МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально-науковий інститут інформаційних технологій та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістра
на тему: **«УПРАВЛІННЯ ПРОЄКТОМ ВПРОВАДЖЕННЯ МОДЕЛІ МАШИННОГО НАВЧАННЯ З ПЕРЕДБАЧЕННЯ РИЗИКІВ КРЕДИТУВАННЯ»**

Виконала: студентка 2 курсу, групи МУП-21
другого (магістерського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Управління проєктами»
Денисюк Юлія Олександрівна

Керівник: *старший викладач Клебан Юрій Вікторович*

Рецензент: *кандидат технічних наук, доцент,*
доцент кафедри прикладної математики
Донецького національного університету
імені Василя Стуса
Загоруйко Любов Василівна

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри інформаційних технологій та аналітики даних
_____ (проф., д.е.н. Кривицька О.Р.)
Протокол № 5 від 04 грудня 2025

Острог - 2025 р.

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня магістра

Тема: УПРАВЛІННЯ ПРОЄКТОМ ВПРОВАДЖЕННЯ МОДЕЛІ МАШИННОГО НАВЧАННЯ З ПЕРЕДБАЧЕННЯ РИЗИКІВ КРЕДИТУВАННЯ

Автор: Денисюк Юлія Олександрівна

Науковий керівник:
Клебан Юрій Вікторович

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: *150 (кількість сторінок роботи) с., 40 (кількість рисунків) рис., 7 (кількість таблиць) табл., 5 (кількість додатків) додатків, 60 (кількість джерел) джерел.*

Ключові слова: кредитні ризики, машинне навчання, Support Vector Machine, PCA, SMOTE oversampling, управління проєктами, канбан-методологія, банківська діяльність, прогнозування ризиків, інтеграція моделей, фінансові технології.

Короткий зміст праці:

Дослідження присвячено комплексному аналізу, розробці та управлінню проєктом впровадження моделі машинного навчання для прогнозування кредитних ризиків у банківській сфері. У роботі розкрито сутність кредитних ризиків та особливості їх впливу на фінансову стабільність банків, здійснено всебічний огляд сучасних підходів і методів машинного навчання для оцінки кредитоспроможності позичальників.

Практична частина роботи включає побудову та тестування декількох моделей прогнозування ризиків з різними конфігураціями налаштувань. За результатами порівняльного аналізу за коефіцієнтом F1 найефективнішою моделлю виявилась Support Vector Machine з компонентами PCA (Principal Component Analysis) для зменшення розмірності даних та SMOTE oversampling для балансування класів у навчальній вибірці.

Значну увагу приділено управлінським аспектам впровадження розробленої моделі у банківські бізнес-процеси. Детально розглянуто етапи інтеграції моделі, включаючи підготовчий етап, технічне планування, розробку інтеграційного шару, пілотне впровадження та повномасштабне розгортання. Запропоновано використання канбан-методології для ефективного управління проєктом впровадження, що забезпечує прозорість процесів та можливість швидкого реагування на зміни.

У роботі розроблено комплексну систему моніторингу ефективності та адаптації моделі після запуску, яка включає автоматизовані механізми виявлення дрейфу даних, систему алертів та процедури безперервного вдосконалення. Сформовано детальний реєстр ризиків впровадження, що охоплює аспекти якості даних, технічної інтеграції, організаційного опору та інформаційної безпеки.

Практична значимість дослідження полягає у розробці конкретних рекомендацій для банківських установ щодо поетапного впровадження моделей машинного навчання, організації міждисциплінарних команд та забезпечення регуляторної відповідності. Запропонована дорожня карта проєкту з орієнтовною тривалістю 18-22 тижні демонструє реалістичність практичного застосування розроблених підходів.

Результати роботи можуть бути використані комерційними банками, фінтех-компаніями та регуляторними органами для оптимізації кредитної політики, зниження рівня неповернень та підвищення загальної фінансової стійкості банківської системи. Дослідження

також окреслює перспективні напрями подальших наукових робіт у сфері поєднання методів Data Science та управління проєктами у фінансових установах.

ABSTRACT
of the qualification work
for obtaining a master's degree

Topic*: *PROJECT MANAGEMENT OF IMPLEMENTATION OF A MACHINE LEARNING MODEL FOR PREDICTION OF CREDIT RISKS
Author: Yuliia Denysiuk

Scientific supervisor:
Yuriy Kleban

Protected «……»…… 20____ .

Explanatory note to the qualification work: *150 (number of pages) p., 40 (number of figures) figures, 7 (number of tables) tables, 5 (number of appendices) appendices, 60 (number of sources) sources.*

Keywords*: *credit risks, machine learning, Support Vector Machine, PCA, SMOTE oversampling, project management, kanban methodology, banking operations, risk forecasting, model integration, financial technologies.

Summary of the work:

The research is devoted to comprehensive analysis, development, and project management of implementing machine learning models for credit risk prediction in the banking sector. The work reveals the essence of credit risks and peculiarities of their impact on banks' financial stability, provides a comprehensive review of modern approaches and machine learning methods for assessing borrowers' creditworthiness.

The practical part of the work includes construction and testing of several risk prediction models with different configuration settings. Based on comparative analysis using the F1 coefficient, the most effective model proved to be Support Vector Machine with PCA (Principal Component Analysis) components for data dimensionality reduction and SMOTE oversampling for class balancing in the training sample.

Significant attention is paid to managerial aspects of implementing the developed model into banking business processes. The stages of model integration are examined in detail, including the preparatory stage, technical planning, integration layer development, pilot implementation, and full-scale deployment. The use of kanban methodology for effective project management is proposed, ensuring process transparency and ability to respond quickly to changes.

The work develops a comprehensive system for monitoring effectiveness and model adaptation after launch, which includes automated mechanisms for data drift detection, alert systems, and continuous improvement procedures. A detailed risk register for implementation is formed, covering aspects of data quality, technical integration, organizational resistance, and information security.

The practical significance of the research lies in developing specific recommendations for banking institutions regarding phased implementation of machine learning models, organization of interdisciplinary teams, and ensuring regulatory compliance. The proposed project roadmap with an estimated duration of 18-22 weeks demonstrates the realistic applicability of the developed approaches.

The research results can be used by commercial banks, fintech companies, and regulatory authorities to optimize credit policies, reduce default rates, and enhance overall financial stability of the banking system. The study also outlines promising directions for further scientific work in combining Data Science methods and project management in financial institutions.

ЗМІСТ

ВСТУП

У сучасних умовах розвитку фінансового сектору питання управління ризиками набуває особливої ваги. Банківські установи постійно стикаються з проблемою кредитних ризиків, тобто ймовірністю неповернення кредитних коштів позичальниками. Невдале управління цими ризиками може призвести до суттєвих фінансових втрат, втрати довіри з боку інвесторів та клієнтів, а в масштабах економіки – до виникнення кризових явищ. Традиційні статистичні методи оцінки кредитоспроможності, що базуються на фінансових коефіцієнтах, кредитних історіях та експертних висновках, сьогодні виявляються недостатньо ефективними через обмеженість у виявленні складних закономірностей у великих і неоднорідних наборах даних.

Зі стрімким розвитком цифрових технологій, накопиченням великих масивів фінансової інформації та поширенням штучного інтелекту відкриваються нові можливості для прогнозування кредитних ризиків. Методи машинного навчання дають змогу аналізувати тисячі змінних, будувати нелінійні залежності та виявляти приховані взаємозв’язки, що значно підвищує точність прогнозів. Крім того, застосування алгоритмів штучного інтелекту дозволяє здійснювати оперативну оцінку нових заявок на кредит та швидко адаптуватися до змін економічного середовища.

Актуальність дослідження полягає не лише у створенні моделей машинного навчання, але й у забезпеченні їх практичного впровадження в реальну діяльність банківських установ. Це передбачає ефективне управління проєктом інтеграції таких систем, врахування організаційних, технічних та економічних аспектів. Таким чином, тема роботи поєднує два важливі напрями – методологію побудови моделей прогнозування кредитних ризиків та управління проєктами їх впровадження.

Метою даного дослідження є комплексний аналіз, розробка та управління проєктом впровадження моделі машинного навчання для прогнозування кредитних ризиків у банківській сфері.

Для досягнення цієї мети у роботі поставлено такі завдання:

- розкрити сутність кредитних ризиків та особливості їх впливу на фінансову стабільність банків;
- здійснити огляд сучасних підходів і методів машинного навчання, що застосовуються для оцінки кредитоспроможності;
- визначити ключові метрики оцінки ефективності моделей (Accuracy, Precision, Recall, ROC-AUC) та економічні показники результативності;
- побудувати та протестувати модель прогнозування ризиків на основі реальних або відкритих наборів даних;
- оцінити економічний ефект від використання моделі у практичній діяльності банку;
- розробити план управління проєктом впровадження моделі в банківські бізнес-процеси, враховуючи часові, фінансові та ресурсні обмеження;
- сформувати рекомендації щодо застосування розроблених підходів у фінансовій сфері та визначити перспективні напрями подальших досліджень.

Об’єктом дослідження виступає процес управління проєктом впровадження інтелектуальних інформаційних систем у банківській сфері.

Предметом дослідження є методи машинного навчання для прогнозування кредитних ризиків та управлінські механізми інтеграції цих моделей у практичну діяльність фінансових установ.

Наукова новизна дослідження полягає у поєднанні підходів із двох різних сфер – Data Science та управління проєктами. Запропонований підхід передбачає оцінку побудованих моделей не лише з точки зору стандартних метрик точності, а й за економічними показниками, що дозволяє інтегрувати результати у процес прийняття управлінських рішень. Важливою складовою новизни є акцент на практичному впровадженні, оскільки ефективність навіть найточнішої моделі може бути знівельована при неправильному управлінні проєктом інтеграції.

Практична значимість полягає у можливості застосування результатів дослідження для оптимізації кредитної політики банківських установ, зниження

рівня неповернень та підвищення фінансової стійкості. Запропоновані методики дозволяють не лише автоматизувати процес оцінки заявок на кредит, але й забезпечити прозорість та контрольованість проєкту впровадження. Отримані висновки та рекомендації можуть бути корисними для комерційних банків, фінтех-компаній та інших установ, що працюють із кредитними продуктами.

Структура дослідження побудована логічно та відповідає поставленим завданням. Робота складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі представлено теоретичні аспекти теми: сутність та класифікація кредитних ризиків, сучасні підходи до їх прогнозування, огляд літератури та аналіз методів машинного навчання, які застосовуються у фінансовому секторі.

Другий розділ містить практичну частину, де здійснюється побудова моделей машинного навчання, їх тренування та тестування, аналіз результатів за допомогою статистичних і економічних метрик, а також розрахунок економічного ефекту від їх застосування.

У третьому розділі розглядається управління проєктом впровадження створеної моделі у практичну діяльність банку. Тут подається опис проєктного підходу, планування етапів, розподіл ресурсів, оцінка ризиків та організаційних бар’єрів, що можуть виникнути у процесі інтеграції.

У висновках узагальнено результати роботи, сформульовано основні рекомендації та визначено перспективи подальших наукових досліджень у сфері поєднання управління проєктами та машинного навчання у фінансових установах.

РОЗДІЛ 1

ТЕОРЕТИЧНІ АСПЕКТИ ВПРОВАДЖЕННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ У ПРОГНОЗУВАННІ КРЕДИТНИХ РИЗИКІВ

1.1. Сутність та значення управління проєктами у фінансовій сфері

Управління проєктами у фінансовій сфері розглядається як невід'ємний елемент сучасного стратегічного менеджменту, що дає змогу досягати конкретних цілей у межах визначених часових, ресурсних та бюджетних обмежень. Фінансова сфера включає банки, страхові компанії, інвестиційні фонди, фінтех-стартапи та інші інституції, діяльність яких тісно пов'язана з акумуляцією та перерозподілом фінансових ресурсів. Для них здатність реалізовувати інноваційні проєкти, адаптуватися до ринкових змін та інтегрувати сучасні технології є запорукою ефективності й конкурентоспроможності.

Сутність управління проєктами у цій галузі полягає в системному підході до планування, організації, контролю й завершення завдань, що спрямовані на створення нових продуктів чи вдосконалення бізнес-процесів. На відміну від поточної операційної діяльності, проєкти у фінансовому секторі завжди мають чіткі часові рамки, обмеження ресурсів та унікальний результат. Наприклад, запуск мобільного застосунку для онлайн-банкінгу або впровадження автоматизованої системи оцінки ризиків вимагає чіткого управління, адже такі завдання не лише створюють нову цінність для клієнтів, а й формують конкурентні переваги компанії. Крім того, проєкти у фінансовій сфері завжди тісно пов'язані з вимогами регуляторів, що зумовлює необхідність врахування міжнародних стандартів, таких як Basel III чи IFRS, а також національного законодавства.

Значення управління проєктами для фінансових установ проявляється у здатності забезпечувати їхній інноваційний розвиток, цифрову трансформацію та стійкість у мінливому ринковому середовищі. Так, саме завдяки

організованому підходу банки успішно впроваджують нові продукти, як-от сервіси миттєвих платежів чи інтеграцію міжнародних платіжних систем, що підвищує рівень клієнтської лояльності та зміцнює позиції на ринку. Цифровізація бізнес-процесів є ще одним прикладом, коли управління проєктами стає ключовим інструментом: системи автоматизованого скорингу чи застосування штучного інтелекту в обробці клієнтських даних дозволяють скоротити час ухвалення рішень і мінімізувати ризики.

Не менш важливою є й роль управління проєктами у сфері ризик-менеджменту. Фінансові організації постійно стикаються з викликами, пов’язаними з неплатоспроможністю клієнтів, коливанням валютних курсів чи зростанням шахрайських схем. Реалізація проєктів зі створення систем прогнозування дефолтів або впровадження аналітичних платформ для моніторингу транзакцій допомагає значно знизити ймовірність фінансових втрат. Крім цього, завдяки чіткій організації проєктної діяльності компанії вчасно й системно реагують на зміни в нормативно-правовому полі, наприклад, під час переходу на міжнародні стандарти звітності.

Практичні приклади підтверджують цінність такого підходу. Компанія Revolut, завдяки грамотно організованому управлінню портфелем проєктів, змогла створити цілу цифрову екосистему, яка охоплює мультивалютні рахунки, миттєві перекази та криптовалютні сервіси. В Україні успішною стала реалізація проєктів із впровадження дистанційної ідентифікації через BankID та Diia.ID, що дало змогу банкам значно спростити процес відкриття рахунків та розширити доступ населення до фінансових послуг. Страхові компанії, своєю чергою, активно використовують телематичні технології для формування індивідуальних тарифів у страхуванні транспорту, що також є результатом ефективного управління проєктами.

Таким чином, управління проєктами у фінансовій сфері виконує не лише технічну чи організаційну функцію, а й стає стратегічним інструментом забезпечення довгострокової стабільності та розвитку. Його значення полягає в тому, що воно дозволяє поєднувати прибутковість із контрольованим рівнем

ризику, підтримувати відповідність регуляторним вимогам і водночас створювати нову цінність для клієнтів. У сучасних умовах ефективне управління проєктами перетворюється на ключовий фактор успіху фінансових інституцій, які прагнуть зберігати конкурентоспроможність та розвиватися в умовах глобальної економічної динаміки.

Розвиток фінансового сектору неможливий без впровадження системних методів управління проєктами, адже саме вони забезпечують структурованість процесів, контроль за використанням ресурсів та досягнення визначених результатів. Методології управління проєктами виконують роль своєрідних рамок, у межах яких проєктна команда рухається від ідеї до практичного впровадження. Вони визначають підхід до планування, організації, контролю та завершення проєктів, а їх правильний вибір у фінансовій сфері має вирішальне значення, адже будь-які помилки у проєктах можуть призвести не лише до фінансових втрат, а й до втрати довіри клієнтів чи навіть до порушення вимог регуляторів.

Однією з найбільш відомих і поширених методологій є PMBOK (Project Management Body of Knowledge), яка пропонує комплексний підхід до управління проєктами через виділення знань, процесів та областей, важливих для успішної реалізації. У фінансовій сфері PMBOK застосовується для проєктів з високим рівнем регуляторних вимог, наприклад, під час впровадження нових систем звітності або інтеграції стандартів міжнародного фінансового обліку. Завдяки чіткому розподілу фаз та обов’язків ця методологія забезпечує прозорість процесів та дає змогу знизити ймовірність невідповідностей чи помилок.

Іншим підходом, який набув особливої популярності в останні роки, є Agile. Його ключова ідея полягає у гнучкості, швидкому реагуванні на зміни та залученні клієнтів до процесу створення продукту. У фінансовому секторі Agile активно використовується під час розробки цифрових сервісів: мобільних додатків для інтернет-банкінгу, платформ для онлайн-кредитування чи страхових калькуляторів. Цей підхід дозволяє компаніям швидко тестувати нові

рішення на обмеженій групі користувачів, отримувати зворотний зв’язок і вносити корективи без втрати часу та зайвих витрат. Прикладом може слугувати впровадження у банках системи миттєвих переказів, коли завдяки Agile команда розробників могла поступово вдосконалювати інтерфейс і функціонал відповідно до реальних потреб клієнтів.

Ще однією важливою методологією є PRINCE2 (Projects IN Controlled Environments), що базується на принципі контролю за кожним етапом реалізації проєкту. На відміну від Agile, PRINCE2 орієнтується на жорстку структуру й підзвітність, що робить її надзвичайно корисною у великих фінансових корпораціях та міжнародних банках, де будь-яке відхилення може призвести до значних ризиків. Використання PRINCE2 забезпечує детальну документацію, чітке визначення ролей і контроль за ресурсами, завдяки чому мінімізуються непередбачувані витрати та підвищується рівень управлінської дисципліни.

Сучасна практика свідчить, що у фінансових організаціях дедалі частіше використовується гібридний підхід, коли поєднуються елементи кількох методологій. Наприклад, стратегічно важливі проєкти, пов’язані з регуляторною звітністю, можуть управлятися за принципами PMBOK чи PRINCE2, тоді як цифрові інновації реалізуються на основі Agile або Scrum. Така комбінація дозволяє не лише дотримуватися високих стандартів контролю, але й зберігати гнучкість у швидкозмінному середовищі.

Значення методологій у фінансовій сфері полягає в тому, що вони знижують ризики, пов’язані з реалізацією проєктів, забезпечують передбачуваність результатів і водночас створюють умови для інновацій. Вибір конкретного підходу залежить від масштабу проєкту, його стратегічної важливості та рівня невизначеності. У результаті правильно організоване управління проєктами стає не лише інструментом досягнення поставлених цілей, а й фактором формування довгострокових конкурентних переваг фінансової організації.

1.2. Поняття та класифікація кредитних ризиків

У фінансовій сфері однією з найбільш критичних категорій ризиків є кредитний ризик, адже саме він безпосередньо впливає на стабільність банківської системи та фінансових установ загалом. Під кредитним ризиком зазвичай розуміють ймовірність того, що позичальник або контрагент не зможе виконати свої фінансові зобов'язання в повному обсязі та у визначені строки. Іншими словами, це ризик втрат для банку чи іншої фінансової організації, пов'язаних з невиплатою основної суми боргу, відсотків чи інших платежів.

Кредитний ризик виникає внаслідок об'єктивної невизначеності щодо майбутнього фінансового стану позичальника. Навіть за умови ретельного аналізу та попередньої перевірки клієнта не існує стовідсоткової гарантії, що зобов'язання будуть виконані. Наприклад, банк може надати позику підприємству з високим рівнем рентабельності, проте непередбачені обставини, такі як економічна криза, втрата ключових ринків збуту або проблеми з ліквідністю, здатні призвести до дефолту.

У фінансовій практиці існує кілька класифікацій кредитних ризиків залежно від обраних критеріїв. Найбільш поширеною є класифікація за об'єктом ризику. У цьому випадку виділяють ризики, пов'язані з фізичними особами, які отримують споживчі кредити чи іпотеку, та ризики корпоративного кредитування, що стосуються підприємств та організацій. Окремо розглядається ризик державних позичальників, коли йдеться про обслуговування державного боргу чи гарантій.

Інший підхід полягає у класифікації за формою прояву ризику. Тут можна виокремити ризик неповернення кредиту внаслідок банкрутства позичальника, ризик прострочення платежів, що створює проблеми з ліквідністю банку, а також ризик реструктуризації боргу, коли позичальник не здатен виконати зобов'язання у первісно погоджених умовах і вимагає перегляду графіка чи відсоткової ставки.

Важливим є й розподіл за рівнем концентрації. Якщо кредитний ризик зосереджений на одному великому позичальнику, він вважається індивідуальним і може суттєво вплинути на фінансовий стан установи. Натомість портфельний ризик стосується всієї сукупності кредитних операцій банку і визначається якістю управління кредитним портфелем. Саме портфельний підхід дозволяє оцінювати ризики системно, застосовуючи інструменти математичного моделювання та статистичного аналізу.

Окремо виділяють класифікацію за ступенем забезпечення. У цьому випадку існують забезпечені кредити, де ризик частково компенсується заставою, гарантіями чи страховими інструментами, та незабезпечені кредити, що несуть вищий рівень ризику через відсутність додаткових механізмів захисту.

Сучасна практика також виокремлює кредитний ризик країн, який стосується іноземних позичальників і залежить від економічної та політичної ситуації в країні їхнього резидентства. Наприклад, навіть фінансово стійка компанія може виявитися неплатоспроможною через валютні обмеження чи санкції в її державі.

Таким чином, класифікація кредитних ризиків має не лише теоретичне значення, але й вагому практичну роль, адже вона допомагає фінансовим установам формувати ефективні системи управління ризиками, розробляти внутрішні політики кредитування та будувати адекватні математичні моделі для оцінювання ймовірності дефолту. Саме системне розуміння природи кредитних ризиків і їхньої багатогранності є ключем до підвищення фінансової стійкості як окремих установ, так і банківського сектору в цілому.

1.3. Роль машинного навчання у прогнозуванні кредитних ризиків

Роль машинного навчання у прогнозуванні кредитних ризиків визначається його здатністю виявляти складні нелінійні взаємозв’язки між великою кількістю ознак позичальника, динамікою його поведінки та макроекономічним контекстом. На відміну від традиційних статистичних підходів, що зазвичай ґрунтуються на лінійних припущеннях і фіксованих

правилах, алгоритми ML уміють автоматично будувати взаємодії ознак, працювати з неоднорідними джерелами даних і адаптуватися до змін середовища. Це дозволяє зменшувати частку помилково схвалених заявок, скорочувати час розгляду та підвищувати відсоток «якісно» виданих кредитів, не втрачаючи в обсягах портфеля.

