL: https://habr.com/ru/company/lanit/blog/328858/ (дата звернення: 13.11.2020).

10. Як підготувати дані для моделювання: 5 операцій Data Preparation: веб-сайт. URL: https://www.bigdataschool.ru/blog/data-preparation-operations.html (дата звернення: 19.11.2020).

11. Навіщо потрібна очистка даних для Data Mining: 10 головних проблем підготовки датасету і способи їх вирішення: веб-сайт. URL: https://www.bigdataschool.ru/blog/data-preparation-operations.html (дата звернення: 19.11.2020).

12. Очищення даних: проблеми та актуальні підходи // BPM World. 2000.

13. Дьяконов А.Г. Методи вирішення задач класифікації з категоріальними ознаками // Прикладна математика та інформатика.2014.

14. Відкритий курс машинного навчання. Навчання на гигабайтах з Vowpal Wabbit: веб-сайт. URL: https://habr.com/ru/company/ods/blog/326418/#rabota-s-kategorialnymi-priznakami-label-encoding-one-hot-encoding-hashing-trick (дата звернення: 21.11.2020).

15. Introducing One of the Best Hacks in Machine Learning: the Hashing Trick: веб-сайт. URL: https://medium.com/value-stream-design/introducing-one-of-the-best-hacks-in-machine-learning-the-hashing-trick-bf6a9c8af18f (дата звернення: 21.11.2020).

16. Нормалізація вхідних векторів (Normalization): веб-сайт. URL: https://wiki.loginom.ru/articles/normalization.html (дата звернення: 22.11.2020).

17. Data Preparation: політ нормальний - що таке нормалізація даних і навіщо вона потрібна: веб-сайт. URL: https://www.bigdataschool.ru/blog/нормализация-feature-transformation-data-preparation.html (дата звернення: 22.11.2020).

18. Дубнов Ю. А. Про ентропійні критерії відбору ознак в задачах аналізу даних // Інформаційні технології та обчислювальні системи. 2018.

19. Методи вибору оптимального набору інформативних ознак для завдань класифікації текстів: веб-сайт. URL: http://seminar.at.ispras.ru/seminar/wp-content/uploads/2012/07/Coursework.pdf (дата звернення: 26.11.2020).

20. R. Kohavi. Wrappers for feature selection // Artificial Intelligence. 1997.

21. Molina L.C., Belanche L., Nebot A. Feature Selection Algorithms: A Survey And Experimental Evaluation // IEEE Computer Society. 2002.

22. Applying Wrapper Methods in Python for Feature Selection: веб-сайт. URL: https://stackabuse.com/applying-wrapper-methods-in-python-for-feature-selection/ (дата звернення: 27.11.2020).

23. Feature Selection For Machine Learning: веб-сайт. URL: https://machinelearningmastery.com/feature-selection-machine-learning-python/ (дата звернення: 27.11.2020).

24. Огляд методів відбору ознак: веб-сайт. URL: https://habr.com/ru/company/jetinfosystems/blog/470622/ (дата звернення: 27.11.2020).

25. Критерій приросту інформації (Information Gain): веб-сайт. URL: https://wiki.loginom.ru/articles/info-gain.html (дата звернення: 27.11.2020).

26. Яку мову програмування вибрати для роботи з даними?: веб-сайт. URL: https://habr.com/ru/post/337330/ (дата звернення: 30.11.2020).

27. 5 ключових бібліотек і пакетів для аналізу даних на Python: веб-сайт. URL: https://dev.by/news/5-glavnyh-bibliotek-i-paketov-dlya-analiza-dannyh-na-python (дата звернення: 30.11.2020).

28. Огляд методів класифікації в машинному навчанні за допомогою Scikit-Learn: веб-сайт. URL: https://tproger.ru/translations/scikit-learn-in-python/ (дата звернення: 30.11.2020).

29. Jonathan Hirko. Intro to Classification and Feature Selection with XGBoost // AI Time Journal. 2019.

30. How to Configure XGBoost for Imbalanced Classification: веб-сайт. URL: https://machinelearningmastery.com/xgboost-for-imbalanced-classification/ (дата звернення: 1.12.2020).

31. Алгоритм XGBoost: нехай він царює довго! : веб-сайт. URL: https://medium.com/nuances-of-programming/алгоритм-xgboost-пусть-он-царствует-долго-dc8c4eca3fbc (дата звернення: 1.12.2020).

32. How does XGBoost Work: веб-сайт. URL: https://towardsdatascience.com/how-does-xgboost-work-748bc75c58aa (дата звернення: 1.12.2020).

33. How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification: веб-сайт. URL: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/ (дата звернення: 1.12.2020).

34. Метрики в задачах машинного навчання: веб-сайт. URL: https://habr.com/ru/company/ods/blog/328372/ (дата звернення: 1.12.2020).