Практична цінність ML починається з даних. Банки поєднують класичні кредитні бюро, внутрішні транзакційні історії, скорингові анкети, логи користування мобільним застосунком, дані про пристрій і гео-поведінку, а в корпоративному сегменті – фінансову звітність, контракти та ланцюжки постачання. Сировинні дані майже завжди неповні, зашумлені й розрізнені у часі, тому першим кроком стає методична інженерія ознак: конструювання показників використання ліміту, кредитної історії в часі, стабільності доходів, сезонності витрат, темпів зростання оборотів, «свіжості» негативних подій, а також агрегатів на рівні клієнта, продукту та каналу. Для високорівневих категоріальних змінних застосовують цільове або частотне кодування, для часових – ковзні вікна й лагові різниці; пропуски обробляють по-різному залежно від природи сутності (наприклад, «відсутність витрат» часто є інформацією, а не помилкою). Важливо уникати витоку інформації: усі перетворення мають навчатися лише на тренувальному періоді та валідовуватися на майбутніх відтинках часу.

Оскільки дефолти зазвичай рідкісні, задачу ускладнює дисбаланс класів. Тому, окрім ваг для класів, використовують методи зважених функцій втрат, фокальну втрату, а інколи синтетичне добудовування міноритарного класу. Однак ключовим є не «вирівняти» класи штучно, а коректно налаштувати мету: замість безумовної максимізації точності у кредитному ризику оптимізують очікувану економічну вигоду. Якщо для кожної заявки відомі приблизні «ціна помилки» та прибуток від справного позичальника, поріг прийняття рішення обирають так, щоб максимізувати $E(\text{прибуток}) = p(\text{недефолт}) \cdot G - p(\text{дефолт}) \cdot C$, де G – очікуваний дохід (процентна маржа з урахуванням витрат на залучення), а C – очікувані збитки з урахуванням стягнення та вартості резервування. У такій

постановці навіть модель із помірним ROC-AUC може бути більш корисною за «статистично сильнішу», якщо вона краще відокремлює сегменти з позитивною маржею.

Вибір алгоритму залежить від типу даних, масштабів і регуляторних обмежень. Логістична регресія залишається надійним еталоном завдяки стабільності, швидкості та легкій інтерпретації; її часто поєднують із монотонними бінінгами ознак, отримуючи класичні скорингові картки. Для виявлення нелінійностей і взаємодій застосовують ансамблеві дерева – градієнтний бустинг і випадкові ліси; у кредитному ризику особливо добре зарекомендували себе XGBoost/LightGBM завдяки роботі з пропусками та швидкій оптимізації. У задачах з послідовностями подій та історіями транзакцій корисними бувають нейромережеві підходи (наприклад, прості одношарові мережі з увагою або рекурентні/трансформерні архітектури для витягування поведінкових патернів), але їх застосування вимагає підсиленої уваги до пояснюваності й контролю ризиків моделі. Для PD-моделювання з часовою залежністю доречні моделі виживаності (hazard-підходи), які оцінюють інтенсивність настання дефолту в часі та природно узгоджуються зі сценарним аналізом і стрес-тестуванням.

Окрема роль ML – у повному контурі ризик-параметрів: PD (імовірність дефолту), LGD (втрати у випадку дефолту) та EAD (обсяг заборгованості на момент дефолту). Для PD зазвичай будують бінарні або часові моделі; LGD моделюють як регресію з цензуруванням і перекошеним розподілом, іноді – двоетапно (спочатку ймовірність стягнення, потім величина втрат умовно на подію), а EAD – із урахуванням ліміту та динаміки вибірки коштів. Машинне навчання покращує точність кожного компонента, а отже, і очікуваних кредитних збитків (ECL) за стандартами IFRS 9, що напряму впливає на резерви та капітал.

Щоб моделі були придатні до прийняття рішень, важлива калібровка. Навіть дуже «ранжуюча» модель може повертати ймовірності, що системно завищені чи занижені; застосовують ізотонічну регресію або Platt-калібрування

й регулярно перевіряють надійність через діаграми калібровки та Brier score. Контроль стабільності здійснюють з використанням PSI/CSI і відстеження розподілів ознак. Паралельно оцінюють не лише ROC-AUC чи PR-AUC, а й бізнес-криві прибутку, кумулятивні «profit lift» та вплив на портфельні KPI.

У реальному банкінгу критичною є пояснюваність. Методи SHAP і LIME дозволяють отримати як глобальне уявлення про важливість ознак, так і локальні «reason codes» для конкретного рішення. Це дає можливість обґрунтувати відмову або погодження кредиту, підтримати права клієнта та виконати вимоги регуляторів щодо прозорості. Додатково застосовують часткові залежності та накопичені локальні ефекти, а в бустингу – монотонні обмеження, щоб забезпечити «здоровий глузд» моделі: погіршення значущих показників не повинно зменшувати оцінений ризик. У чутливих сегментах запроваджують політики відсікання ознак-проксі, регуляризацію за справедливістю та тестування на відповідність принципам рівності можливостей, аби знизити алгоритмічну упередженість.[51]

Складність вибірки породжує ще одну прикметну проблему – «reject inference». Оскільки банк спостерігає фактичні дефолти лише для схвалених заявок, навчальні дані мають системне упередження. Для його зменшення використовують повторне зважування за ймовірністю відбору, моделі двоетапного вибору, а також контрольоване добудовування даних зі «сірими зонами». Якість цих процедур істотно впливає на переносимість моделі з пілоту на повний продакшн.

Життєвий цикл ML-рішення набагато ширший за розробку. У банку модель проходить стадії дослідження, валідації, пілоту промислового розгортання та експлуатації. Практики MLOps забезпечують відстежуваність даних і коду, контроль версій артефактів, відтворюваність навчання, автоматизовані тести на зсув концепції та деградацію якості, а також безпечне оновлення з катастрофічним відкотом. Регулярний моніторинг включає технічні показники (латентність, відсоток помилок), статистичні (дрейф ознак, стабільність калібровки), бізнес-метрики (частка схвалень, дефолтність за

ризиковими бінами, маржинальний прибуток) і комплаєнс-сигнали (справедливість, пояснюваність, стабільність політик). Саме завдяки такому «керованому континууму» ML перестає бути лабораторним експериментом і стає надійним бізнес-механізмом.

Роль ML не зводиться лише до фронтового скорингу. Алгоритми оцінюють ліміти й ціноутворення, оптимізують стратегії колекшну (динамічні пріоритети контактів, персоналізовані пропозиції реструктуризації), підказують превентивні дії для утримання якісних клієнтів, а також допомагають у виявленні шахрайства. На портфельному рівні моделі прогнозують міграцію між «стейджами» IFRS 9, чутливість до макроіндикаторів та результат сценарних шоків, створюючи базу для рішень ALM і капітального планування. Поєднання скорингу на рівні індивіда з портфельними симуляціями підсилює керованість ризику в різних горизонтах.

Разом із перевагами зростають і вимоги. Регулятори очікують від банків модульного підходу до управління ризиками моделей: незалежну валідацію, документовану логіку, контроль змін, періодичні стрес-тести та аудит даних. Права клієнта на приватність диктують принципи мінімізації даних і чіткі правові підстави їх використання; будь-яке залучення альтернативних джерел повинно мати прозоре обґрунтування, а результати – не призводити до системної дискримінації. В операційному вимірі моделі мають бути стійкими до «геймінгу»: якщо заявники можуть пристосуватися до правил, слід змінювати ознаки, виявляти аномальну поведінку і поєднувати автоматичні рішення з вибірковою ручною перевіркою.

У підсумку машинне навчання відіграє подвійну роль. З одного боку, це інструмент підвищення точності та швидкості рішень у кредитному циклі; з іншого – каталізатор перебудови операційних процесів і культури управління ризиком. Ефект досягається не стільки за рахунок «найскладнішого алгоритму», скільки завдяки дисципліні даних, бізнес-орієнтованій метриці успіху, пояснюваності, етичній відповідальності та зрілій інженерії впровадження. Саме така інтегрована парадигма робить ML невід’ємною опорою сучасного

прогнозування кредитних ризиків і підвищує стійкість фінансових установ у мінливому середовищі.

1.4. Огляд літератури та існуючих методів прогнозування

У науковій літературі останніх десятиліть значна увага приділяється питанням прогнозування кредитних ризиків та застосуванню машинного навчання для підвищення точності оцінювання. Дослідження, виконані зарубіжними авторами, демонструють інтенсивний розвиток методологічної бази та зростання інтересу до поєднання традиційних економетричних моделей із сучасними алгоритмами штучного інтелекту.

Систематичний огляд S. Borrelli, A. Ianneli та співавторстві узагальнює 76 праць за останні роки й пропонує класифікацію ML-підходів до кредитного ризику (деревні ансамблі, SVM, нейромережі, гібридні моделі), а також «рейтинги» на відкритих наборах даних. Автори підкреслюють, що якість рішень істотно залежить від особливостей даних (дисбаланс класів, витік інформації, вибір метрик), і закликають оцінювати моделі не лише ROC-AUC, а й бізнес-орієнтованими показниками та стабільністю в часі. Огляд також фіксує тренд на пояснюваність («reason codes») і MLOps-процедури для продуктивної експлуатації моделей. [2]

Практичне дослідження у журналі Risks (MDPI) порівнює логістичну регресію, нейронні мережі, AdaBoost/XGBoost/LightGBM на задачі дефолту за кредитними картками. За звітами авторів, XGBoost показав найвищі результати за стандартними метриками (у статті наведено дуже високу точність), водночас робота послідовно аналізує й інші метрики (Precision, Recall, F1, ROC, MCC), що важливо для дисбалансних задач. Зміст підкреслює перевагу градієнтного бустингу на табличних даних, але водночас ілюструє ризик «оптимізму» на конкретних вибірках (залежність від джерела даних/валідації). [3]

Стаття в European Journal of Operational Research про інтерпретоване ML на дисбалансних наборах показує: зі зростанням дисбалансу (особливо коли дефолтів <5%) локальні пост-хок пояснення LIME/SHAP стають менш

стабільними (коливається порядок важливостей, зростає варіативність). Для практики це означає потребу в обережній інтерпретації, калібровці та перевірках стабільності пояснень, а також у поєднанні з глобальними обмеженнями/монотонністю. [1]

Класичний огляд Crook, Edelman & Thomas (2007) окреслює фундамент сучасного скорингу: логістична регресія й скорингові картки, відбір ознак і бінінг, «reject inference», частотні/байєсівські підходи та ранні ансамблі. Попри давність, він лишається ключовою точкою відліку для політик відбору змінних, побудови карток та регуляторної інтерпретованості.[7]

Публікація НБУ (Вісник НБУ) описує «економетричну» модель оцінки кредитного ризику та альтернативу на базі SVM, обґрунтовуючи вибір моделі вимогами практичності, даних і регуляторної відтворюваності. У статті підкреслюється еволюція від традиційних статистичних моделей (z-score Альтмана) до ML-підходів (SVM, ANN) і пояснюється, чому регулятор схиляється до стандартизованих, контрольованих методик у банківському нагляді (стала методологія, можливість аудиту, єдині правила для банків). [9]

Праця Дацько та Друшчак (ЛНУ, 2022) стосується моделювання скорингу методами ML у прикладному академічному ключі: робота описує місце кредитного скорингу серед задач класифікації, оглядає популярні алгоритми та питання якості даних. Цінність статті – у систематизації методів і постановці задачі для українського читача/практика, однак обмеження – відносно невелика емпірична база та фокус на загальнотеоретичному огляді, а не на великомасштабних промислових експериментах.[12]

Методологічна глибина та масштаб емпірії. Міжнародні огляди працюють із великими корпусами статей і відкритими наборами (German Credit, Taiwanese default, Home Credit тощо), що дозволяє робити узагальнення щодо відносних переваг бустингу, SVM і NN, а також ретельно досліджувати питання дисбалансу та інтерпретації. Сильна сторона – широка вибірка методів/даних, слабка – ризик «бенчмарк-оптимізму» та обмежена переносимість результатів у строгу регуляторну практику конкретної країни. [13]

Українські роботи, особливо НБУ, більше прив’язані до практики нагляду й звітності: акцент на моделі, які можна аудіювати, відтворити та пояснити (логістика/SVM), і на процедурних аспектах – єдині підходи для банків, відповідність стандартам, стабільність у часі. Це знижує ризик «чорної скриньки», але може гальмувати впровадження новітніх deep-підходів без додаткових інструментів пояснюваності й контролю.[9]

Закордонні праці детально досліджують стійкість локальних пояснень і показують їхню чутливість до дисбалансу; це корисний сигнал і для українських банків, де частка дефолтів часто низька. Практичний висновок: поряд із LIME/SHAP потрібні глобальні обмеження (монотонність, стабільний бінінг), калібровка й валідація стабільності пояснень у часі.[51]

Бізнес-метрики проти «суто статистики». Закордонні дослідження дедалі частіше просувають економічно зважені метрики (MCC, прибуткові криві, cost-sensitive learning), тоді як у регуляторно орієнтованих публікаціях важливішими лишаються стабільність, контрольованість і відповідність політикам нагляду. Баланс між цими підходами – ключ до успішного впровадження: модель має одночасно проходити наглядову перевірку та покращувати економіку портфеля.

Якщо порівняти зарубіжні дослідження з українськими, то вітчизняні науковці, зокрема О. І. у своїй статті для журналу НБУ[9], роблять акцент переважно на економетричних моделях, таких як логістична регресія. Такий підхід є зрозумілим, оскільки він забезпечує високу прозорість і відповідає вимогам регулятора, але при цьому поступається в точності сучасним ML-алгоритмам. Робота Дацька та Друшчака (2022)[12] показує поступове проникнення методів машинного навчання у вітчизняні дослідження. Автори демонструють ефективність алгоритмів SVM, дерева рішень та ансамблевих методів у порівнянні з класичною логістичною регресією. Водночас вони зазначають, що проблеми доступності якісних даних та недостатня обчислювальна інфраструктура залишаються ключовими бар’єрами для широкого впровадження сучасних моделей у фінансовому секторі України.

Таким чином, зарубіжні дослідження характеризуються ширшим спектром використаних методів, включно з глибинними нейронними мережами та інтерпретованим ML, тоді як українські роботи демонструють обережне впровадження нових технологій на тлі сильного акценту на прозорості моделей і відповідності регуляторним вимогам.

1.5. Порівняльний аналіз традиційних статистичних підходів та методів машинного навчання

Оцінювання кредитних ризиків історично базувалося на статистичних підходах, які довгий час були основним інструментом для банків та фінансових установ. Найбільш поширеним методом залишається логістична регресія, яка дозволяє на основі обмеженого набору змінних визначити ймовірність дефолту позичальника. Популярність цього підходу пояснюється його простотою, математичною прозорістю та відповідністю регуляторним вимогам. Логістична регресія забезпечує легкість у поясненні результатів, що є важливим фактором у банківському середовищі, де кожне рішення повинно бути обґрунтованим і зрозумілим як для менеджменту, так і для регуляторів. Крім того, класичні статистичні моделі добре працюють у ситуаціях, коли дані є відносно структурованими, збалансованими та мають лінійні залежності.

Проте із розвитком цифрової економіки та появою великих масивів різнорідних даних виявилося, що традиційні моделі не завжди здатні відображати складні й нелінійні взаємозв’язки, властиві фінансовій поведінці клієнтів. Саме тут свою роль починають відігравати методи машинного навчання. Алгоритми, такі як дерева рішень, випадкові ліси, метод опорних векторів та нейронні мережі, демонструють значно вищу гнучкість і здатність адаптуватися до нових умов. Наприклад, дерево рішень здатне враховувати складні комбінації факторів, які традиційна модель могла б ігнорувати. Ансамблеві методи, на кшталт випадкових лісів чи градієнтного бустингу, дозволяють зменшити ризик перенавчання та підвищити точність прогнозування. Нейронні мережі, своєю чергою, відкривають можливості

роботи з великими, слабо структурованими даними, зокрема транзакційними записами, поведінковими патернами клієнтів та навіть текстовими або соціальними даними.

Разом із тим, використання ML-методів у фінансовій сфері пов’язане з низкою проблем. Насамперед ідеться про так звану «чорну скриньку» – складність пояснення того, як модель прийшла до конкретного рішення. Для банків, що працюють у жорсткому регуляторному середовищі, це є серйозною перешкодою. Крім того, побудова ефективних моделей машинного навчання потребує великих обсягів якісних даних і значних обчислювальних ресурсів, що не завжди доступно фінансовим установам у країнах із менш розвиненою інфраструктурою.[43]

Якщо порівнювати ці два підходи в комплексі, то статистичні моделі залишаються більш зручними з точки зору прозорості, стабільності та простоти реалізації. Вони ідеально підходять для базових задач кредитного скорингу, де необхідно швидко і зрозуміло оцінити ймовірність дефолту позичальника. Методи машинного навчання, своєю чергою, виявляють свою перевагу у складних та динамічних умовах, коли дані є різноманітними, обсяг інформації зростає, а традиційні моделі втрачають точність.[8][10]

У цьому сенсі майбутнє прогнозування кредитних ризиків бачиться у комбінуванні обох підходів. Уже сьогодні деякі банки застосовують гібридні моделі, у яких логістична регресія використовується як базовий метод оцінки, тоді як машинне навчання залучається для виявлення прихованих залежностей або для аналізу додаткових джерел даних. Такий підхід дозволяє зберегти баланс між прозорістю, вимогами регуляторів та підвищенням точності прогнозів.[26]

Таким чином, статистичні методи можна розглядати як фундамент, на якому базується система управління кредитними ризиками, тоді як машинне навчання є перспективним інструментом, що поступово розширює її можливості. Еволюція у цьому напрямку свідчить про поступовий перехід від класичних до сучасних методів, однак не через заміну, а через доповнення та інтеграцію, що є характерним для сучасного фінансового аналізу.

1.6. Моделі машинного навчання, що використані в дослідженні

1.6.1. Особливості методу логістичної регресії

Узагальнені лінійні моделі (УЛМ) – це широкий клас моделей, що забезпечують об'єднуючу основу для багатьох поширених статистичних методів, таких як лінійна регресія. Вони характеризуються такими трьома компонентами.

Випадковий компонент визначає змінну відгуку Y та припускає для неї розподіл ймовірностей. Стандартні УЛМ розглядають m спостережень Y як незалежні, і ми позначаємо ці спостереження як (y1,...,ym).

Лінійний предиктор визначає пояснювальні змінні. Назва відображає, що змінні лінійно входять як предиктори у правій частині рівняння моделі у вигляді

$$\beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n = \mathbf{x}^\top \beta \tag{1}$$

Функція зв'язку визначає функцію від $\mathbb{E}$(Y), яку GLM пов'язує з лінійним предиктором.

Отже, в GLM очікувана відповідь для заданого вектора ознак x = [x1, …, xn]^T має вигляд

$$\mathbb{E}[Y|X = \mathbf{x}] = h(\mathbf{x}^\top \beta) \tag{2}$$

для деякої функції активації h, яка є оберненою до функції зв'язку.

В свою чергу логістична регресія – це статистичний метод і один із базових алгоритмів машинного навчання, який використовується для задач класифікації. На відміну від лінійної регресії, що передбачає неперервні значення, логістична регресія прогнозує ймовірність належності об’єкта до певного класу (наприклад, «так/ні», «0/1», «позитивний/негативний результат»).

Модель ґрунтується на застосуванні сигмоїдної (логістичної) функції до лінійної комбінації ознак. Це дозволяє перетворити будь-яке числове значення у діапазон (0, 1), який інтерпретується як ймовірність.

$$P(Y_i = 1|X = \mathbf{x}_i) = h(\mathbf{x}_i^\top \beta) = \frac{e^{\mathbf{x}_i^\top \beta}}{1 + e^{\mathbf{x}_i^\top \beta}} \tag{3}$$

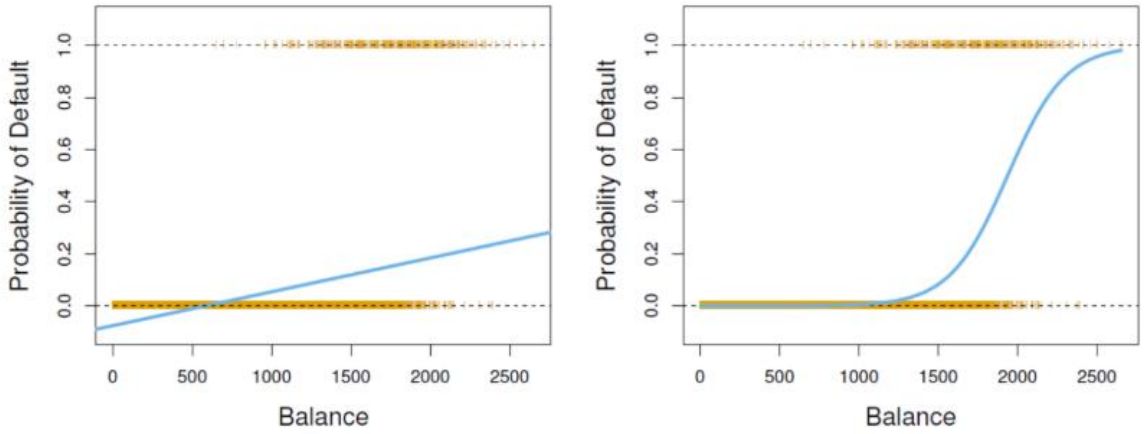


Рис.1.1. Порівняння лінійної та логістичної функцій [57]

У процесі класифікаційного аналізу важливо правильно інтерпретувати залежність між ознаками та цільовою змінною. Використання лінійної регресії для прогнозування ймовірності події є проблематичним, оскільки модель може повертати значення, що виходять за межі інтервалу [0,1]. Це добре видно на лівому графіку, де частина прогнозів виявляється від'ємною або більшою за одиницю, що суперечить ймовірнісній інтерпретації. Такий підхід не здатен адекватно відобразити нелінійний характер зв'язку між балансом і фактом дефолту.[40]

Натомість логістична регресія, представлена на правому графіку, використовує сигмоїдну функцію зв'язку, яка обмежує значення прогнозованих ймовірностей у межах від 0 до 1. Крива має S-подібну форму, що відповідає реалістичному сценарію: при низьких значеннях балансу ймовірність дефолту є близькою до нуля, тоді як при високих балансах вона наближається до одиниці. У середньому діапазоні балансів модель фіксує зону найбільшої невизначеності, де ймовірність змінюється найшвидше.[56]

Таким чином, логістична регресія забезпечує коректне ймовірнісне моделювання та є адекватним інструментом для вирішення задач бінарної класифікації. На відміну від лінійної регресії, вона враховує специфіку ймовірностей, гарантує інтерпретацію результатів у правильних межах і відображає закономірності даних у більш природний спосіб.

1.6.2. Особливості методів Дерев Рішень та Random Forest

Деревоподібні методи забезпечують простий, інтуїтивно зрозумілий та потужний механізм як для регресії, так і для класифікації. Основна ідея полягає в стратифікації (потенційно складного) простору ознак на менші області та підборі простої функції прогнозування для кожної області. Для класифікації заданого спостереження ми зазвичай використовуємо значення модальної реакції для навчальних спостережень в області, до якої воно належить. Оскільки набір правил розділення, що використовуються для сегментації простору предикторів, можна звести до дерева, ці типи підходів відомі як методи дерева рішень.

Класифікатори дерев рішень є привабливими моделями, якщо нам потрібна інтерпретованість. Вони передбачають створення набору бінарного розбиття на змінні-предиктори, щоб створити дерево, яке можна використовувати для класифікації нових спостережень в одну з двох груп.

Використовуючи алгоритм прийняття рішень, ми починаємо з кореня дерева та розділяємо дані за ознакою, яка призводить до найбільшого інформаційного приросту (IG):

- якщо предиктор є неперервним, ми вибираємо точку відсіку, яка максимізує чистоту для двох створених груп;
- якщо змінна-предиктор є категоріальною, ми об'єднуємо категорії, щоб отримати дві групи з максимальною чистотою.

В ітеративному процесі ми можемо повторювати цю процедуру розділення на кожному дочірньому вузлі, доки листя не стануть чистими. Це означає, що навчальні приклади на кожному вузлі належать до одного класу. На жаль, цей процес має тенденцію створювати дерево, яке є занадто великим і страждає від перенавчання. Таким чином, ми зазвичай обрізаємо дерево, встановлюючи обмеження на максимальну глибину дерева.[1]

Розглядаючи бінарне дерево рішень, ми визначаємо IG на кожному розділенні наступним чином:

$$IG(D_p, f) = I(D_p) - \sum_{j \in \{left, right\}} \frac{N_j}{N_p} I(D_j) \quad (4)$$

де,

f – ознака для виконання розділення;

Dp та Dj – набір даних батьківського та j-го дочірнього вузлів;

I – наша міра домішок;

Np – загальна кількість навчальних прикладів на батьківському вузлі, а Nj – кількість прикладів у j-му дочірньому вузлі.

Отже, інформаційний приріст – це просто різниця між домішками батьківського вузла та сумою домішок дочірніх вузлів. Щоб розділити вузли за найбільш інформативними ознаками, нам потрібно максимізувати IG при кожному розділенні.

Коефіцієнт Джині (Ig) та ентропія (Ih) є найчастіше використовуваними критеріями розділення в бінарних деревах рішень. Визначивши як p(i|t) частку прикладів, що належать до класу i для певного вузла t, ми можемо записати ентропію як:

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t) \quad (5)$$

та коефіцієнт Джині як:

$$I_G(t) = \sum_{i=1}^{c} p(i|t)(1 - p(i|t)) \quad (6)$$

Random Forest (випадковий ліс) - це алгоритм машинного навчання, що використовується для задач класифікації та регресії. Він заснований на ансамблі дерев рішень, де кожне дерево будується на випадковій підмножині навчальних даних та випадкових підмножинах характеристик.

Кожне дерево випадкового лісу навчається на випадковій підмножині даних, взятій з навчальної вибірки з повтореннями. Це означає, що кожне дерево

отримує лише підмножину даних та характеристик для навчання, що зменшує взаємозв'язок між деревами та забезпечує більшу стійкість до перенавчання.

У випадковому лісі кінцевий результат класифікації або регресії визначається шляхом голосування або середнього значення дерев. Тобто, кожне дерево дає свій власний результат, а потім ці результати об'єднуються в один.

Random Forest є досить потужним і універсальним алгоритмом, який зазвичай добре показує себе на багатьох типах даних. Він зазвичай працює швидко і дає точні результати без потреби у великих обсягах налаштувань. Однак, як і будь-який алгоритм, він має свої обмеження, такі як складність підготовки даних та потребу в великих кількостях даних для досягнення найкращої продуктивності.

Random Forest використовується в багатьох галузях, включаючи медицину, банківський сектор, маркетинг та багато інших. Він є одним з найбільш популярних алгоритмів машинного навчання, оскільки він має декілька переваг порівняно з іншими алгоритмами, такими як логістична регресія, дерева рішень і градієнтний бустинг.

1.6.3. Особливості методу опорних векторів

Методи опорних векторів (SVM) вважаються одними з найбільш теоретично обґрунтованих та практично найефективніших алгоритмів класифікації в сучасному машинному навчанні. Як методи навчання з учителем, вони вирішують задачу оптимізації знаходження гіперплощини(оптимальної межі поділу), яка максимізує межу між двома класами в просторі ознак.

SVM є ефективним алгоритмом лінійної класифікації та має вагоме теоретичне обґрунтування. Однак на практиці існують набори даних, які не є лінійно роздільними та потребують складнішої функції для розрізнення їхніх класів.

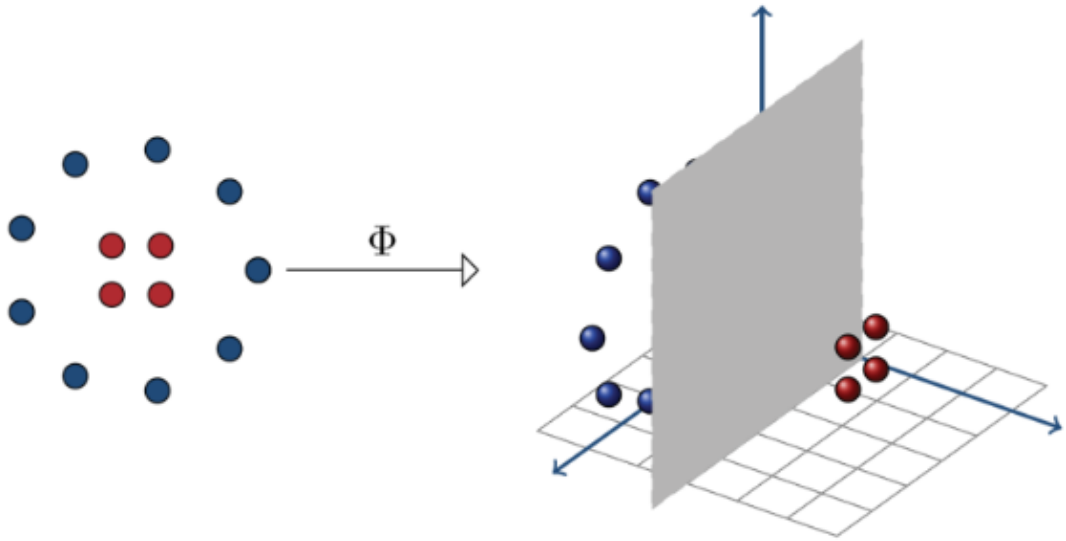


Рис. 1.2. Перехід із 2-вимірного простору у 3-вимірний [58]

На рисунку продемонстровано приклад нелінійного відображення даних із двовимірного простору у тривимірний, яке дозволяє зробити задачу класифікації лінійно роздільною. У початковому просторі ознак червоні об’єкти знаходяться в центрі, тоді як сині утворюють навколо них кільце. У такому випадку неможливо провести лінійну межу, яка б відокремила два класи.

Після застосування нелінійного перетворення Ф(x) дані переносяться у простір більшої розмірності, де стає можливим провести гіперплощину, що чітко розділяє класи. Таким чином, завдяки відображенню у простір вищих вимірів, задача, яка була нелінійною у початкових координатах, перетворюється на лінійно роздільну.

Однак пряме виконання таких перетворень потребує значних обчислювальних витрат, оскільки включає велику кількість обчислень скалярних добутків у просторі високої розмірності. Для оптимізації цього процесу застосовують метод ядра (kernel trick), який дозволяє обчислювати внутрішні добутки у просторі перетворених ознак, працюючи лише з початковими координатами. Це забезпечує ефективність і практичну придатність методу опорних векторів для вирішення задач із нелінійно роздільними даними.[46]

Отож, нелінійне відображення у простір більшої розмірності робить можливим відокремлення даних, які у вихідному просторі не мають лінійної

межі. Застосування ядрових методів у складі SVM дозволяє реалізувати це відображення без надмірних обчислювальних витрат.

1.7. K-кратна перехресна валідація

Замість методу holdout, більш надійним методом налаштування моделі є k-кратна перехресна валідація, де ми повторюємо метод holdout k разів на k підмножинах навчальних даних. Зокрема, ми випадковим чином розділяємо навчальний набір даних на k складок без заміни, де k-1 складок використовується для навчання моделі, а одна складка - для оцінки продуктивності. Потім ми обчислюємо середню продуктивність моделей на основі різних незалежних тестових складок, щоб отримати оцінку продуктивності, яка менш чутлива до підрозділів навчальних даних порівняно з методом holdout.[59]

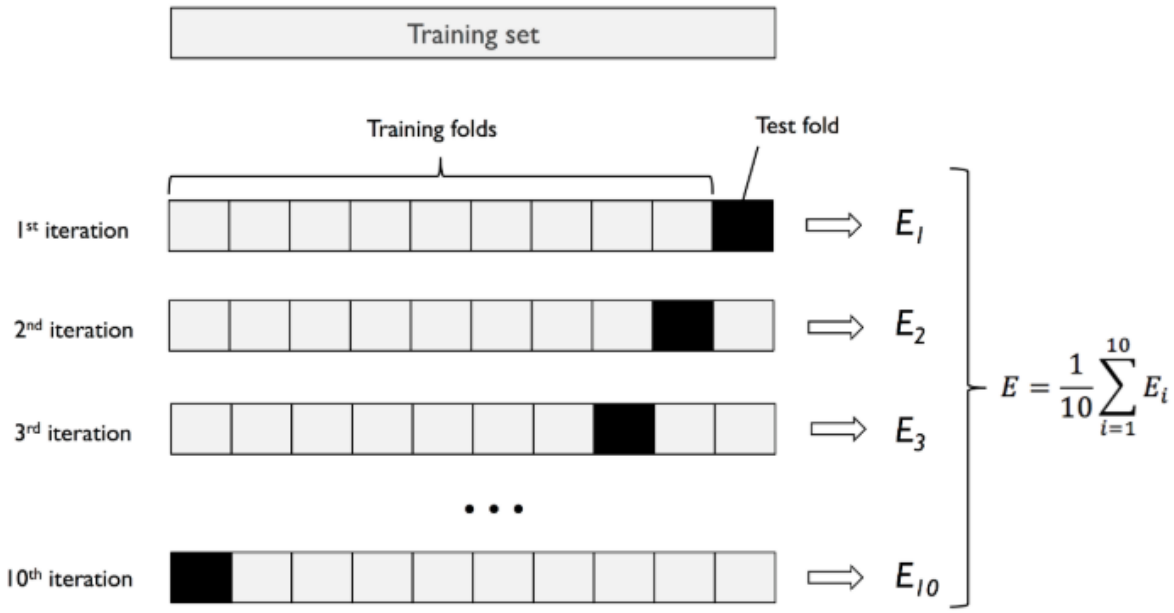


Рис. 1.3. Приклад k-кратної перехресної перевірки з k = 10 [59]

Після того, як ми знайдемо задовільні значення гіперпараметрів, ми можемо перенавчити модель на повному навчальному наборі даних та отримати остаточну оцінку продуктивності, використовуючи незалежний тестовий набір даних. Обґрунтування підгонки моделі до всього навчального набору даних після k-кратної перехресної перевірки полягає в тому, що надання більшої кількості навчальних прикладів алгоритму навчання зазвичай призводить до більш точної та надійної моделі.

Оскільки k-кратна перехресна перевірка є методом повторної вибірки без заміни, перевагою цього підходу є те, що кожен приклад буде використано для навчання та перевірки (як частину тестового згортання) рівно один раз, що дає оцінку продуктивності моделі з меншою дисперсією, ніж метод holdout.

Гарним стандартним значенням k у k-кратній перехресній перевірці є 10. Однак, оскільки ми працюємо з великим набором даних, ми можемо вибрати менше значення k, наприклад, k = 5, і все одно отримати точну оцінку середньої продуктивності моделі, одночасно зменшуючи обчислювальні витрати на переналаштування та оцінку моделі на різних складках.[24]

Як і раніше, стратифікація застосовується, щоб гарантувати, що кожна складка є репрезентативною для пропорцій класів у навчальному наборі даних.

1.8. Метрики оцінювання якості моделей (Accuracy, Precision, Recall, Economic Efficiency)

Оцінювання якості моделей прогнозування кредитних ризиків є критично важливим етапом будь-якого дослідження, адже від цього залежить правильність прийняття рішень і економічна ефективність діяльності банків. У фінансовій сфері найбільш часто застосовуються такі метрики, як Accuracy (точність), Precision (точність позитивних прогнозів), Recall (чутливість) та економічні показники ефективності (Economic Efficiency). Кожна з цих метрик має власну специфіку та статистичне підґрунтя.[25]

Точність відображає загальну частку правильних прогнозів моделі серед усіх спостережень. У випадку класифікації кредитних ризиків вона показує, наскільки модель правильно відокремлює позичальників з високим ризиком дефолту від тих, хто його не має. Формально Accuracy обчислюється як відношення суми правильно класифікованих позитивних та негативних випадків до загальної кількості спостережень:[56]

$$Accuracy=(TP+TN)/(FP+FN+TP+TN) \quad (7)$$

,де

- TP (True Positive) – правильні позитивні прогнози (модель правильно

передбачила дефолт);

- TN (True Negative) – правильні негативні прогнози (модель правильно передбачила відсутність дефолту);
- FP (False Positive) – хибнопозитивні прогнози (модель передбачила дефолт, якого не було);
- FN (False Negative) – хибнонегативні прогнози (модель не передбачила дефолт).

Хоча Accuracy є інтуїтивно зрозумілою метрикою, вона має суттєвий недолік у випадку дисбалансних даних, що характерно для кредитного скорингу: коли частка дефолтів низька, навіть проста модель, що передбачає «без дефолту» для всіх клієнтів, покаже високий показник Accuracy, хоча практична користь такої моделі буде мінімальною.

Precision оцінює, яку частку випадків, передбачених як дефолтні, модель дійсно передбачила правильно. Це показник «чистоти» позитивних прогнозів і критично важливий у фінансовому контексті, коли хибнопозитивні рішення (FP) можуть призвести до зайвого відхилення потенційно вигідних клієнтів. Формула Precision:[56]

$$\text{Precision} = TP/(TP+FP) \quad (8)$$

Високий Precision означає, що серед всіх клієнтів, визначених як високий ризик, більшість дійсно матимуть проблеми з погашенням кредиту, що допомагає зменшити втрати від надмірної консервативності моделі.

Recall показує, яку частку реальних дефолтів модель змогла виявити. Ця метрика важлива для оцінки ризиків непередбачених дефолтів, тобто ситуацій, коли клієнт, який справді є високоризиковим, отримує кредит. Формула Recall:[56]

$$\text{Recall} = TP/(TP+FN) \quad (9)$$

Високий Recall дозволяє мінімізувати кількість непомічених ризикових клієнтів, проте часто він суперечить Precision: підвищення Recall може збільшити FP, тобто точність позитивних прогнозів знижується. Тому на практиці часто використовують F1-score, яка є гармонійним середнім Precision

та Recall:[56]

$$F1=(2\cdot Precision\cdot Recall)/(Precision+Recall) \quad (10)$$

У фінансовій сфері важливо оцінювати не лише статистичну точність моделей, а й їхній економічний вплив. Економічна ефективність моделі визначає, скільки банк заробляє або втрачає внаслідок застосування конкретної моделі прогнозування кредитного ризику. На практиці її розраховують через баланс втрат від непередбачених дефолтів і втрат від відхилених потенційно вигідних клієнтів:

Economic Efficiency= (Виграш від правильно передбачених позичальників−Втрати від FP і FN)/ Загальна кількість клієнтів (11)

Для банків критично важливо знаходити баланс між статистичною точністю та економічною доцільністю. Наприклад, модель із високим Recall, але низьким Precision може зменшити втрати від непередбачених дефолтів, але одночасно збільшити відхилення вигідних позичальників, що негативно позначається на прибутку. Таким чином, економічна ефективність дозволяє інтегрувати метрики точності з фінансовими наслідками рішень моделі.

1.9. Висновки

Проведене дослідження підтверджує, що машинне навчання поступово стає невід'ємною частиною сучасних систем управління кредитними ризиками у фінансовій сфері. Аналіз літератури свідчить про зростаючий інтерес до поєднання традиційних економетричних підходів з алгоритмами штучного інтелекту, що дозволяє підвищити точність прогнозування та адаптуватися до мінливих ринкових умов.

Традиційні статистичні методи, зокрема логістична регресія, зберігають свою актуальність завдяки математичній прозорості та відповідності регуляторним вимогам. Такі моделі забезпечують легкість інтерпретації результатів і стабільність роботи в умовах жорстких нормативних обмежень. Водночас їхні можливості обмежені при роботі зі складними нелінійними

залежностями та великими обсягами різнорідних даних, що характерно для сучасного банківського середовища.

Методи машинного навчання демонструють значні переваги у виявленні прихованих патернів поведінки позичальників та обробці великих масивів інформації. Ансамблеві алгоритми, такі як Random Forest та градієнтний бустинг, показують особливо високу ефективність завдяки здатності автоматично виявляти взаємодії між ознаками та зменшувати ризик перенавчання. Методи опорних векторів виявляються оптимальними для класифікації у високовимірних просторах ознак, особливо при роботі з нелінійно роздільними даними.

Критичним фактором успіху є якість інженерії ознак та правильна обробка дисбалансу класів. У кредитному скорингу частка дефолтів зазвичай становить невеликий відсоток від загальної кількості позичальників, що створює специфічні виклики для навчання моделей. Застосування методів балансування класів, зважених функцій втрат та економічно орієнтованих метрик оцінювання стає необхідною умовою для отримання практично корисних результатів.

Дослідження підкреслює недостатність використання лише статистичних метрик для оцінювання якості моделей у фінансовій сфері. Показники точності, такі як Accuracy, Precision та Recall, мають доповнюватися аналізом економічної ефективності, що враховує реальні витрати від помилкових рішень та прибуток від правильних прогнозів. Такий підхід дозволяє знаходити оптимальний баланс між консервативністю моделі та її здатністю генерувати прибуток.

Впровадження методів машинного навчання у банківську практику стикається з низкою серйозних викликів. Проблема інтерпретованості складних алгоритмів ускладнює їх використання в регульованому середовищі, де кожне рішення повинно бути обґрунтованим та зрозумілим для наглядових органів. Дрейф концепції, коли якість моделей знижується внаслідок зміни економічних умов та поведінки клієнтів, вимагає постійного моніторингу та періодичного переналаштування систем.

Регуляторні обмеження накладають додаткові вимоги щодо прозорості

алгоритмічних рішень та забезпечення справедливості у доступі до фінансових послуг. Банки мають демонструвати, що їхні моделі не призводять до дискримінації окремих груп клієнтів та відповідають принципам етичного використання штучного інтелекту.

Майбутнє прогнозування кредитних ризиків вбачається у розробці гібридних підходів, які поєднують переваги традиційних методів з потужністю сучасних алгоритмів. Перспективними напрямами є створення інтерпретованих моделей машинного навчання з вбудованими обмеженнями монотонності, розробка ансамблів з різними рівнями складності для різних сегментів клієнтів та інтеграція альтернативних джерел даних.

Успішне впровадження потребує системного підходу, що включає поетапне ускладнення архітектури моделей, інвестиції у розвиток відповідної інфраструктури та підготовку кваліфікованих спеціалістів. Критично важливим є налагодження діалогу з регуляторними органами для формування збалансованих стандартів використання штучного інтелекту у фінансовій сфері.

Результати дослідження підтверджують, що машинне навчання має значний потенціал для підвищення ефективності управління кредитними ризиками. Водночас реалізація цього потенціалу вимагає врахування специфіки фінансової галузі та знаходження балансу між технологічними можливостями та практичними обмеженнями. Лише комплексний підхід, що поєднує технічну досконалість з розумінням бізнес-процесів та регуляторних вимог, може забезпечити успішне впровадження інноваційних методів прогнозування у банківську практику.

РОЗДІЛ 2

ПРАКТИЧНА РЕАЛІЗАЦІЯ ПОБУДОВИ МОДЕЛІ ПРОГНОЗУВАННЯ КРЕДИТНИХ РИЗИКІВ

2.1. Загальний опис даних використаних у дослідженні

Набір даних, використаний у цьому дослідженні, – це дані про прострочені платежі клієнтів кредитних карток з репозиторію машинного навчання UCI.[18]

Він складається з 30000 спостережень, які представляють окремих клієнтів кредитних карток. Кожне спостереження має 24 атрибути, що містять інформацію про прострочені платежі, демографічні фактори, кредитні дані, історію платежів та виписки з рахунків клієнтів кредитних карток на Тайвані з квітня 2005 року по вересень 2005 року.