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи магістра

№ з/п	Позначення	Назва	Кількість	Примітки
1				
2		Документація		
3				
4	САіУ.РД. 23.13.ПЗ	Пояснювальна записка	96	Формат А4
5				
6	САіУ.РД. 23.13.ДМ	Демонстраційні матеріали		Презентація на CD-R
7				
8	САіУ.РД. 23.13.КР	Копія роботи	1	Диск CD-R
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

					САіУ.РД.23.13.ДА.ПЗ.
Змін.	Аркуш	№ докум.	Підпис	Дата	

Розроб.	Подвисоцький А.С.			Матеріали дипломної роботи	Літ.	Аркуш	Аркушів
Керівн.	Алексєєв О.М.						
Керівн. Сп. Р.	Алексєєв О.М.				НТУ ДП 124; 124м-22-1		
Н.контр.	Хом’як Т.В.						
Зав. каф.	Желдак Т.А.						

ДОДАТОК Б

Відгук
на кваліфікаційну роботу магістра
Студента групи *124м-22-1 Подвисоцького Артема Сергійовича*
Спеціальності 124 Системний аналіз

Тема кваліфікаційної роботи магістра: «Застосування інтелектуального аналізу даних для підвищення точності класифікації споживачів кредитних запозичень»

Обсяг кваліфікаційної роботи магістра: 97 с., 40 рис., 1 табл., 3 додатки, 39 джерел.

Мета кваліфікаційної роботи магістра: дослідження якості класифікації позичальників в задачі кредитного скорингу за допомогою моделі алгоритму машинного навчання XGBoost.

Актуальність теми обумовлена тим, що на сьогоднішній день існує потреба у покращенні точності прогнозу кредитоспроможності позичальників для зниження ризиків при видачі кредиту. Тому було вирішено провести дослідження роботи алгоритму машинного навчання XGBoost, який встиг себе проявити у системі організації конкурсів з Data Mining під назвою Kaggle, виявити позитивні на негативні сторони його використання та зробити висновки щодо його ефективності до та після аналізу і обробки даних.

Тема кваліфікаційної роботи магістра безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 124 Системний аналіз і управління, оскільки включає всі необхідні етапи системного дослідження, такі як аналіз об'єкта дослідження, побудову відповідних моделей і їх застосування до практичних даних.

Виконані в кваліфікаційній роботі магістра завдання **відповідають** вимогам до професійної діяльності фахівця освітньо-кваліфікаційного рівня магістра.

Практичне значення результатів кваліфікаційної роботи магістра полягає в тому, що реалізований алгоритм можна успішно використовувати у банківській справі для покращення роботи сучасних систем скорингу, що, в свою чергу, сприятиме більш ефективному прогнозуванню кредитоспроможності позичальників та зниженню ризиків для банківської установи при видачі кредиту.

Оформлення пояснювальної записки та демонстраційного матеріалу до неї виконано згідно з вимогами. Роботу виконано самостійно, відповідно до завдання та у повному обсязі.

Кваліфікаційна робота магістра в цілому заслуговує на оцінку: добре, а її автор Подвисоцький А.С. заслуговує на присвоєння кваліфікації «магістр з системного аналізу».

Керівник кваліфікаційної роботи
магістра, к.т.н., доцент Алексєєв О.М. _____

ДОДАТОК В

Рецензія
на кваліфікаційну роботу магістра
Студента групи *124м-22-1 Подвисоцького Артема Сергійовича*
Спеціальності 124 Системний аналіз

Тема кваліфікаційної роботи: «Застосування інтелектуального аналізу даних для підвищення точності класифікації споживачів кредитних запозичень»

Обсяг кваліфікаційної роботи: 97 с., 40 рис., 1 табл., 3 додатки, 39 джерел.

Висновок про відповідність роботи завданню та освітньо-професійній програмі спеціальності – тема та зміст кваліфікаційної роботи відповідає вимогам до професійної діяльності фахівця освітньо-кваліфікаційного рівня магістра спеціальності 124 Системний аналіз і управління.

Загальна характеристика кваліфікаційної роботи, ступінь використання нормативно-методичної літератури та передового досвіду. Кваліфікаційна робота містить два розділи. В інформаційно-аналітичному розділі розглядаються поняття кредитного скорингу, основні алгоритми для вирішення задачі кредитного скорингу, розглянуто послідовність препроцесінгу даних та методів вибору оптимального набору інформативних ознак, описано переваги мови Python для аналізу даних. У спеціальному розділі виконано аналіз та підготовку вихідних даних, створено модель алгоритму XGBoost та відібрано найбільш інформативні ознаки, підібрано параметри, що максимально покращують результати прогнозу та представлено результати розв'язку задачі кредитного скорингу.

Позитивні сторони кваліфікаційної роботи: у роботі надано чітку та детальну послідовність підготовки даних, процесу побудови моделі для розв'язку задачі кредитного скорингу; також були використані інші алгоритми класифікації та проведений аналіз їх результатів у порівнянні з алгоритмом XGBoost.

Основні недоліки кваліфікаційної роботи:

1) не показано процес підбору параметрів для алгоритму XGBoost;
2) не було виконано підбір параметрів для інших використаних алгоритмів класифікації, тому немає можливості порівняти найкращі результати класифікаторів для виявлення більш придатного для вирішення поставленої задачі;

Незважаючи на вищевказані недоліки, кваліфікаційна робота магістра в цілому заслуговує оцінки: *добре*, а її автор Подвисоцький А.С. заслуговує присвоєння кваліфікації «магістр з системного аналізу».

Рецензент,

._____