Перша група змінних містить інформацію про особисту інформацію клієнта:

- ID: ID кожного клієнта, категоріальна змінна
- LIMIT_BAL: Сума наданого кредиту в нових тайванських доларах (включає індивідуальний та сімейний/додатковий кредит)
- SEX: Стать, категоріальна змінна (1=чоловік, 2=жінка)
- EDUCATION: рівень освіти, категоріальна змінна (1=аспірантура, 2=університет, 3=середня школа, 4=інше, 5=невідомо, 6=невідомо)
- MARRIAGE: Сімейний стан, категоріальна змінна (1=одружений/одружена, 2=неодружений/незаміжня, 3=інше)
- AGE: Вік у роках, числова змінна

Наступні атрибути містять інформацію про затримку минулого платежу, що стосується певного місяця:

- PAY_0: Статус погашення у вересні 2005 року (-1=сплатити належним чином, 1=затримка платежу на один місяць, 2=затримка

платежу на два місяці, … 8=затримка платежу на вісім місяців, 9=затримка платежу на дев'ять місяців і більше)

- PAY_2: Статус погашення у серпні 2005 року (та ж шкала, що й раніше)
- PAY_3: Статус погашення у липні 2005 року (та ж шкала, що й раніше)
- PAY_4: Статус погашення у червні 2005 року (та ж шкала, що й раніше)
- PAY_5: Статус погашення у травні 2005 року (та ж шкала, що й раніше)
- PAY_6: Статус погашення у квітні 2005 року (та ж шкала, що й раніше)

Інші змінні натомість враховують інформацію, пов'язану з сумою виписки за рахунками (тобто щомісячний звіт, який компанії-емітенти кредитних карток видають власникам кредитних карток у певному місяці):

- BILL_AMT1: Сума виписки за рахунками у вересні 2005 року (новий тайванський долар)
- BILL_AMT2: Сума виписки за рахунками у серпні 2005 року (новий тайванський долар)
- BILL_AMT3: Сума виписки за липень 2005 року (новий тайванський долар)
- BILL_AMT4: Сума виписки за рахунками у червні 2005 року (новий тайванський долар)
- BILL_AMT5: Сума виписки за рахунками у травні 2005 року (новий тайванський долар)
- BILL_AMT6: Сума виписки за рахунками у квітні 2005 року (новий тайванський долар)

Наступні змінні враховують суму попереднього платежу за певний місяць:

- PAY_AMT1: Сума попереднього платежу у вересні 2005 року (NT долари)
- PAY_AMT2: Сума попереднього платежу у серпні 2005 року (NT долари)
- PAY_AMT3: Сума попереднього платежу у липні 2005 року (NT долари)
- PAY_AMT4: Сума попереднього платежу у червні 2005 року (NT долари)
- PAY_AMT5: Сума попереднього платежу у травні 2005 року (NT долари)
- PAY_AMT6: Сума попереднього платежу у квітні 2005 року (NT долари)

Остання змінна є тією, яку потрібно передбачити:

- default.payment.next.month: вказує, чи є власники кредитних карток дефолтерами чи ні (1=так, 0=ні)

Для проведення аналізу було використано кілька ключових бібліотек мови Python, кожна з яких виконує власну функцію у процесі роботи з даними. Scikit-learn є центральним інструментом для побудови моделей машинного навчання, оскільки містить широкий набір алгоритмів для класифікації, регресії, кластеризації та методів оцінювання якості моделей. Важливу роль у підготовці даних відіграють NumPy і pandas. NumPy забезпечує швидкі обчислення з багатовимірними масивами та дозволяє працювати з математичними функціями, тоді як pandas спрощує обробку табличних структур завдяки DataFrame, роблячи трансформацію, очищення та фільтрацію даних більш зручною.[16]

Окрему увагу приділено бібліотеці imbalanced-learn (imblearn), яка застосовується тоді, коли дані мають дисбаланс між класами. Вона надає методи збалансування вибірок, зокрема oversampling, undersampling і комбіновані підходи, що особливо корисно у задачах класифікації. Для представлення результатів та наочного аналізу використовуються інструменти візуалізації.

Matplotlib є базовою бібліотекою, яка дозволяє створювати різноманітні графіки та діаграми з високим рівнем гнучкості у налаштуванні відображення. На її основі розвинулась seaborn, яка розширює можливості matplotlib, пропонуючи більш зручний синтаксис і привабливе оформлення графіків. Вона особливо добре підходить для статистичної візуалізації та тісно інтегрується з pandas, що робить її незамінною при швидкому дослідженні даних. (Лістинг 1)[17][21]

Щоб зрозуміти, як виглядають дані, в таблиці нижче наведено деякі спостереження. Цільовий стовпець default.payment.next.month перейменовано на DEFAULT, щоб скоротити, а стовпець PAY_0 перейменовано на PAY_1. (Лістинг 2)

Out[3]:

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_
1	20000	2	2	1	24	2	2	-1	-1	-2	...	0	0	0	
2	120000	2	2	2	26	-1	2	0	0	0	...	3272	3455	3261	
3	90000	2	2	2	34	0	0	0	0	0	...	14331	14948	15549	
4	50000	2	2	1	37	0	0	0	0	0	...	28314	28959	29547	
5	50000	1	2	1	57	-1	0	-1	0	0	...	20940	19146	19131	

5 rows × 24 columns

Out[3]:

PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	DEFAULT
-1	-2	...	0	0	0	0	689	0	0	0	0	1
0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
0	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0

Рис. 2.1. Огляд наявних даних

Інформація з набору даних, наведеного нижче, показує, що для жодного зі зразків немає відсутніх ознак. (Лістинг 3)

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30000 entries, 1 to 30000
Data columns (total 24 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   LIMIT_BAL  30000 non-null  int64
 1   SEX        30000 non-null  int64
 2   EDUCATION  30000 non-null  int64
 3   MARRIAGE   30000 non-null  int64
 4   AGE        30000 non-null  int64
 5   PAY_1      30000 non-null  int64
 6   PAY_2      30000 non-null  int64
 7   PAY_3      30000 non-null  int64
 8   PAY_4      30000 non-null  int64
 9   PAY_5      30000 non-null  int64
 10  PAY_6      30000 non-null  int64
 11  BILL_AMT1  30000 non-null  int64
 12  BILL_AMT2  30000 non-null  int64
 13  BILL_AMT3  30000 non-null  int64
 14  BILL_AMT4  30000 non-null  int64
 15  BILL_AMT5  30000 non-null  int64
 16  BILL_AMT6  30000 non-null  int64
 17  PAY_AMT1   30000 non-null  int64
 18  PAY_AMT2   30000 non-null  int64
 19  PAY_AMT3   30000 non-null  int64
 20  PAY_AMT4   30000 non-null  int64
 21  PAY_AMT5   30000 non-null  int64
 22  PAY_AMT6   30000 non-null  int64
 23  DEFAULT    30000 non-null  int64
```

Рис. 2.2. Огляд даних

2.1.1. Очистка даних

Нижче наведено опис характеристик персональної інформації клієнта. Атрибути LIMIT_BAL, SEX та AGE, здається, узгоджуються з описом, тоді як EDUCATION та MARRIAGE мають деякі недокументовані категорії. EDUCATION коливається від 0 до 6, тоді як MARRIAGE починається з категорії 0. (Лістинг 4)

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	167484.322667	1.603733	1.853133	1.551867	35.485500
std	129747.661567	0.489129	0.790349	0.521970	9.217904
min	10000.000000	1.000000	0.000000	0.000000	21.000000
25%	50000.000000	1.000000	1.000000	1.000000	28.000000
50%	140000.000000	2.000000	2.000000	2.000000	34.000000
75%	240000.000000	2.000000	2.000000	2.000000	41.000000
max	1000000.000000	2.000000	6.000000	3.000000	79.000000

Рис. 2.3. Огляд описової інформації обраних факторів

```
0       14
1    10585
2    14030
3     4917
4      123
5      280
6       51
Name: EDUCATION, dtype: int64
```

Рис. 2.4. Огляд розподілу даних у факторі рівня освіти

```
0       54
1    13659
2    15964
3      323
Name: MARRIAGE, dtype: int64
```

Рис. 2.5. Огляд розподілу даних у факторі сімейного стану

Наявність помилок у наборі даних можна вирішити, виправивши неправильний атрибут або видаливши рядки, пов'язані з помилкою. Ми могли б застосувати консервативний підхід і згрупувати недокументовані категорії в інші, але оскільки аномальних записів відносно мало (399, 1,33% від загальної кількості), ми вирішили їх виключити. (Лістинг 5)

```
Dataset size before:    30000
Dataset size after:     29601
```

Рис. 2.6. Порівняння розміру датасету до та після видалення непотрібних

2.1.2. Історія попередніх оплат

Нижче ми наводимо опис історії попередніх платежів. Значення в стовпцях PAY_n коливаються від -2 (недокументовано) до 8, тому ми вважаємо, що їх потрібно масштабувати відповідно до офіційного опису. (Лістинг 6-7)

	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6
count	29601.000000	29601.000000	29601.000000	29601.000000	29601.000000	29601.000000
mean	-0.014932	-0.131313	-0.163440	-0.218303	-0.263978	-0.287558
std	1.124503	1.199642	1.199793	1.172220	1.136217	1.152206
min	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000
25%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000

Рис. 2.7. Огляд опису даних про минулі платежі

2.1.3. Сума рахунку-фактури та попередній платіж

Нижче виводиться опис особливостей, пов'язаних із сумою виписки з рахунку(Лістинг 8) та сумою попереднього платежу(Лістинг 9). Жодних аномалій виявлено не було.

	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6
count	29601.000000	29601.000000	2.960100e+04	29601.000000	29601.000000	29601.000000
mean	50957.432012	48942.189554	4.680320e+04	43122.554204	40235.545184	38858.449816
std	73370.242404	70923.985151	6.912389e+04	64196.383913	60699.344884	59519.893043
min	-165580.000000	-69777.000000	-1.572640e+05	-170000.000000	-81334.000000	-339603.000000
25%	3528.000000	2970.000000	2.652000e+03	2329.000000	1780.000000	1278.000000
50%	22259.000000	21050.000000	2.003500e+04	19005.000000	18091.000000	17118.000000
75%	66623.000000	63497.000000	5.983000e+04	54271.000000	50072.000000	49121.000000
max	964511.000000	983931.000000	1.664089e+06	891586.000000	927171.000000	961664.000000

Рис. 2.8. Огляд інформації про суми виписки

	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6
count	29601.000000	2.960100e+04	29601.000000	29601.000000	29601.000000	29601.000000
mean	5649.560319	5.894788e+03	5198.415898	4828.659268	4795.032735	5181.326374
std	16568.264941	2.308919e+04	17580.914806	15711.057992	15244.217154	17657.260739
min	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000
25%	1000.000000	8.250000e+02	390.000000	298.000000	259.000000	138.000000
50%	2100.000000	2.007000e+03	1800.000000	1500.000000	1500.000000	1500.000000
75%	5005.000000	5.000000e+03	4500.000000	4014.000000	4042.000000	4000.000000
max	873552.000000	1.684259e+06	896040.000000	621000.000000	426529.000000	528666.000000

Рис.2.9. Огляд інформації про суму попереднього платежу

2.2. Дослідницький аналіз даних (EDA)

Головна мета полягає в тому, щоб розрізнити клієнтів, у яких прогнозується дефолт за кредитною карткою наступного місяця, відповідно до стовпця default.payment.next.month, який має значення «0» для клієнтів, що не є дефолтами, та «1» для тих, хто є дефолтами. Таким чином, це задача бінарної класифікації на відносно незбалансованому наборі даних, як показано на наступному рисунку. (Лістинг 11)

Рис. 2.10. Розподіл класів

По-перше, ми бачимо, що набір даних, з яким ми маємо справу, досить незбалансований, і містить лише 6636 невиконачих зобов'язань (22,1% від загальної кількості). Це досить актуальна проблема, яку потрібно вирішити. Якщо нею нехтувати, моделі класифікації, як правило, зосереджуватимуться на класі більшості та ігноруватимуть клас меншості.

Для ознаки LIMIT_BAL ми створюємо карту щільності відповідно до типу дефолту, і результат показано нижче ліворуч. Здається, що коли задана сума кредиту приблизно менша за 140000, ймовірність дефолту більша, ніж ймовірність відсутності дефолту. Це свідчить про те, що ті, хто має нижчі кредитні ліміти, частіше опиняються в дефолті, тоді як ті, хто має вищі кредитні ліміти, менш схильні до дефолту. Це спостереження має сенс, оскільки вищий кредитний ліміт надається людям з вищою кредитоспроможністю, які мають меншу ймовірність дефолту. (Лістинг 12)

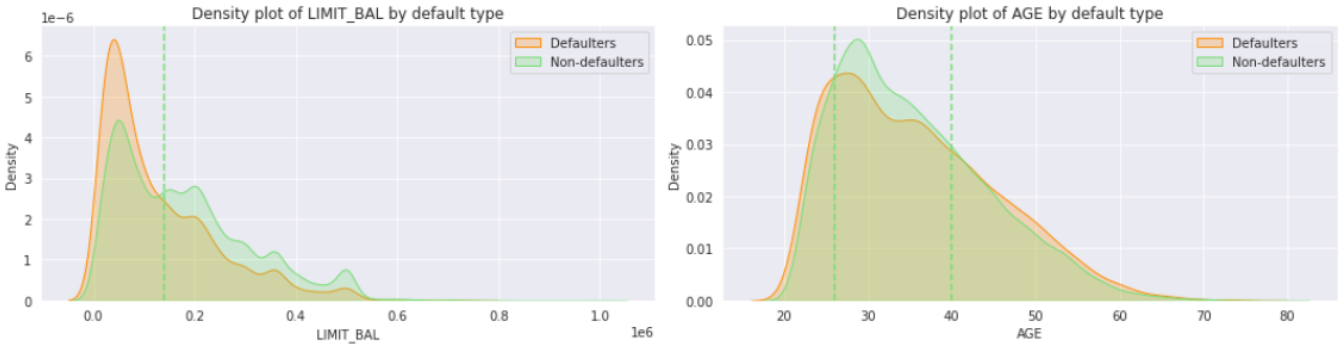


Рис. 2.11. Карта щільності для 2 вибраних факторів

Для ознаки віку ми проводимо аналогічний візуальний аналіз, як показано вище праворуч. Ймовірність відсутності дефолту у віці приблизно від 25 до 40 років вища, що свідчить про те, що споживачі цієї вікової групи більш здатні погашати кредити за кредитними картками. Це може бути пов'язано з тим, що їхня робота та сім'я, як правило, стабільні без надмірного тиску.

Ознаки СТАТЬ, ОСВІТА та ШЛЮБ відображаються відповідно до цільової змінної за допомогою гістограм, як показано нижче. Незалежно від того, чоловіки це чи жінки, частка неплатників відповідає загальній ситуації, і те саме можна сказати про категорії двох інших ознак. (Лістинг 13)

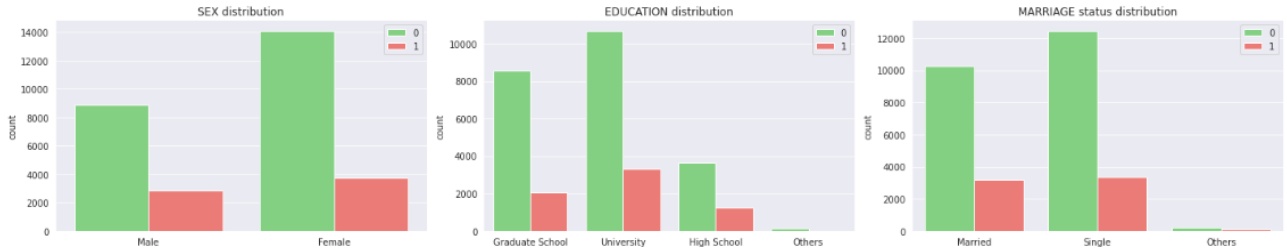


Рис. 2.12. Огляд розподілу дефолтних та не дефолтних клієнтів за факторами

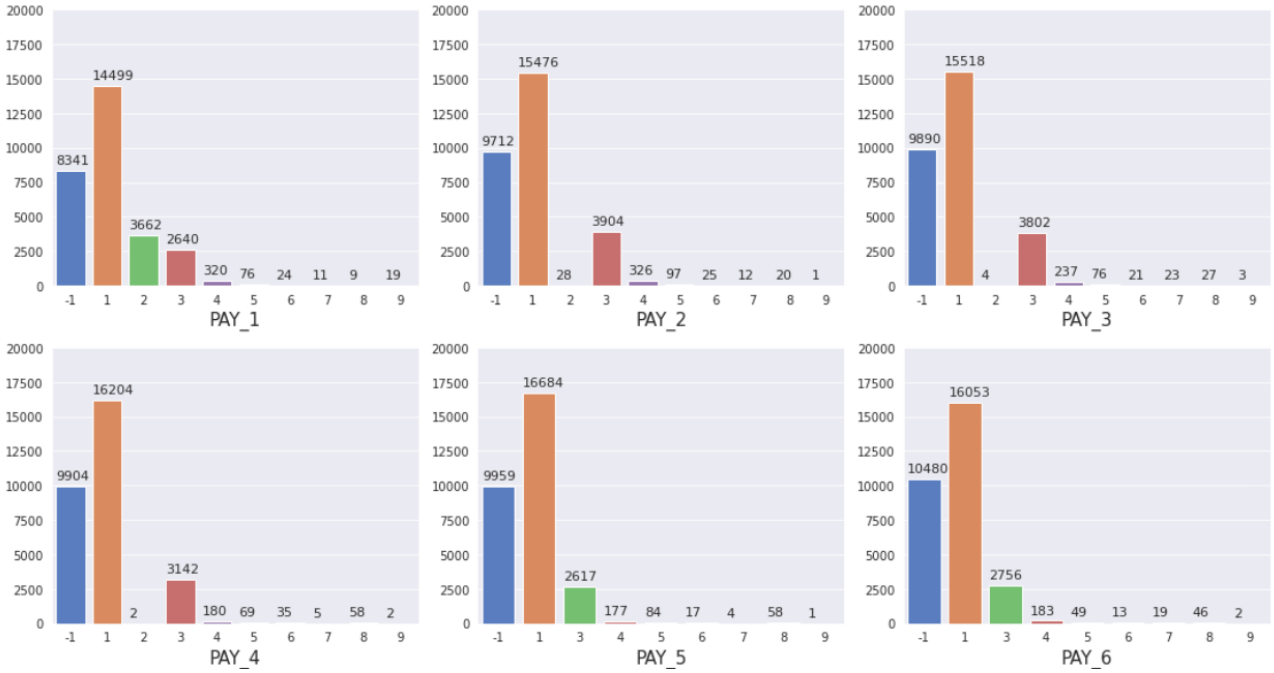


Рис. 2.13. Розподіл спостережень у факторах оплати (Лістинг 14)

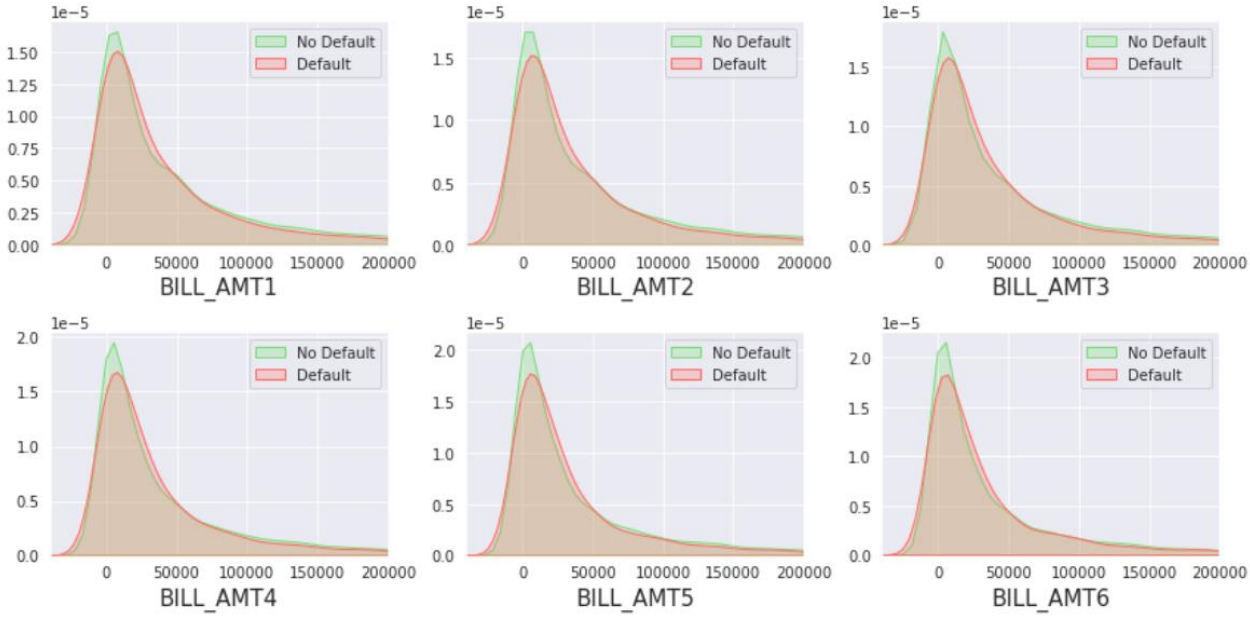


Рис. 2.14. Огляд розподілу по сумах виписки(Лістинг 15)

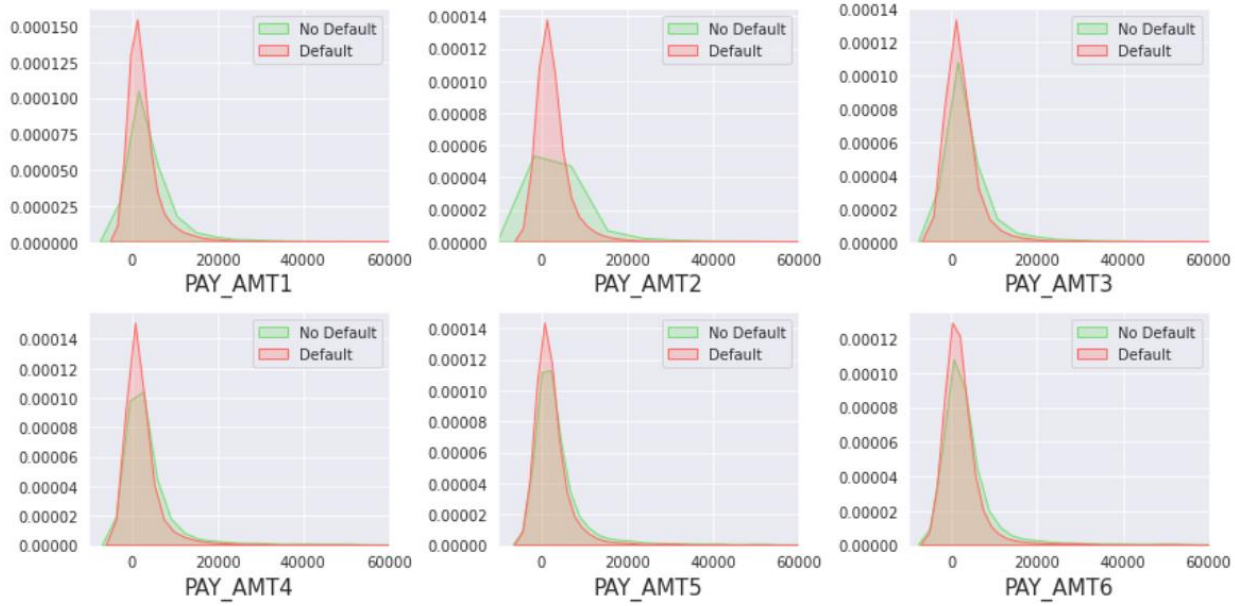


Рис. 2.15. Огляд розподілу по сумах попереднього платежу(Лістинг 16)

Набір ознак статусу платежу показано нижче за допомогою блокових діаграм. Видно, що клієнти, які затримують платіж на один місяць або менше, мають менше прострочень за кредитними картками. Зокрема, статус погашення у вересні, тобто PAY_1, має більшу дискримінаційну силу, ніж статус погашення в інші місяці. (Лістинг 17)

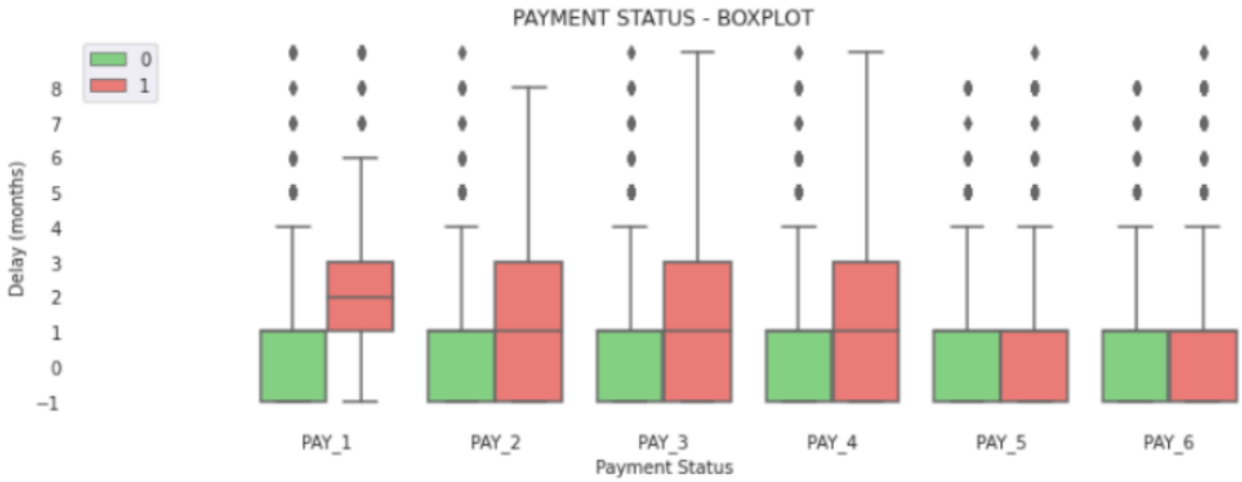


Рис. 2.17. Графік ознак статусу платежу

2.2.1. Кореляція між факторами

Розглянутий набір даних має багато атрибутів, і наявність сильно корельованих ознак може призвести до зниження продуктивності деяких

алгоритмів класифікації. Дійсно, існують деякі методи, які припускають незалежність предикторів і отримують вигоду від зменшення розмірності.

Коефіцієнт кореляції Пірсона p допомагає нам з'ясувати зв'язок між двома змінними, вимірюючи силу зв'язку між ними. Він розраховується як коваріація між двома ознаками X та Y (чисельник), поділена на добуток їх стандартних відхилень (знаменник):

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} \tag{12}$$

Значення коефіцієнта кореляції Пірсона можуть варіюватись від -1 до +1.

- +1 означає, що вони сильно корелюють,
- 0 означає відсутність кореляції,
- -1 означає, що існує негативна кореляція (обернена пропорційність).

Коефіцієнт кореляції Пірсона здатний відображати лише лінійні тенденції, тому він може призвести до значення 0 для сильно нелінійно корельованих змінних (наприклад, квадратичний тренд).

Нижче наведено лише нижню половину матриці кореляції між числовими ознаками (діагональ виключена). (Лістинг 18)

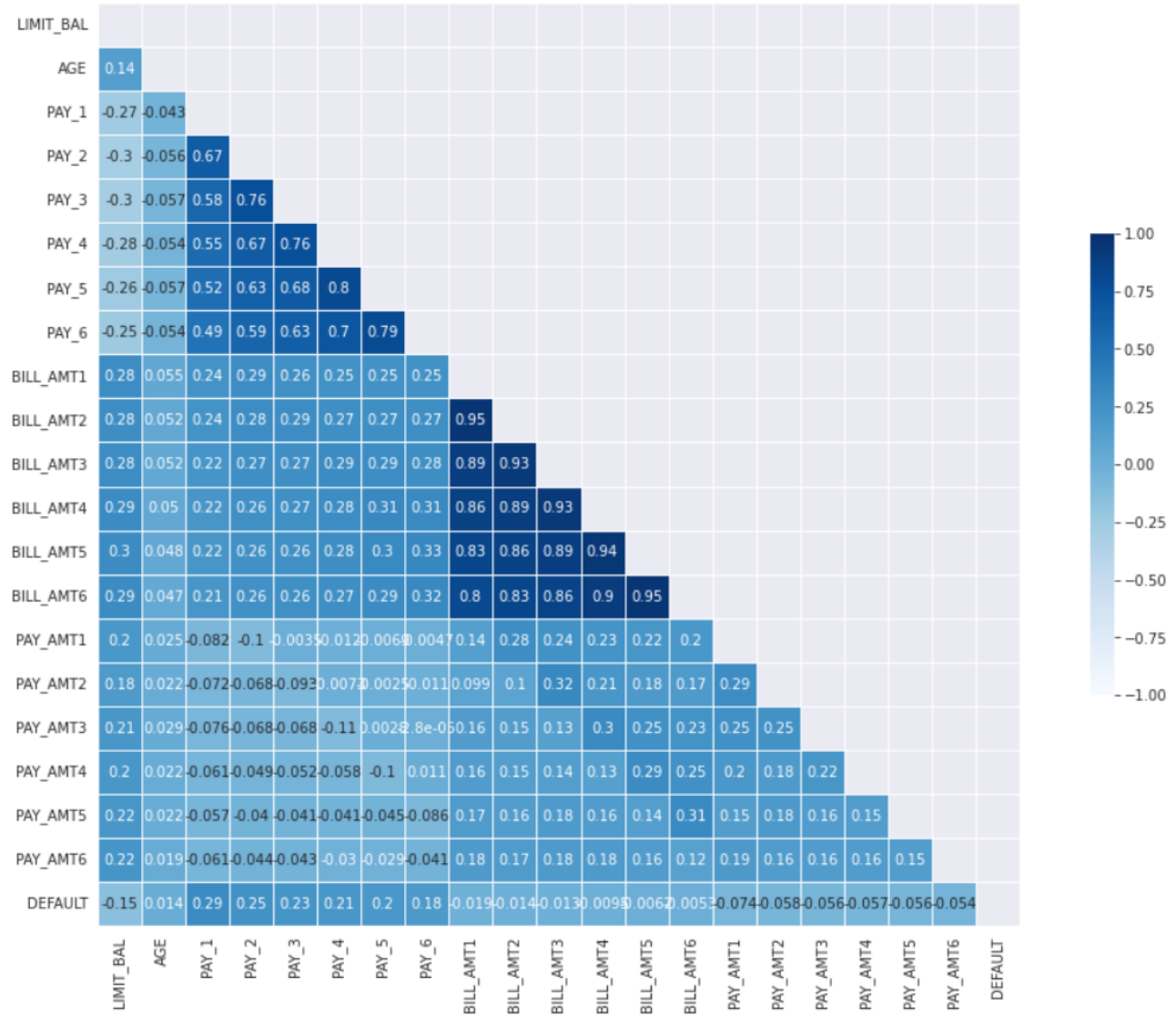


Рис. 2.18. Кореляційна матриця

Ми зазначаємо, що між ознаками BILL_AMTn існує сильна позитивна кореляція, що може свідчити про надмірність інформації. Отже, ми наводимо діаграми розсіювання цих змінних, щоб підкреслити їхню взаємодію. Матриця нижче містить для кожної комбінації змінних невелику діаграму розсіювання, яка відображає розподіл точок даних між двома розглянутими змінними, розділеними двома класами. Лінійний тренд підтверджує те, що припускав коефіцієнт Пірсона, а саме, що ознаки BILL_AMTn кодують досить схожу інформацію. (Лістинг 19)

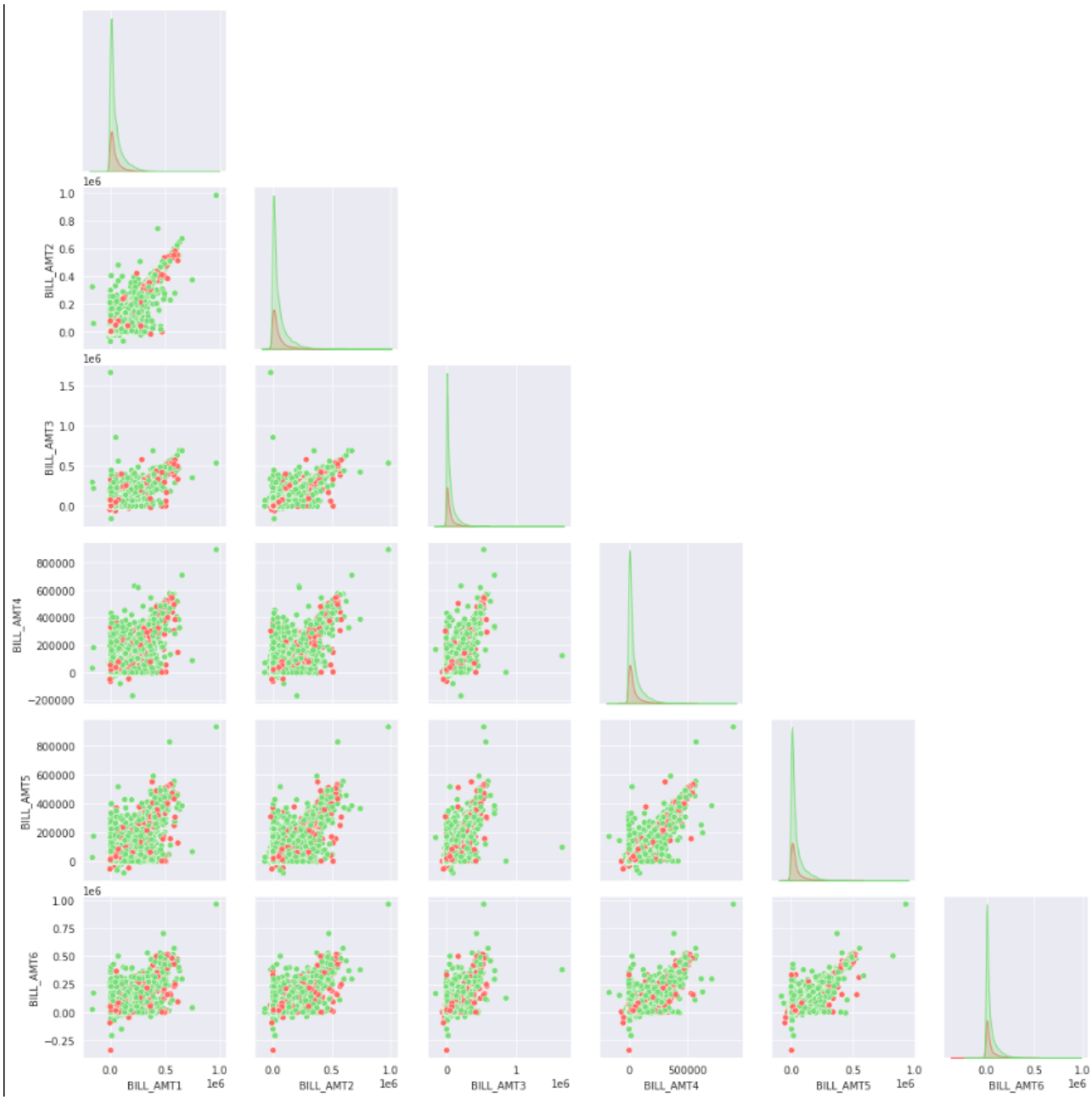


Рис. 2.19. Діаграми розсіювання для змінних групи BILL_AMTn

2.2.2. Перевірка на нормальність розподілу

Параметричні статистичні методи припускають, що дані мають відомий та специфічний розподіл, часто гаусівський розподіл. Якщо ми застосовуємо такі методи до даних з іншого розподілу, наші результати можуть бути оманливими або відверто неправильними. Щоб перевірити, чи є наші дані гаусівськими, ми використовуємо графічний метод, який називається квантиль-квантильною (QQ) діаграмою, що дозволяє нам дати якісну оцінку. На QQ діаграмі квантилі незалежної змінної відображаються відносно очікуваних квантилів нормального

розподілу. Якщо змінна має нормальний розподіл, точки на QQ діаграмі повинні розташовуватися вздовж діагоналі 45 градусів.

Наступні діаграми показують, що немає жодних доказів того, що числові ознаки розподілені нормально. (Лістинг 20)(Рис.2.20)

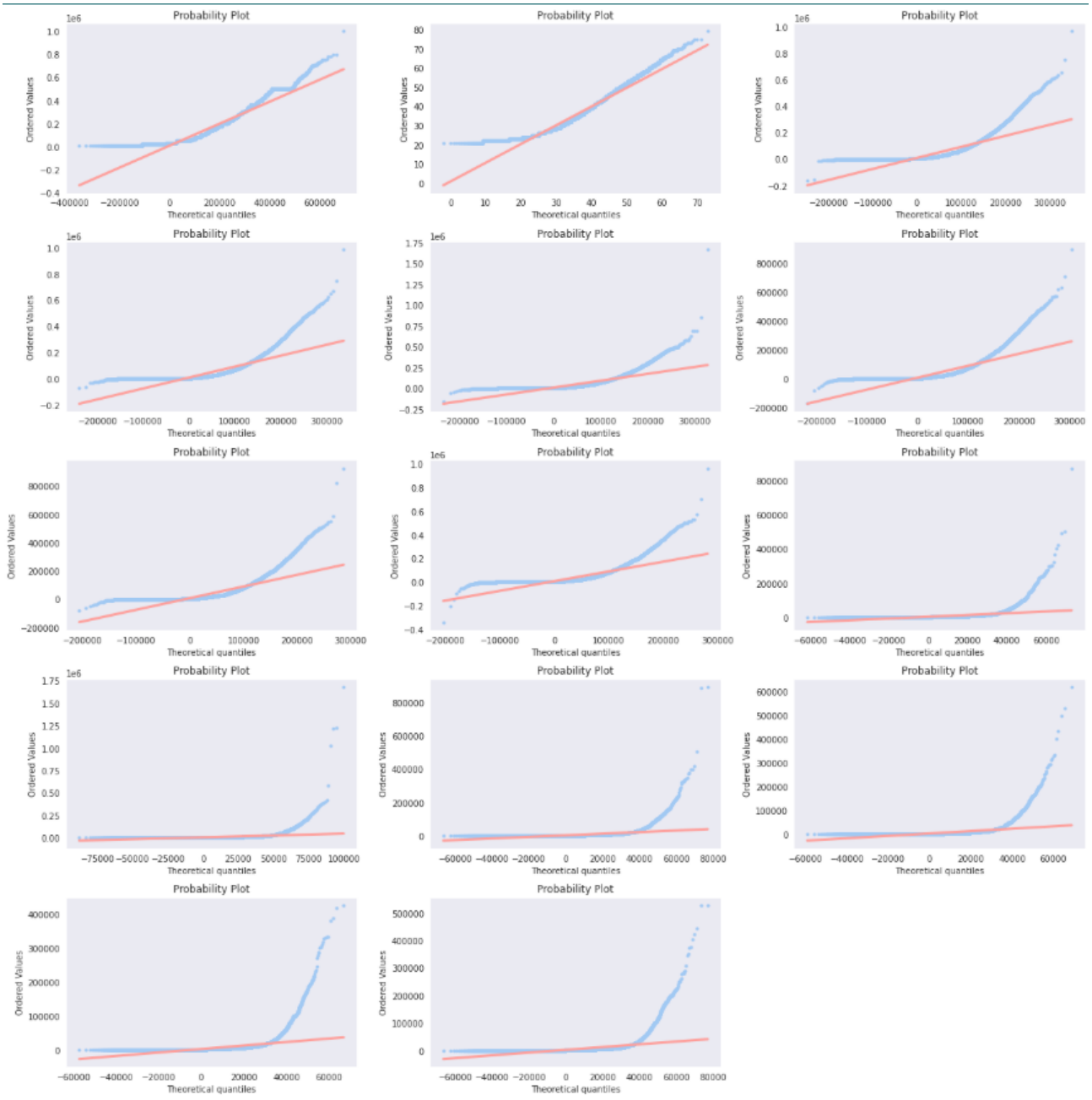


Рис. 2.20. Графік перевірки на нормальність розподілу

2.3. Попередня підготовка даних для подальшого моделювання

Якість даних та обсяг корисної інформації, що міститься в них, є ключовими факторами, що визначають, наскільки добре може навчатися алгоритм машинного навчання. Тому вкрай важливо переконатися, що ми

перевіряємо та попередньо обробляємо наш набір даних, перш ніж передати його алгоритму навчання.

Категоричні ознаки освіти, статі та сімейного стану вже закодовані цілими числами та можуть бути передані алгоритму машинного навчання. Однак це номінальні ознаки, для яких було б неоптимальним припускати впорядкування. Одноразове кодування дозволяє нам видалити будь-які порядкові зв'язки, які були б безглуздими між цими категоріальними змінними. Ідея цього підходу полягає у створенні нової фіктивної ознаки для кожного унікального значення у стовпці номінальної ознаки. Двійкові значення потім можна використовувати для позначення конкретного класу прикладу.

Хоча Scikit-Learn надає методи для автоматичного виконання одноразового кодування, було вирішено виконувати відображення ознак вручну, оскільки їх небагато. Таким чином ми пом'якшуємо проблему мультиколінеарності, яка виникає, коли є висококорельовані ознаки. Таким чином, ми створюємо такі логічні стовпці та видаляємо старі: освіта, стать та сімейний стан.

- MALE: 1 = чоловік; 0 = жінка.
- MARRIED: 1 = одружений/заміжня сімейний стан; 0 = інакше.
- GRAD_SCHOOL: 1 = рівень освіти аспірантури; 0 = інакше.
- UNIVERSITY: 1 = рівень освіти університету; 0 = інакше.
- HIGH_SCHOOL: 1 = рівень освіти старшої школи; 0 = інакше.

Варто зауважити, що жодної важливої інформації не втрачається при видаленні стовпця характеристики; наприклад, було видалено стовпець EDUCATION=4 (інші), але інформація про характеристику все ще зберігається, оскільки якщо ми спостерігаємо GRAD_SCHOOL=0, UNIVERSITY=0 та HIGH_SCHOOL=0, це означає, що спостереження має бути OTHERS. (Лістинг 21)

2.3.1. Розділення датасету

Щоб оцінити здатність до узагальнення алгоритмів машинного навчання, які ми будемо використовувати, ми розділимо набір даних на навчальний набір та тестовий набір у співвідношенні 3:1. Зокрема, ми застосовуємо стратифікацію до цільової змінної DEFAULT, щоб забезпечити приблизно збереження відносних частот класів як у навчальному, так і в тестовому розподілі. Ми встановлюємо random_state = 24, щоб зробити експерименти повторюваними. (Лістинг 22)

```
Training set shape:  (22200, 25)
- Defaulters:    4954
- Non-defaulters: 17246
Test set shape:  (7401, 25)
- Defaulters:    1651
- Non-defaulters: 5750
```

Рис. 2.21. Результат розподілу датасету на тестову та тренувальну вибірки

2.3.2. Скейлінг факторів

Більшість алгоритмів машинного навчання та оптимізації поводять себе набагато краще, якщо числові ознаки знаходяться в одному масштабі. Дерева рішень та випадкові ліси – це два з небагатьох алгоритмів машинного навчання, де немає потреби турбуватися про масштабування ознак, оскільки вони є масштабно-інваріантними.

Існує два поширені підходи до об'єднання різних ознак в один масштаб: нормалізація та стандартизація. Нормалізація (Лістинг 23) стосується перемасштабування ознак до діапазону [0, 1], що є окремим випадком міні-максимального масштабування:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{13}$$

Тут X – це окремий приклад, Xmin – найменше значення у стовпці ознак, а Xmax – найбільше значення.

Використовуючи стандартизацію(Лістинг 24), ми центруємо стовпці ознак на середньому значенні μ = 0 зі стандартним відхиленням σ = 1, щоб стовпці ознак мали ті ж параметри, що й стандартний нормальний розподіл, що полегшує визначення ваг.

$$Z = \frac{X - \mu}{\sigma} \tag{14}$$

Щоб забезпечити справедливий контрольний показник, ми підбираємо статистику (мінімум-максимум для нормалізації, середнє значення та стандартне відхилення для стандартизації) на навчальному наборі, а потім трансформуємо навчальний та тестовий набори відповідно до них.

Ми порівнюємо два підходи, показуючи діаграми типу «коробка та вуси» відповідних масштабованих даних. Варто звернути увагу на велику кількість викидів.

Хоча нормалізація(Лістинг 23) пригнічує вплив викидів, стандартизація(Лістинг 24) зберігає інформацію про них. Оскільки метод, який ми незабаром застосуємо, чутливий до наявності викидів, ми вирішуємо продовжити наше дослідження з нормалізованими даними.(Рис.2.22)

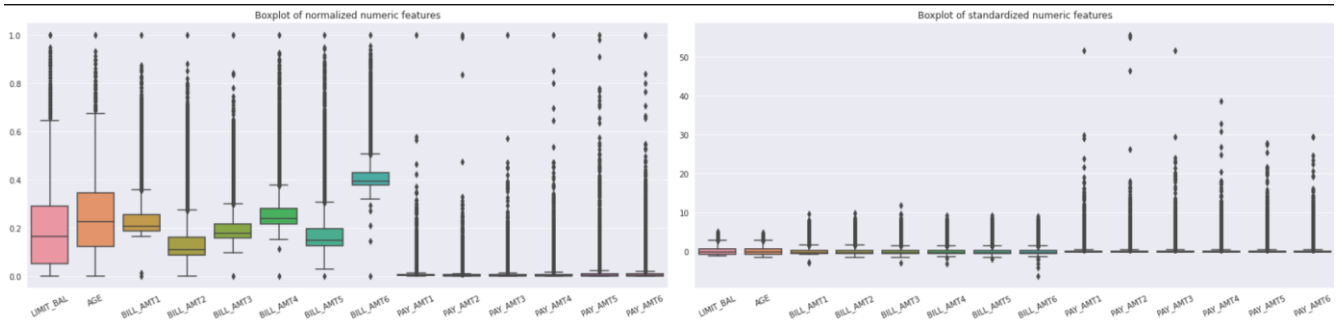


Рис. 2.22. Результати скейлінгу факторів

2.3.3. Зменшення розмірності

Багато алгоритмів отримують вигоду від видалення сильно корельованих ознак і загалом від зменшення розмірності даних. Наявність багатьох ознак призводить до надмірно складних моделей, які занадто точно відповідають параметрам щодо конкретних спостережень у навчальному наборі, але погано

узагальнюються на нові дані. Ми кажемо, що ці моделі мають високу дисперсію та перенавчають навчальний набір.

Проблема перенавчання стає ще серйознішою у випадку алгоритму K-середніх через так зване «прокляття розмірності». Прокляття розмірності описує явище, коли простір ознак стає все більш розрідженим для зростаючої кількості вимірів навчального набору даних фіксованого розміру. Можна вважати навіть найближчих сусідів занадто далекими у багатовимірному просторі, щоб дати хорошу оцінку.

Простіші моделі можна отримати, вимагаючи менше параметрів для підгонки даних. Окрім меншої вартості обчислень та кращої інтерпретації, ми також отримується краща здатність до узагальнення. Процес зменшення розмірності можна здійснити за допомогою вибору ознак або вилучення ознак. За допомогою вибору ознак вибирається підмножина вихідних ознак, тоді як при вилученні ознак ми отримуємо інформацію з набору ознак для побудови нового підпростору ознак. У контексті зменшення розмірності вилучення ознак можна розуміти як підхід до стиснення даних з метою збереження більшої частини відповідної інформації.

Тепер можна вручну відкинути висококорельовані ознаки, які було знайдено раніше. Однак ми вирішили зберегти їх для виконання аналізу головних компонентів – методу вилучення ознак.

2.3.4. Метод головних компонент

Метод головних компонентів – це метод лінійного перетворення без учителя, який допомагає нам виявляти закономірності в даних на основі кореляції між ознаками. Коротко кажучи, метод має на меті знайти напрямки максимальної дисперсії у високовимірних даних та проектувати ці дані на новий підпростір з рівними або меншими вимірами, ніж початковий. Ортогональні осі (головні компоненти) нового підпростору можна інтерпретувати як напрямки максимальної дисперсії, враховуючи обмеження, що осі нових ознак ортогональні одна одній, як показано на наступному рисунку.[60]

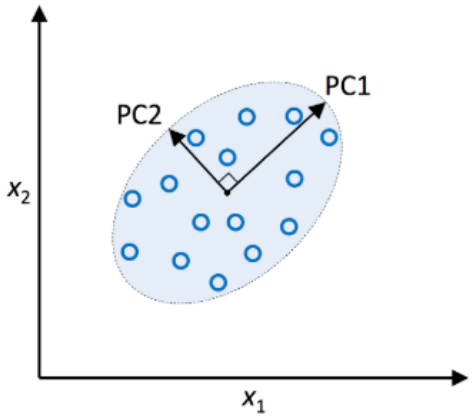


Рис. 2.23. Головні компоненти у двовимірному прикладі [60]

На попередньому рисунку x1 та x2 – це вихідні осі ознак, а PC1 та PC2 – головні компоненти.

Виведення даних методом аналізу основних компонент

Нехай x1,.....,xm – це m векторів вибірки в R^d з нашого набору даних. Проблему зменшення розмірності часто можна розглядати як вбудовування вихідних даних з R^d у новий простір R^n, розмірність якого значно менша (n<d).

PCA виконує зменшення розмірності як простий автоенкодер, який здійснює стиснення та відновлення за допомогою лінійних перетворень. Ідея полягає в тому, щоб знайти матрицю W є R^(n, d), яка індукує відображення.

$$\mathbf{x} \mapsto W\mathbf{x}, \qquad \mathbf{x} \in \mathbb{R}^d,\ W\mathbf{x} \in \mathbb{R}^n\ (n < d) \tag{15}$$

та другу матрицю U є R^(d, n), яка здатна відновити кожен вихідний вектор x з його стиснутої версії y = Wx. Загалом, точне відновлення неможливе, і PCA прагне знайти лінійні перетворення, для яких різниці між відновленими векторами x~ = Uy та вихідними векторами x є мінімальними в сенсі методу найменших квадратів:

$$\underset{W \in \mathbb{R}^{n,d},\, U \in \mathbb{R}^{d,n}}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{x}_i - UW\mathbf{x}_i\|_2^2 \tag{16}$$

Оптимальне рішення (U, W) цієї задачі має такий вигляд:

- стовпці U ортонормовані: U T U = In є R^n;

- W = U^T.

Отже, можна записати:

$$\underset{U \in \mathbb{R}^{d,n}: U^\top U = \mathbb{I}_n}{\operatorname{argmin}} \sum_{i=1}^{m} \|\mathbf{x}_i - UU^\top \mathbf{x}_i\|_2^2 \tag{17}$$

І може бути доведено, що:

$$\|\mathbf{x} - UU^\top \mathbf{x}\|^2 = \|\mathbf{x}\|^2 - \operatorname{tr}\left(U^\top \mathbf{x}\mathbf{x}^\top U\right) \tag{18}$$

де слід tr матриці є сумою її діагональних елементів, тобто лінійним оператором. Це дозволяє нам переписати рівняння як наступну задачу максимізації:

$$\underset{U \in \mathbb{R}^{d,n}: U^\top U = \mathbb{I}_n}{\operatorname{argmax}} \operatorname{tr}\left(U^\top \sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^\top U\right) \tag{19}$$

Тепер ми можемо визначити матрицю розсіювання $A = \sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^\top$, і, оскільки ця матриця є симетричною, її можна записати, використовуючи її спектральний розклад A = VDV^T, де D є діагональною, а V^T V = VV^T = In. Тут елементи на діагоналі D є власними значеннями A, а стовпці V є відповідними власними векторами. Зокрема, ми можемо сміливо припустити, що діагональні елементи D відсортовані за найбільшим, і всі вони є додатними, оскільки A є напіввизначено додатною:

$$D_{1,1} \geq D_{2,2} \geq \ldots \geq D_{d,d} \geq 0 \tag{20}$$

Виходячи з цих передумов, можна стверджувати, що розв'язком задачі оптимізації є матриця U, стовпці якої u1,...,un є n власними векторами матриці A, що відповідають найбільшим n власним значенням, тоді як W = U^T.

Варто звернути увагу, що $\hat{\Sigma} = \frac{1}{m-1} A$ – це емпірична коваріаційна матриця, і її можна ввести замість A, оскільки це подібні матриці та мають

однаковий слід. Тому PCA можна інтерпретувати як метод знаходження напрямків максимальної дисперсії, які задаються головними компонентами, тобто власними векторами, що відповідають найбільшим власним значенням $\hat{\Sigma}$.

Оскільки ми хочемо зменшити розмірність нашого набору даних, стиснувши його на новий підпростор ознак, ми вибираємо лише підмножину власних векторів (головних компонентів), яка містить найбільшу кількість інформації (дисперсії). Власні значення визначають величину власних векторів, тому, щоб вибрати n найбільш інформативних власних векторів, ми можемо побудувати графіки пояснених дисперсійних коефіцієнтів власних значень. (Лістинг 26)

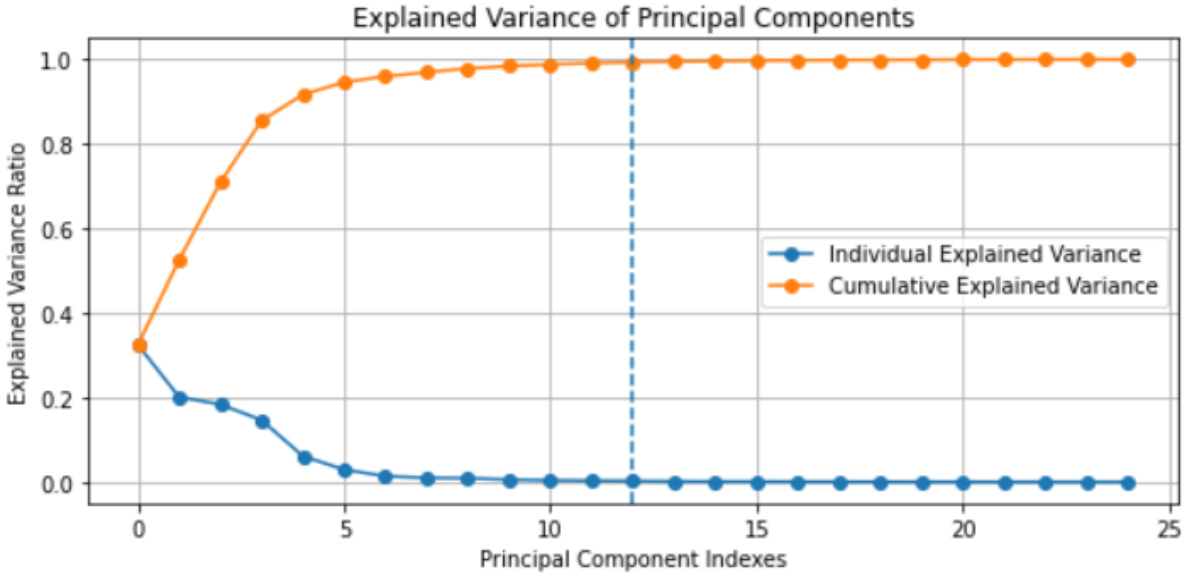


Рис. 2.24. Пояснення вибірки факторами

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15
Cumulative Explained Variance	32.46%	52.56%	70.93%	85.57%	91.69%	94.63%	96.01%	96.98%	97.88%	98.44%	98.84%	99.14%	99.36%	99.51%	99.61%

Рис. 2.25. Об’єм вибірки пояснюваний факторами (Лістинг 27)

Результати, показані на графіку вище(Рис.2.24), вражають, оскільки за допомогою перших 5 головних компонентів можна охопити понад 90% загальної дисперсії. Однак для нашого дослідження ми вирішили розглянути перші 12 головних компонентів. Таким чином, пояснюється понад 99% дисперсії, тоді як кількість ознак зменшилася вдвічі.

Алгоритм PCA було підігнано лише до навчального набору, щоб уникнути вбудовування інформації з тестового набору та зберегти її невідомою. Після цього ми проектуємо тестові дані на підпростір, згенерований першими 12 головними компонентами, знайденими шляхом обчислення коваріаційної матриці на навчальному наборі. (Лістинг 28)

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12
0	-0.541856	-0.551393	-0.412480	0.122101	0.238448	0.715080	-0.118730	-0.217807	0.089929	0.010003	-0.012406	-0.002596
1	-0.499287	-0.532587	-0.456984	0.088489	-0.464270	0.046294	-0.005356	-0.012562	0.043945	0.027395	0.004628	0.006394
2	0.832777	-0.331479	-0.471075	-0.018672	0.173999	0.077012	-0.117121	-0.145521	-0.023793	0.052872	0.009305	-0.071040
3	0.863633	-0.282837	-0.492387	-0.059972	-0.293545	-0.108782	0.099432	-0.142524	0.090122	0.036494	0.002218	0.010245
4	-0.474527	-0.631709	0.548848	0.022170	-0.370284	0.012118	0.082537	-0.129728	-0.140121	-0.031074	0.067107	0.071883

Рис. 2.26 Навчальний набір після накладання на нього вибірки 12 факторів

2.3.5. Балансування даних

Ми показали, що набір даних, з яким ми маємо справу, незбалансований, оскільки приклади без дефолту представлені надмірно. Насправді, класовий дисбаланс є досить поширеною проблемою під час роботи з реальними даними. (Лістинг 29)

	Number	Percentage
Non-defaulters	17246	77.68%
Defaulters	4954	22.32%

Рис.2.27. Пропорції тренувального датасету

У нашому випадку ми могли б досягти майже 80% точності, просто прогнозуючи клас більшості (тих, хто не виконує вимоги) для всіх прикладів, без допомоги алгоритму машинного навчання з учителем. Таким чином, коли ми будемо підбирати класифікатори до наших наборів даних, було б доцільно зосередитися на інших метриках, ніж точність, під час порівняння різних моделей.

Окрім оцінки моделей машинного навчання, дисбаланс класів впливає на сам алгоритм навчання під час підбору моделі. Оскільки алгоритми машинного навчання зазвичай оптимізують функцію винагороди або вартості, яка обчислюється як сума за навчальними прикладами, які він бачить під час підбору, правило прийняття рішення, ймовірно, буде зміщене в бік класу більшості.

Одразу було виключено можливість збору більшої кількості даних. Один із способів боротьби з незбалансованими пропорціями класів під час підбору моделі – це призначити більший штраф за неправильні прогнози щодо класу меншості. Інші популярні стратегії боротьби з дисбалансом класів включають збільшення вибірки класу меншості, зменшення вибірки класу більшості та генерацію синтетичних навчальних прикладів. На жаль, не існує універсально найкращого рішення чи методу, який найкраще працює в різних областях проблеми. Таким чином, на практиці рекомендується випробувати різні стратегії для даної проблеми, оцінити результати та вибрати метод, який здається найбільш доцільним.

Було вирішено виключити наївні методи надмірної (недостатньої) вибірки, оскільки вони випадковим чином дублюють (видаляють) дані з класу меншості (більшості), доки не буде досягнуто бажаного рівня. З одного боку, випадкова недостатня вибірка не дозволяє контролювати, яка інформація відкидається, з іншого боку, випадкова надмірна вибірка призводить до перенавчання, оскільки модель навчається на багатьох ідентичних даних. Щоб зменшити ці проблеми, було вирішено застосувати метод кластерного центроїда та техніку синтетичної надмірної вибірки меншості (SMOTE).

- **Метод андерсеймплінгу кластерного центроїда**

Набір даних містить достатньо зразків у класі меншості, щоб було можливо виконати андерсеймплінг(обрізання) вибірки. Однак однією з основних проблем використання недостатньої вибірки є те, що важлива інформація може бути втрачена з класу більшості, що може призвести до надмірно загальних правил. Це неможливо для розробки моделі прогнозування дефолту за кредитною

карткою, особливо для зразків за дефолтом. Отже, щоб подолати цю проблему, було запроваджено метод кластерних центроїдів.

Ідея методу кластерних центроїдів полягає в заміні кластерів вибірок більшості відповідними центроїдами кластерів. До даних підбирається алгоритм K-середніх, а кількість кластерів встановлюється рівною кількості зразків кластерного класу. Потім більшість зразків з кластерів повністю замінюються наборами центроїдів кластерів з K-середніх. Центроїди кластерів містять найбільш репрезентативні варіації класу більшості, в яких значення ознак візуалізуються в центрі.(Рис.29)[20]

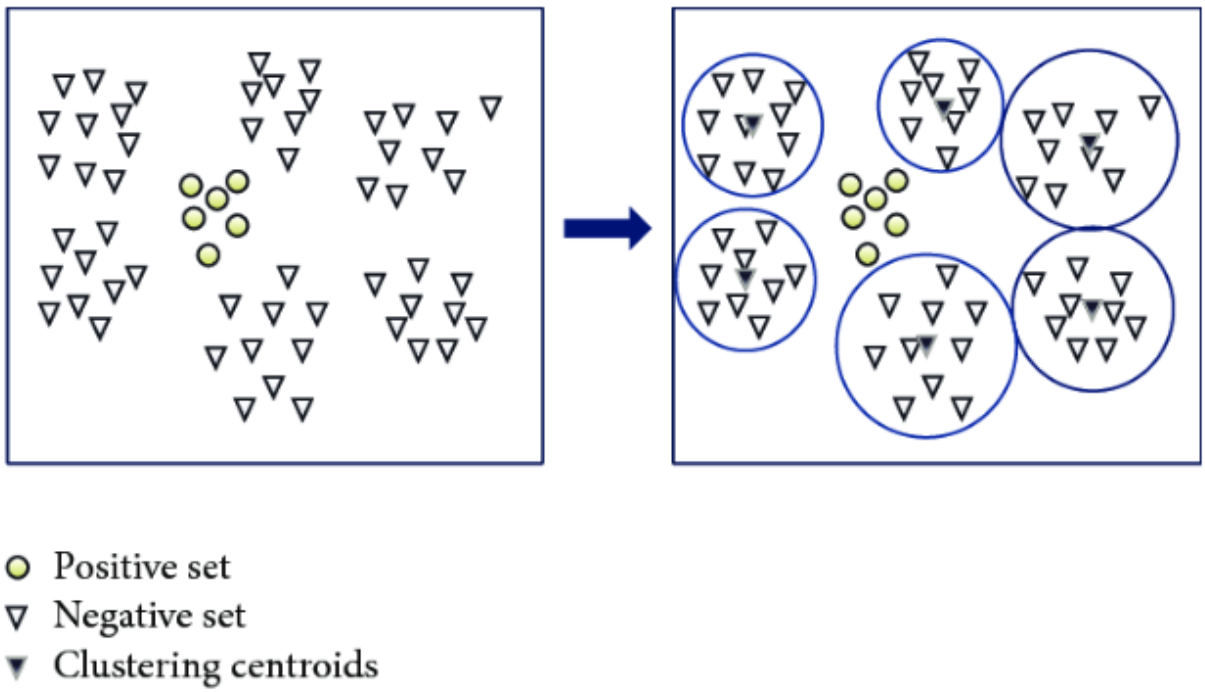


Рис.2.28. Андерсеймплінг кластерних центроїдів

У наведеному нижче фреймі даних ми показуємо результат недостатньої вибірки. Тепер навчальний набір збалансований, а кількість тих, хто не виконає зобов'язання, дорівнює кількості тих, хто виконає зобов'язання.

	Number	Percentage
Non-defaulters	4954	50.00%
Defaulters	4954	50.00%

Рис. 2.29. Результат обрізання вибірки по меншині прогнозованої категорії

- **Метод синтетичного оверсеймплінгу меншин (SMOTE)**

Метод синтетичної передискретизації меншин (SMOTE) був запропонований у 2000 році, щоб уникнути ризику перенавчання, з яким стикається випадкова передискретизація. Замість простого відтворення існуючих спостережень, цей метод генерує штучні вибірки. Як показано на рисунку 3.3, це досягається шляхом лінійної інтерполяції випадково вибраного спостереження меншини та одного з його сусідніх спостережень меншини. Точніше, SMOTE виконує три кроки для генерації синтетичної вибірки.[22]

1. Спочатку вибирається випадкове спостереження меншості a.
2. Серед k найближчих сусідів класу меншості вибирається екземпляр b.
3. Нарешті, нова вибірка x створюється шляхом випадкової інтерполяції двох вибірок: x = a + w * (b - a), де w – випадкова вага в [0, 1].

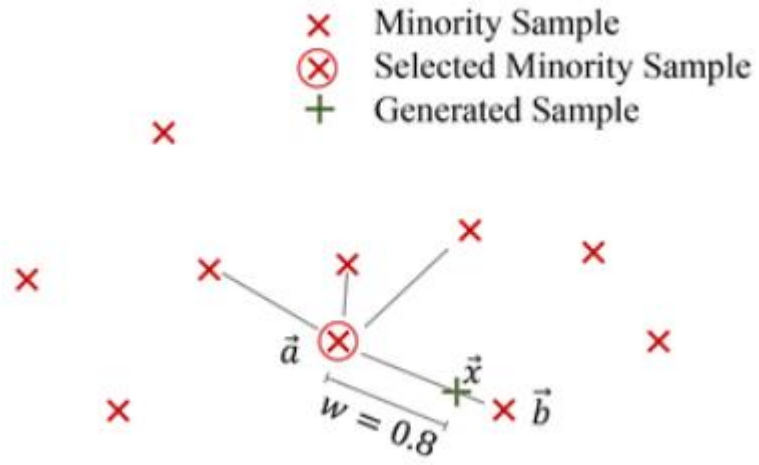


Рис. 2.30. SMOTE лінійно інтерполює випадково вибрану вибірку меншості та одного з її k = 4 найближчих сусідів.

У наведеному нижче фреймі даних ми показуємо результат передискретизації SMOTE. Тепер навчальний набір збалансований, а кількість тих, хто виконав зобов'язання, дорівнює кількості тих, хто не виконав зобов'язання.(Лістинг 31)

	Number	Percentage
Non-defaulters	17246	50.00%
Defaulters	17246	50.00%

Рис. 2.31. Пропорції тренувального датасету після SMOTE

Однак, алгоритм має деякі слабкі місця, пов'язані з дисбалансом та шумом, як показано на рисунку. Оскільки SMOTE випадковим чином вибирає екземпляр меншості для передискретизації з рівномірною ймовірністю, густонаселені райони меншин, ймовірно, будуть ще більше завищені, тоді як малонаселені райони меншин залишаються розрідженими. Ще однією важливою проблемою є те, що SMOTE схильний до генерації шуму. Це відбувається тому, що алгоритм не розпізнає шумні зразки меншин, які розташовані серед екземплярів класу більшості, і все ще інтерполює їх з їхнім найближчим сусідом меншини. Нарешті, алгоритм не забезпечує чітке дотримання межі прийняття рішення, хоча стверджується, що класифікатори могли б отримати користь від генерації зразків ближче до межі класу.

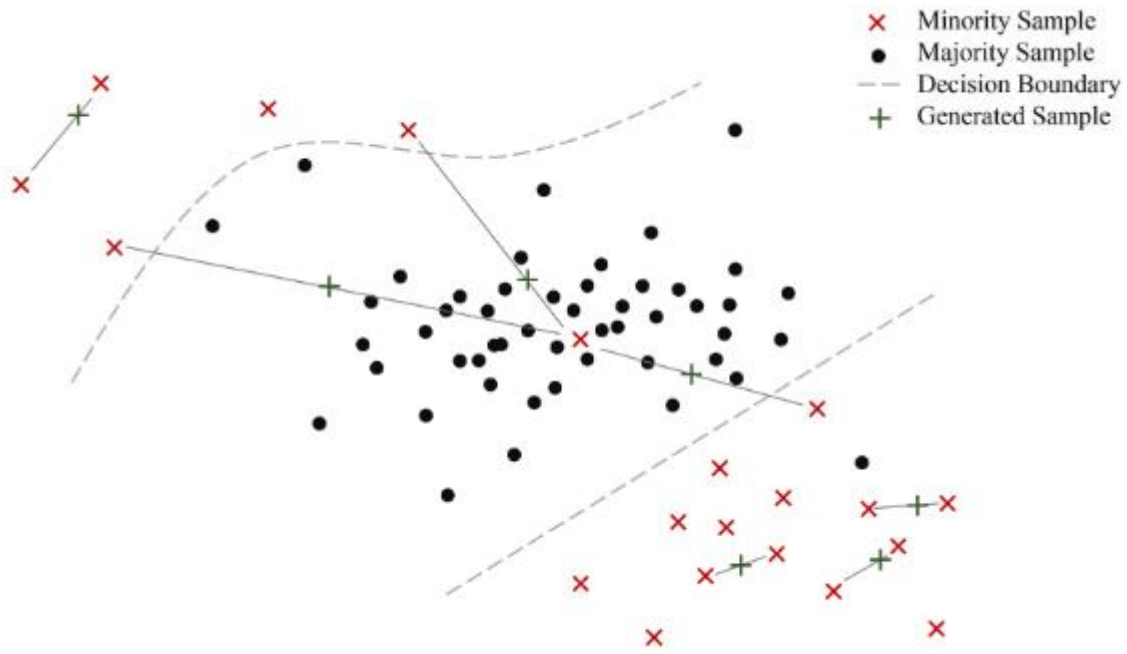


Рис. 2.32. Поведінка SMOTE за наявності шуму та дисбалансу всередині класу

- **K-means SMOTE**

Цей метод використовує простий і популярний алгоритм кластеризації K-means у поєднанні з оверсеймплінгом SMOTE для відновлення балансу асиметричних наборів даних. Він уникає генерації шуму шляхом оверсеймплінгу лише в безпечних областях (тобто областях, що складаються щонайменше з 50% вибірок меншин). Більше того, його увага зосереджена як на дисбалансі між класами, так і всередині класів, борючись з проблемою малих диз'юнктів шляхом роздування розріджених областей меншин.

K-means SMOTE складається з трьох кроків: кластеризація, фільтрація та оверсеймплінг.

На кроці кластеризації вхідний простір кластеризується в групи за допомогою кластеризації K-means.

Крок фільтрації вибирає кластери для оверсейплінгу, зберігаючи ті з високою часткою вибірок класів меншин. Потім він розподіляє кількість синтетичних вибірок для генерації, призначаючи більше вибірок кластерам, де вибірки меншин розподілені рідко.

Нарешті, на етапі оверсеймплінгу, SMOTE застосовується в кожному вибраному кластері для досягнення цільового співвідношення меншості та більшості випадків.

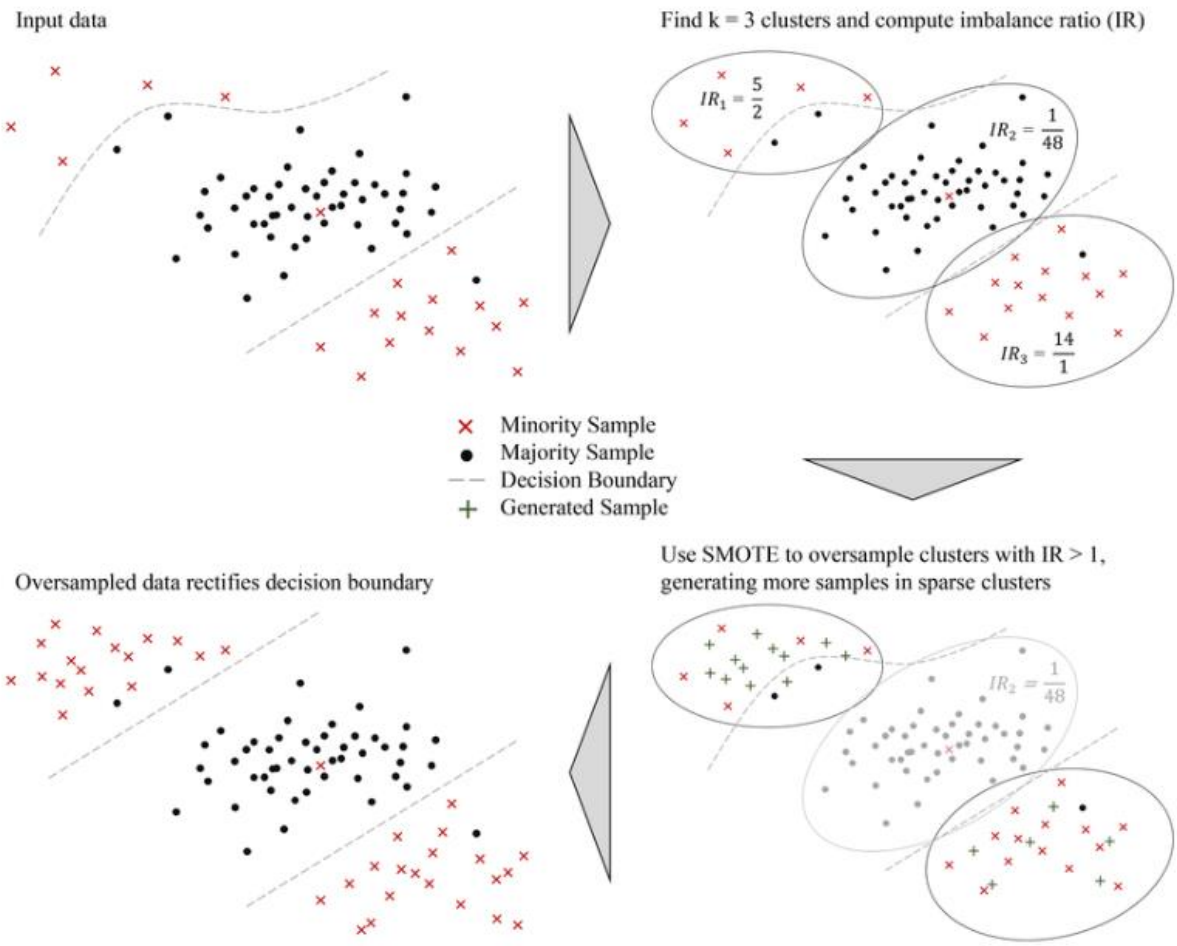


Рис. 2.33. K-means SMOTE оверсеймплить безпечні області та бореться з дисбалансом всередині класу.

У наведеному нижче фреймі даних ми показуємо результат K-means SMOTE оверсеймплінгу. Тепер навчальний набір збалансований, а кількість тих, хто не виконує вимоги, дорівнює кількості тих, хто не виконує вимоги. (Лістинг 32)

	Number	Percentage
Non-defaulters	17246	49.99%
Defaulters	17252	50.01%

Рис. 2.34 Пропорція класів навчальної вибірки після оверсеймплінгу SMOTE за допомогою K-середніх

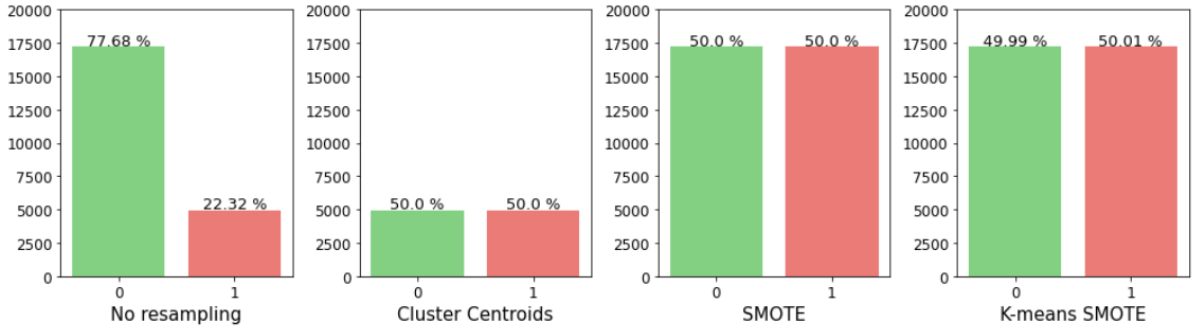


Рис. 2.35. Порівняння методів балансування

На Рис.2.35. Можна чітко спостерігати який результат мають представлені методи балансування.

2.4. Побудова моделей та налаштування гіперпараметрів

2.4.1. Побудова логістичної регресії

У ході дослідження було побудовано повний цикл роботи логістичної регресії з різними підходами до підготовки даних і балансування класів. Для цього реалізувавались допоміжні функції, які дозволяють не лише тренувати модель, а й наочно оцінювати її результати. Зокрема, функція plot_summary формує підсумковий графік: зліва вона показує стовпчикову діаграму значень F1 для кожної стратегії обробки даних, а справа — теплову карту матриці помилок для обраних прогнозів. Це дає змогу відразу бачити, які саме методи дозволяють краще розпізнавати приклади позитивного класу і де найчастіше трапляються помилки.

Окрема функція plot_test_scores використовується для детальнішої перевірки моделей на тестових даних. Вона обчислює точність, повноту, прецизійність, F1 та площу під кривою «precision–recall». Важливо, що в цій моделі використовується саме PR-AUC, оскільки в умовах незбалансованих даних ця метрика краще відображає здатність моделі працювати з рідкісним класом. Крім того, я додано базову лінію «no skill», яка відповідає випадковому вгадуванню і служить орієнтиром для оцінки якості. За потреби ця функція також будує матрицю помилок і криву precision–recall, що дозволяє не лише бачити числові значення, а й інтерпретувати поведінку моделі графічно.

Ключовим елементом є функція pipeline. Вона відповідає за вибір

підготовки даних: у режимі «raw» модель тренується на початкових нормалізованих ознаках без зниження розмірності; у режимі «PCA» дані попередньо перетворюються для зменшення кількості вимірів; а також передбачено використання методів балансування (SMOTE, KMeans-SMOTE та ClusterCentroids) у поєднанні з PCA. Таким чином можна буде порівняти, як саме різні підходи до балансування і підготовки даних впливають на результати логістичної регресії. Для налаштування моделі використано GridSearchCV, де підбирається параметр регуляризації C, що контролює складність моделі. Оцінювання проводилося за метрикою F1, оскільки вона рівномірно враховує як помилки першого, так і другого роду.

У підсумку було протестовано п’ять сценаріїв: роботу з «сирими» даними, PCA без балансування, PCA із SMOTE, PCA із KMeans-SMOTE та PCA із ClusterCentroids. Для кожного сценарію отримано набір метрик на тестових даних та зібрано їх у таблицю. Далі за допомогою підсумкового графіка відбувається порівняння усіх результаів, що дозволило зробити висновок про те, який підхід найкраще збалансовує точність і повноту в умовах дисбалансу класів.

Таким чином, побудована модель не лише показує значення стандартних метрик, а й дає змогу комплексно оцінити вплив різних стратегій препроцесингу на якість класифікації. Це дозволяє аргументовано визначити, яка конфігурація логістичної регресії є найефективнішою. (Лістинг 34)

Таблиця 2.1

Результати побудови варіацій моделі логістичної регресії

	Accuracy	**Recall**	**Precision**	**F1-score**	**AUC**
Raw data	0.804891	0.196245	0.734694	0.309751	0.555119
PCA	0.804216	0.192005	0.733796	0.304369	0.553024
PCA + SMOTE oversampling	0.631942	0.679588	0.338257	0.451691	0.544661

PCA + KMeansSMOTE oversampling	0.724767	0.363416	0.378310	0.370714	0.441867
PCA + ClusterCentroids oversampling	0.597757	0.666263	0.311968	0.424957	0.526340

Наведена вище Таблиця 2.1 підсумовує результати логістичної регресії на тестовому наборі, отримані шляхом експериментування з різними методами попередньої обробки, що застосовуються до навчального набору. PCA дозволяє нам досягти такої ж продуктивності, як і при використанні всіх даних, але з половиною ознак. Варто звернути увагу, що відсутність використання методу передискретизації призводить до найвищої точності та AUC, але призводить до нижчого показника F1. Оригінальний SMOTE перевершує як Cluster Centroid, так і новий K-means SMOTE.

На графіках ми бачимо різке падіння на початку кривих PR. Це пов'язано з тим, що поріг там особливо високий, тому для обчислення точності мало TP та FP, і навіть одна неправильна класифікація може призвести до великого зсуву. (Див. Додаток Б)

2.4.2. Побудова моделі за методом опорних векторів

У дослідженні також було реалізовано повний цикл побудови та перевірки моделей на основі методу опорних векторів (SVM). Основна увага приділялася тому, як різні підходи до підготовки та балансування даних впливають на результати класифікації.

Для цього було використано кілька варіантів препроцесингу. Перший режим – «Raw data» – передбачав роботу моделі з нормалізованими ознаками без додаткових перетворень. Другий варіант – застосування методу головних компонент (PCA) для зниження розмірності простору ознак. Третій, четвертий та п’ятий режими комбінували PCA з різними методами балансування: SMOTE, KMeans-SMOTE та ClusterCentroids. Такий підхід дав змогу оцінити, наскільки важливим є попереднє балансування у випадку сильно нерівномірного розподілу класів.[45][46][47]

Для налаштування моделі SVM було проведено підбір гіперпараметрів за допомогою сіткового пошуку (GridSearchCV). Випробовувалися значення параметра регуляризації C, типи ядерних функцій (rbf та poly), а також параметр gamma, що визначає вплив окремих точок на побудову гіперплощини. Оптимальні комбінації підбиралися з урахуванням метрик якості, що дозволило обрати найкращу конфігурацію моделі в кожному сценарії препроцесингу. (Лістинг 35)

У процесі аналізу було зібрано значення основних метрик: Accuracy, Recall, Precision, F1-score та AUC. Для підсумкової візуалізації було побудовано графіки, де відображено результати порівняння F1-score для кожного з підходів. Окремо було виведено матрицю помилок, що дало можливість візуально оцінити, як модель справляється з класифікацією обох класів.

У результаті було отримано порівняльну таблицю з усіма метриками для п’яти сценаріїв: «Raw data», «PCA», «PCA + SMOTE oversampling», «PCA + KMeans-SMOTE oversampling» та «PCA + ClusterCentroids oversampling». Це дозволило зробити висновки щодо того, як саме поєднання методів зменшення розмірності та балансування впливає на якість роботи SVM.

Таблиця 2.2

Результат моделювання методом опорних векторів

	Accuracy	**Recall**	**Precision**	**F1-score**	**AUC**
Raw data	0.822186	0.350697	0.703524	0.468068	0.599533
PCA	0.821646	0.349485	0.701094	0.466451	0.597847
PCA + SMOTE oversampling	0.769896	0.569352	0.486542	0.524700	0.575981
PCA + KMeansSMOTE oversampling	0.747061	0.490612	0.439978	0.463918	0.522112
PCA + ClusterCentroids oversampling	0.571409	0.738340	0.307906	0.434581	0.552308

Таким чином, побудована модель на основі методу опорних векторів

дозволила не лише визначити оптимальні параметри класифікації, а й дослідити роль різних стратегій підготовки даних у досягненні найкращої якості прогнозування. (Див. Додаток В)

Наведені результати показують, що Grid-Search зазвичай вибирає ядро та високе значення C. Це означає, що модель прагне правильно класифікувати всі навчальні приклади, а не побудувати гладку межу рішення.(Див. Додаток В)

Ми можемо бачити на датафреймі, що в цьому випадку відсутність передискретизації все ще призводить до хороших результатів на тестовому наборі з точки зору F1-оцінки. Ми вважаємо, що це пов'язано з тим, що моделі потрібно кілька значущих прикладів, які можна використовувати як опорні вектори для побудови межі рішення.(Див. Додаток В)

2.4.3. Побудова моделі методом дерев рішень

У ході дослідження було реалізовано експеримент із використанням алгоритму класифікації на основі рішучих дерев. На першому етапі було побудовано базову модель дерева рішень з обмеженням глибини до трьох рівнів та використанням критерію ентропії для поділу. Навчання проводилося на початковому наборі тренувальних даних, після чого структура дерева була візуалізована за допомогою графічного відображення, що дозволило наочно оцінити процес прийняття рішень та вагу окремих ознак.[38]

Для перевірки якості моделей було створено функцію, яка забезпечує тестування на вибірці перевірочних даних. Під час перевірки обчислювалися основні метрики класифікації: точність (Accuracy), повнота (Recall), точність у вузькому сенсі (Precision), F1-міра та площа під кривою Precision-Recall (AUC). Результати відображалися у вигляді числових значень, а також графічно: будувалися діаграми важливості ознак, матриця помилок та крива Precision-Recall із порівнянням відносно випадкового («No Skill») класифікатора.

Наступним кроком було створення узагальненої функції, що поєднує процес підбору оптимальних параметрів дерева рішень та оцінювання його результатів. Для цього застосовувався метод GridSearchCV з п’ятиразовою крос-

валідацією та цільовою функцією у вигляді максимізації F1-міри. У результаті було відібрано найкращу конфігурацію моделі для кожного випадку обробки даних.

Окрему увагу приділено аналізу впливу різних методів балансування та попередньої обробки вибірки. Для дослідження було використано п’ять підходів: необроблені дані без змін, застосування PCA, комбінація PCA з SMOTE, PCA з KMeansSMOTE та PCA з методом ClusterCentroids. Для кожної з цих стратегій було проведено навчання та тестування моделей, після чого результати зведено у таблицю для подальшого порівняння. (Лістинг 36) (Див. Додаток Г)

У підсумку було отримано структуровану оцінку роботи алгоритму дерева рішень у різних умовах(Таблиця 2.3), що дало змогу простежити залежність якості класифікації від методів попередньої обробки даних та технік балансування класів. Це створило підґрунтя для подальшого аналізу та порівняння з іншими алгоритмами машинного навчання.

Таблиця 2.3

Результат моделювання методом дерев рішень

	Accuracy	**Recall**	**Precision**	**F1-score**	**AUC**
Raw data	0.820565	0.359782	0.686705	0.472178	0.594653
PCA	0.807864	0.357965	0.620147	0.453917	0.560668
PCA + SMOTE oversampling	0.684232	0.481526	0.349297	0.404889	0.473242
PCA + KMeansSMOTE oversampling	0.745980	0.422168	0.429452	0.425779	0.490261
PCA + ClusterCentroids oversampling	0.546007	0.674743	0.282957	0.398712	0.515128

Результати, наведені у Таблиці 2.3, показують, що дерево рішень найкраще працює з необробленими даними. Маючи цю базову лінію, спробуємо покращити прогнозну ефективність, агрегуючи різні дерева рішень у випадковому лісі.

2.4.4. Побудова моделі методом Random Forest

У рамках дослідження була реалізована побудова прогнозної моделі із використанням ансамблевого алгоритму Random Forest, який поєднує в собі велику кількість дерев рішень для підвищення стійкості та точності класифікації. Мета експерименту полягала у порівнянні результатів цього методу з попередніми моделями, а також у вивченні впливу різних способів попередньої обробки даних.

Для оцінки ефективності алгоритму було застосовано п’ять підходів до формування навчальної вибірки: необроблені дані, використання головних компонент (PCA), поєднання PCA зі стандартним SMOTE, PCA з KMeansSMOTE та PCA з методом ClusterCentroids. Такий підхід дозволив простежити, наскільки методи балансування класів та зменшення розмірності впливають на роботу ансамблевих моделей.

У процесі навчання застосовувався метод GridSearchCV для підбору оптимальних параметрів. Серед них розглядалася кількість дерев у лісі (від 10 до 200) та різні варіанти визначення максимальної кількості ознак, що враховувалися при поділі вузлів. Оцінка відбувалася за допомогою п’ятиразової крос-валідації з метою максимізації F1-коефіцієнту. За результатами пошуку було визначено найкращі параметри для кожного з наборів даних.

Після цього моделі тестувалися на контрольній вибірці. Для кожного випадку обчислювалися основні метрики якості: точність класифікації (Accuracy), повнота (Recall), точність у вузькому сенсі (Precision), F1-міра та площа під кривою Precision-Recall (AUC). Отримані результати були зведені у підсумкову таблицю, що дало змогу провести порівняльний аналіз ефективності роботи алгоритму на даних, оброблених різними способами. (Лістинг 37)

Додатково результати оцінювалися візуально. Використовувалися графіки важливості ознак, матриці помилок та криві Precision-Recall, що дозволяло не лише зафіксувати числові показники, але й оцінити характер розподілу помилок моделі. (див. Додаток Д)

У підсумку було показано, що використання Random Forest забезпечує вищу стабільність та точність порівняно з окремим деревом рішень, а також демонструє суттєву залежність від методів балансування та трансформації даних. Отримані результати створили основу для подальшого порівняння ансамблевих алгоритмів із традиційними статистичними методами та іншими моделями машинного навчання.

Таблиця 2.4

Результати побудови моделі методом Random Forest

	Accuracy	**Recall**	**Precision**	**F1-score**	**AUC**
Raw data	0.820970	0.396729	0.665650	0.497154	0.598478
PCA	0.807458	0.362810	0.616255	0.456729	0.560604
PCA + SMOTE oversampling	0.764762	0.516051	0.474916	0.494630	0.549463
PCA + KMeansSMOTE oversampling	0.784489	0.427014	0.520679	0.469218	0.537757
PCA + ClusterCentroids oversampling	0.583029	0.649909	0.299637	0.410168	0.513822

Результати, наведені у Таблиці 2.4, показують, що агрегування кількох дерев у лісі призводить до загального покращення ефективності прогнозування. Моделі «Бегінг» та «Випадковий ліс» не можна інтерпретувати як окремі дерева рішень. Однак вони дозволяють нам вимірювати та будувати графік важливості ознак як усереднене зменшення домішок, обчислене з усіх дерев рішень в ансамблі, без будь-яких припущень щодо того, чи є наші дані лінійно роздільними чи ні.

Зокрема, ми можемо звернутися до випадкового лісу, навченого на необроблених даних, оскільки він пропонує дійсно корисний графік важливості ознак.(Див. Додаток Д) Коли фінансові установи розглядають можливість видачі клієнту кредитної картки, їм потрібно перевірити історію платежів цієї особи, оскільки якщо він вже має численні рахунки, він, ймовірно, затримає платіж

поточного місяця. Окрім історії платежів, також важливо переглянути кредитний ліміт заявників за їхніми поточними кредитними картками. Це результат позитивного кола: люди, які своєчасно платять, як правило, мають кращі кредитні рейтинги, тому банки вважають за краще збільшувати кредитні лінії цих людей, беручи на себе менший ризик. Персональна інформація клієнта також впливає на поведінку за замовчуванням, оскільки банки збирають її, коли люди подають заявки на кредитні картки. Однак ми вважаємо, що фінансові установи повинні однаково враховувати своїх потенційних клієнтів, незалежно від того, чи є вони чоловіками чи жінками, випускниками середньої школи чи університету, самотніми чи одруженими, коли вирішують, чи схвалювати їхні заявки на кредитні картки.

2.5. Оцінка результатів за допомогою визначених метрик

Наведена нижче Таблиця 2.5 підсумовує результати комбінації різних алгоритмів машинного навчання та методів передискретизації на тестовому наборі. На рисунку нижче порівнюються F1-оцінки різних моделей на тестовому наборі. Варто також уточнити, що логістична регресія та SVM працювали не зі справжніми необробленими даними, а з нормалізованими даними, на відміну від дерева рішень та випадкового лісу.

Таблиця 2.5

Порівняння результатів моделювання

		Accuracy	**Recall**	**Precision**	**F1-score**	**AUC**
Logistic Regression	**Raw data**	0.804891	0.196245	0.734694	0.309751	0.555119
	PCA	0.804216	0.192005	0.733796	0.304369	0.553024
	PCA + SMOTE oversampling	0.631942	0.679588	0.338257	0.451691	0.544661
	PCA + KMeansSMOTE oversampling	0.724767	0.363416	0.378310	0.370714	0.441867

	PCA + ClusterCentroids oversampling	0.597757	0.666263	0.311968	0.424957	0.526340
Support Vector Machine	**Raw data**	0.822186	0.350697	0.703524	0.468068	0.599533
	PCA	0.821646	0.349485	0.701094	0.466451	0.597847
	PCA + SMOTE oversampling	0.769896	0.569352	0.486542	0.524700	0.575981
	PCA + KMeansSMOTE oversampling	0.747061	0.490612	0.439978	0.463918	0.522112
	PCA + ClusterCentroids oversampling	0.571409	0.738340	0.307906	0.434581	0.552308
Decision Tree	**Raw data**	0.820565	0.359782	0.686705	0.472178	0.594653
	PCA	0.807864	0.357965	0.620147	0.453917	0.560668
	PCA + SMOTE oversampling	0.684232	0.481526	0.349297	0.404889	0.473242
	PCA + KMeansSMOTE oversampling	0.745980	0.422168	0.429452	0.425779	0.490261
	PCA + ClusterCentroids oversampling	0.546007	0.674743	0.282957	0.398712	0.515128
Random Forest	**Raw data**	0.820970	0.396729	0.665650	0.497154	0.598478
	PCA	0.807458	0.362810	0.616255	0.456729	0.560604
	PCA + SMOTE oversampling	0.764762	0.516051	0.474916	0.494630	0.549463
	PCA + KMeansSMOTE oversampling	0.784489	0.427014	0.520679	0.469218	0.537757

	PCA + ClusterCentroids oversampling	0.583029	0.649909	0.299637	0.410168	0.513822

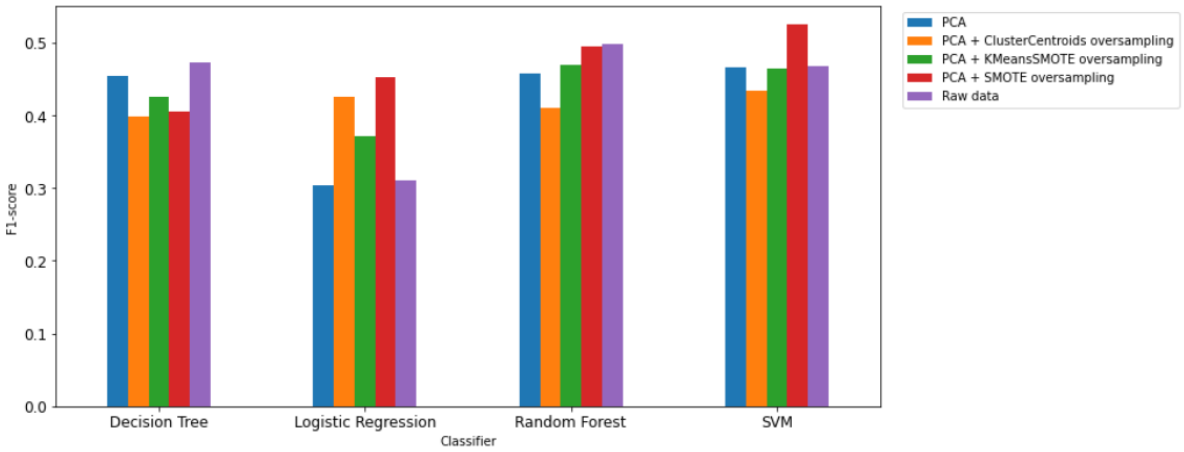


Рис. 2.36. Порівняння результатів моделювання за коефіцієнтом F1 (Лістинг 38)

Загалом, оверсеймплінг покращує прогностичну ефективність порівняно з недостатньою вибіркою. Оригінальний алгоритм SMOTE працює краще, ніж нова версія, і дозволяє отримати найкращу модель при використанні разом з SVM. Однак, варто визнати, що результати випадкового лісу є вражаючими. Фактично, ця модель не тільки добре працює з необробленими даними (другий найкращий результат загалом), але й надає рейтинг важливості ознак, який може бути корисним для фінансових установ, з якими ми працюємо.

2.6. Оцінка економічного ефекту від впровадження моделі у діяльність банку

Оцінювання економічного ефекту ґрунтується на визначенні того, які фінансові переваги матиме банк у разі впровадження побудованої моделі прогнозування ймовірності прострочення чи неповернення кредитної заборгованості. Упродовж дослідження було сформовано систему показників, які демонструють можливості моделей підвищувати точність кредитного скорингу, впливати на ризикові рішення та, відповідно, знижувати потенційні фінансові втрати.

Передусім варто підкреслити, що застосовані алгоритми машинного навчання показали здатність ефективно ідентифікувати клієнтів з підвищеною

кредитною ризиковістю. Навіть у випадках, коли точність моделі (accuracy) не демонструє кардинального зростання, такі метрики, як recall та F1-score, свідчать про покращення здатності моделі «вловлювати» проблемні випадки. Для банку саме ця здатність має найвагоміше економічне значення, оскільки дозволяє мінімізувати кількість клієнтів, які потенційно можуть привести до збитків.

Економічний ефект формується за декількома ключовими напрямами:

По-перше, точніше прогнозування дефолту зменшує ймовірність видачі ризикових кредитів, що безпосередньо знижує рівень формування резервів під можливі втрати. Навіть незначне підвищення recall у моделях SVM та Random Forest суттєво впливає на зменшення частки проблемних кредитів у портфелі. У практичному вимірі це означає оптимізацію кредитного портфеля, підвищення його ліквідності та зниження потреби у додатковому капіталі для покриття потенційних збитків.

По-друге, модель дозволяє банку оптимізувати процес прийняття рішень, прискорюючи оцінювання заявок і зменшуючи трудові витрати. Автоматизація, що ґрунтується на точних прогнозах, скорочує час аналітиків, знижує навантаження на кредитні відділи та дає змогу обробляти більшу кількість заявок без розширення штату. Це формує додатковий непрямий економічний ефект, пов’язаний зі зменшенням операційних витрат.

По-третє, покращення точності прогнозів сприяє підвищенню якості взаємодії з клієнтами. Банк може індивідуалізувати пропозиції, впроваджувати диференційоване кредитування та встановлювати більш обґрунтовані процентні ставки, що загалом підвищує дохідність та конкурентоспроможність фінансової установи.

Окремого значення набуває можливість моделі працювати з незбалансованими даними – характерною проблемою банківських кредитних портфелів. Використання стратегій балансування, таких як SMOTE чи ClusterCentroids, продемонструвало підвищення чутливості моделей до рідкісного класу «дефолтних» клієнтів. Це створює додатковий економічний ефект у вигляді точнішого відбору та зменшення частки кредитів, що стають

проблемними вже в перші місяці після видачі.

У межах дослідження було здійснено детальну оцінку потенційного економічного ефекту від упровадження побудованої моделі машинного навчання у практичну діяльність банківської установи. Основна увага була приділена можливості зниження кредитних втрат, підвищенню точності раннього виявлення ризикових клієнтів та оцінці операційної економії, що виникає внаслідок автоматизації процесів прийняття рішень.

Для побудови орієнтовної фінансової моделі було використано гіпотетичний кредитний портфель обсягом 100 000 клієнтів із середнім залишком 1 000 дол. США. Частку фактичних дефолтів прийнято на рівні 5%, що відповідає приблизно 5 000 несправним кредитам. Втрати при дефолті оцінювалися через параметр LGD, прийнятий рівним 60%, тобто середня втрата на один дефолт становила близько 600 доларів США. Порівняння базової моделі з розробленою дозволило встановити, що використання алгоритму SVM у поєднанні із методом балансування SMOTE забезпечує істотне зростання показника recall – з приблизно 0.196 (логістична регресія на необроблених даних) до 0.569. Це означає, що система здатна додатково виявити близько 0.373 частки дефолтів, що в абсолютних значеннях становить орієнтовно 1 866 клієнтів.

Усі додатково виявлені ризикові випадки можуть стати об’єктом превентивних заходів, таких як реструктуризація, узгодження індивідуальних графіків погашення або зниження кредитного ліміту. Для оцінки ефективності було прийнято припущення, що такі заходи здатні зменшити втрати на 50%, тобто зберегти близько 300 доларів на кожен потенційний дефолт. За цих умов загальна економія на зниженні кредитних втрат становить приблизно 559 660 доларів США.

Окрім безпосереднього впливу на рівень дефолтів, було враховано потенційну операційну економію, яка виникає завдяки автоматизації процесів ухвалення рішень. За умови, що модель дозволяє скоротити ручну обробку заявок на 20%, а вартість одного ручного розгляду становить 10 доларів США, загальна економія для портфеля з 100 000 клієнтів дорівнює близько 200 000

доларів США. Витрати на впровадження моделі та її супровід у перший рік за умовним розрахунком складають 150 000 доларів. У підсумку сумарний економічний ефект за перший рік становить близько 759 660 доларів, а чистий з урахуванням витрат – орієнтовно 609 660 доларів США. Це відповідає приблизному ROI на рівні понад 500% у перший рік використання.

Отримані результати демонструють, що впровадження моделі машинного навчання здатне забезпечити значну економію, як у частині зниження втрат, так і в частині оптимізації внутрішніх операцій. Наведені розрахунки мають умовний характер, однак вони дозволяють сформувати уявлення про порядок величини економічного ефекту та обґрунтувати доцільність інвестицій у подібні рішення. Таке моделювання може бути використане банком для підготовки бізнес-кейсу, планування впровадження та подальшого удосконалення систем управління кредитними ризиками.

2.7. Висновки

Проведене емпіричне дослідження на наборі даних кредитних карток клієнтів з Тайваню дозволило отримати комплексне уявлення про ефективність різних методів машинного навчання у прогнозуванні кредитних ризиків та вплив підходів до підготовки даних на якість моделей.

Дослідження підтвердило критичну важливість етапу попередньої обробки даних. Виявлено, що навіть відносно невеликий відсоток некоректних значень може суттєво впливати на результати моделювання. Очищення даних від недокументованих категорій у змінних освіти та сімейного стану, хоча й призвело до втрати незначної частини спостережень, забезпечило підвищення стабільності моделей. Застосування методу головних компонент дозволило зменшити розмірність простору ознак удвічі, зберігши при цьому понад 99% дисперсії даних, що підтверджує наявність значної кореляції між окремими змінними.

Особливу увагу приділено проблемі дисбалансу класів, характерній для задач кредитного скорингу. Частка дефолтних клієнтів становила лише 22,1% від

загальної кількості, що створювало ризик упереджених прогнозів у бік класу більшості. Порівняння трьох підходів до балансування показало неоднозначні результати: класичний SMOTE демонстрував кращу ефективність порівняно з новішими модифікаціями KMeans-SMOTE та ClusterCentroids. Водночас метод недостатньої вибірки з використанням кластерних центроїдів, хоча й зберігав репрезентативність даних, призводив до зниження загальної точності класифікації.

Дослідження виявило суттєві відмінності у поведінці різних алгоритмів машинного навчання. Логістична регресія показала стабільну роботу та забезпечила високу інтерпретованість результатів, проте її здатність виявляти складні нелінійні залежності виявилася обмеженою. F1-показник для найкращої конфігурації логістичної регресії склав 0,452 при використанні SMOTE-балансування, що свідчить про помірну ефективність у розпізнаванні дефолтних клієнтів.

Метод опорних векторів продемонстрував найвищі результати серед усіх досліджених алгоритмів. Найкраща конфігурація SVM з SMOTE-балансуванням досягла F1-показника 0,525 та AUC 0,576, що підтверджує ефективність ядрових методів у роботі з нелінійно роздільними даними. Характерно, що SVM показав стабільні результаті як на нормалізованих даних без додаткової обробки, так і після застосування методів балансування.

Деревоподібні алгоритми виявили цікаві особливості поведінки. Окреме дерево рішень досягло F1-показника 0,472 на необроблених даних, демонструючи природну стійкість до дисбалансу класів. Random Forest показав подальше покращення результатів до F1-показника 0,497, підтверджуючи переваги ансамблевого підходу. Важливою перевагою деревоподібних методів стала можливість ранжування важливості ознак, що виявило ключову роль історії платежів та кредитних лімітів у прогнозуванні дефолтів.

Аналіз впливу масштабування даних показав, що нормалізація переважає стандартизацію у випадку наявності викидів, що характерно для фінансових даних. Методи, чутливі до масштабу ознак, такі як логістична регресія та SVM,

продемонстрували кращу продуктивність після нормалізації, тоді як деревоподібні алгоритми залишилися інваріантними до типу масштабування.

Дослідження виявило парадоксальний результат щодо методів балансування. У більшості випадків моделі, навчені на несбалансованих даних, показували вищу загальну точність та AUC, але гірший F1-показник порівняно з моделями після балансування. Це пояснюється тим, що балансування підвищує чутливість до класу меншості за рахунок збільшення кількості хибнопозитивних прогнозів. Для практичного застосування це означає необхідність ретельного вибору стратегії залежно від економічних наслідків різних типів помилок.

Результати дослідження підтверджують відсутність універсального "найкращого" алгоритму для всіх сценаріїв. SVM з SMOTE-балансуванням показав найвищий F1-показник, Random Forest на необроблених даних забезпечив другий результат з додатковою перевагою інтерпретованості, а логістична регресія зберігає актуальність завдяки простоті та відповідності регуляторним вимогам.

Практичне значення дослідження полягає у демонстрації важливості комплексного підходу до вибору методу машинного навчання з урахуванням специфіки даних, бізнес-цілей та операційних обмежень. Результати свідчать, що успішне впровадження ML-методів у кредитному скорингу потребує не лише технічної досконалості моделей, а й глибокого розуміння природи фінансових даних та економічного контексту прийняття рішень.

Дослідження також виявило обмеження використаного набору даних, зокрема його відносно невеликий розмір та обмежений часовий період спостережень. Для підвищення практичної цінності результатів рекомендується проведення аналогічних досліджень на більших та більш сучасних наборах даних з включенням додаткових типів ознак, характерних для цифрової епохи банкінгу.

Побудована фінансова модель показала, що підвищення чутливості системи до ризикових клієнтів дозволяє банку виявляти близько 1 866 додаткових випадків потенційного дефолту. За умови застосування

превентивних заходів, здатних зменшити очікувані втрати на 50%, економія на скороченні кредитних збитків може становити близько 559 тис. доларів США. Додатково враховано операційну економію, пов’язану зі зниженням навантаження на співробітників завдяки автоматизації процесів ухвалення рішень. Скорочення ручної обробки заявок на 20% дає змогу зекономити ще близько 200 тис. доларів. Навіть з урахуванням витрат на впровадження моделі, оцінених у 150 тис. доларів, чистий економічний ефект перевищує 600 тис. доларів, демонструючи високий рівень рентабельності інвестицій.

У цілому, проведене дослідження підтверджує потенціал методів машинного навчання для підвищення ефективності прогнозування кредитних ризиків, водночас демонструючи складність та багатогранність цього процесу. Успішне впровадження потребує збалансованого підходу, що враховує як статистичну точність моделей, так і їх практичну застосовність у реальному банківському середовищі.

РОЗДІЛ 3

УПРАВЛІННЯ ПРОЄКТОМ ВПРОВАДЖЕННЯ МОДЕЛІ МАШИННОГО НАВЧАННЯ У БАНКІВСЬКУ ПРАКТИКУ

3.1. Планування проєкту впровадження моделі

Планування проєкту впровадження моделі прогнозування кредитних ризиків є фундаментальним етапом, що визначає успіх усіх подальших дій. Воно охоплює як концептуальне бачення цілей і завдань, так і практичну організацію процесу реалізації. На цьому етапі формується стратегічний документ – план проєкту, у якому відображаються ключові віхи, необхідні ресурси, строки виконання та очікувані результати.

Першим кроком є визначення мети та постановка завдань. Основна мета полягає у створенні та впровадженні моделі машинного навчання, здатної з високою точністю прогнозувати ймовірність дефолту позичальників та знижувати ризики кредитного портфеля банку. Завдання конкретизуються у вигляді підцілей: збір та підготовка даних, побудова кількох моделей із різними конфігураціями, їхнє порівняння за допомогою релевантних метрик, інтеграція найефективнішої моделі у бізнес-процеси. У цьому дослідженні було реалізовано кілька моделей машинного навчання, серед яких найвищий показник F1 продемонструвала Support Vector Machine із застосуванням методу головних компонент (PCA) та техніки SMOTE для балансування вибірки. Саме цей результат закладає основу для планування подальших інтеграційних кроків.

Другим важливим аспектом є розробка дорожньої карти проєкту. Для цього доцільно поєднувати традиційний підхід до проєктного менеджменту (із чітким визначенням етапів та контрольних точок) із гнучкими методиками, такими як Kanban. Використання Kanban на етапі планування дозволяє структурувати великий обсяг робіт у вигляді окремих завдань, які відображаються на візуальній дошці. Завдяки цьому забезпечується прозорість процесу, легкість у визначенні пріоритетів і можливість швидкого реагування на

зміни. Дорожня карта включає три взаємопов’язані блоки: підготовчий (збір даних, їхня очистка та нормалізація), аналітико-експериментальний (побудова моделей, тестування та вибір оптимальної конфігурації), а також інтеграційний (розробка прототипу у середовищі банку, адаптація процесів і навчання персоналу).

Наступним кроком у плануванні виступає визначення показників успішності проєкту. Вони мають охоплювати не лише технічні метрики, як-от F1, Precision чи Recall, але й бізнес-орієнтовані індикатори: скорочення частки проблемних кредитів, підвищення прибутковості кредитного портфеля, зменшення часу на прийняття рішення про видачу позики. Таким чином, планування поєднує два рівні: математичний (якість моделі) та економічний (ефективність для банку).

Важливе місце займає також розробка комунікаційного плану. Впровадження нових аналітичних інструментів у банку нерозривно пов’язане з необхідністю взаємодії між підрозділами: ІТ-відділом, аналітиками ризиків, відділом кредитування та управлінням персоналу. На етапі планування визначаються канали та періодичність комунікацій, що дозволяє уникати непорозумінь і сприяє узгодженості дій. У рамках Kanban це може реалізовуватися через регулярні щотижневі зустрічі команди для обговорення прогресу та виявлення можливих проблемних точок.

Отже, планування проєкту впровадження моделі прогнозування кредитних ризиків у банку постає як багатовимірний процес. Воно включає постановку мети та завдань, розробку дорожньої карти із використанням гнучких підходів управління, визначення критеріїв успішності та формування ефективної системи комунікацій. Саме цей етап забезпечує узгодженість технічної, організаційної та бізнесової складових, створюючи підґрунтя для успішної реалізації проєкту.

Таблиця 3.1

Дорожня карта проєкту впровадження

Етап	Завдання	Орієнтовна тривалість	Очікуваний результат
Підготовчий	Збір та валідація даних; очистка, нормалізація та кодування змінних	3–4 тижні	Якісна навчальна вибірка для побудови моделей
Аналітико-експериментальний	Побудова моделей з різними параметрами; застосування PCA та SMOTE; оцінка метрик (F1, Precision, Recall)	5–6 тижнів	Вибір оптимальної моделі – SVM із PCA та SMOTE
Інтеграційний	Розгортання прототипу в IT-середовищі банку; тестування на історичних даних; навчання персоналу	4 тижні	Робочий прототип моделі в бізнес-процесах
Фінальний	Пілотне застосування на сегменті клієнтів; моніторинг ефективності; корекція гіперпараметрів	6–8 тижнів	Доказ практичної цінності моделі та підготовка до масштабного впровадження

Така структура дозволяє інтегрувати технічні завдання з управлінськими, визначаючи пріоритети та послідовність дій.

3.2. Оцінка ресурсів, строків та бюджетування

Оцінка ресурсів, строків та бюджетування є другим критично важливим етапом у впровадженні моделі прогнозування кредитних ризиків. Вона дає змогу

сформувати реалістичний план виконання проєкту, визначити необхідні інвестиції та оцінити, наскільки обґрунтованим є очікуваний економічний ефект від впровадження.

Передусім здійснюється оцінка ресурсів. До них належать як технічні, так і людські. З технічного боку банку необхідні обчислювальні потужності для навчання та тестування моделей. Це можуть бути як внутрішні сервери, так і орендовані хмарні рішення (наприклад, AWS, Google Cloud чи Microsoft Azure), що дають можливість масштабувати обчислення у періоди високого навантаження. Важливо також врахувати потребу у спеціалізованому програмному забезпеченні для аналізу даних та побудови моделей машинного навчання. Серед людських ресурсів визначаються аналітики даних, дата-сайентісти, програмісти, бізнес-аналітики та фахівці з інформаційної безпеки. Кожен із цих учасників має специфічні завдання: від підготовки даних та побудови моделей до забезпечення інтеграції та захисту даних.

Наступний етап стосується оцінки строків реалізації проєкту. Вона здійснюється шляхом декомпозиції робіт на окремі завдання з визначенням їхньої тривалості. Для побудови реалістичного графіка зручно поєднувати традиційні методи (наприклад, діаграму Ґанта) із гнучкими підходами, зокрема Kanban. Діаграма Ґанта дозволяє візуалізувати залежності між етапами: підготовка даних → побудова моделей → тестування → інтеграція. Kanban, своєю чергою, допомагає уникати перевантаження команди завдяки обмеженню кількості завдань у роботі та забезпечує прозорість у виконанні задач. Орієнтовно весь процес упровадження моделі може тривати від трьох до шести місяців, залежно від складності даних і масштабів інтеграції.

Третій складник цього етапу – бюджетування. Воно охоплює прямі та непрямі витрати. До прямих належать витрати на хмарні обчислювальні ресурси, ліцензії програмного забезпечення та заробітні плати членів проєктної команди. Непрямі витрати включають витрати на навчання персоналу, модернізацію IT-інфраструктури, а також резервний фонд на випадок непередбачуваних обставин (наприклад, необхідності придбати додаткові дані чи інструменти). Важливою

частиною бюджетування є прогноз економічного ефекту від застосування моделі. Він може вимірюватися через зменшення обсягу прострочених кредитів, підвищення точності ухвалення рішень та скорочення витрат на обробку кредитних заявок.

Ключовим інструментом управління ресурсами та бюджетом виступає баланс між витратами і вигодами. Банківська установа має розглядати впровадження моделі не лише як інновацію, а й як інвестицію, яка повинна окупитися у визначені строки. Для цього доцільно застосовувати методи фінансової оцінки ефективності, зокрема розрахунок ROI (Return on Investment) чи NPV (Net Present Value).

Отже, оцінка ресурсів, строків та бюджетування формує міцну основу для подальших етапів проєкту. Вона поєднує технічні, організаційні та економічні складові, забезпечуючи збалансованість між можливостями банку, реалістичністю виконання та очікуваною вигодою. Завдяки цьому впровадження моделі прогнозування кредитних ризиків стає не лише технічно обґрунтованим, а й економічно доцільним процесом.

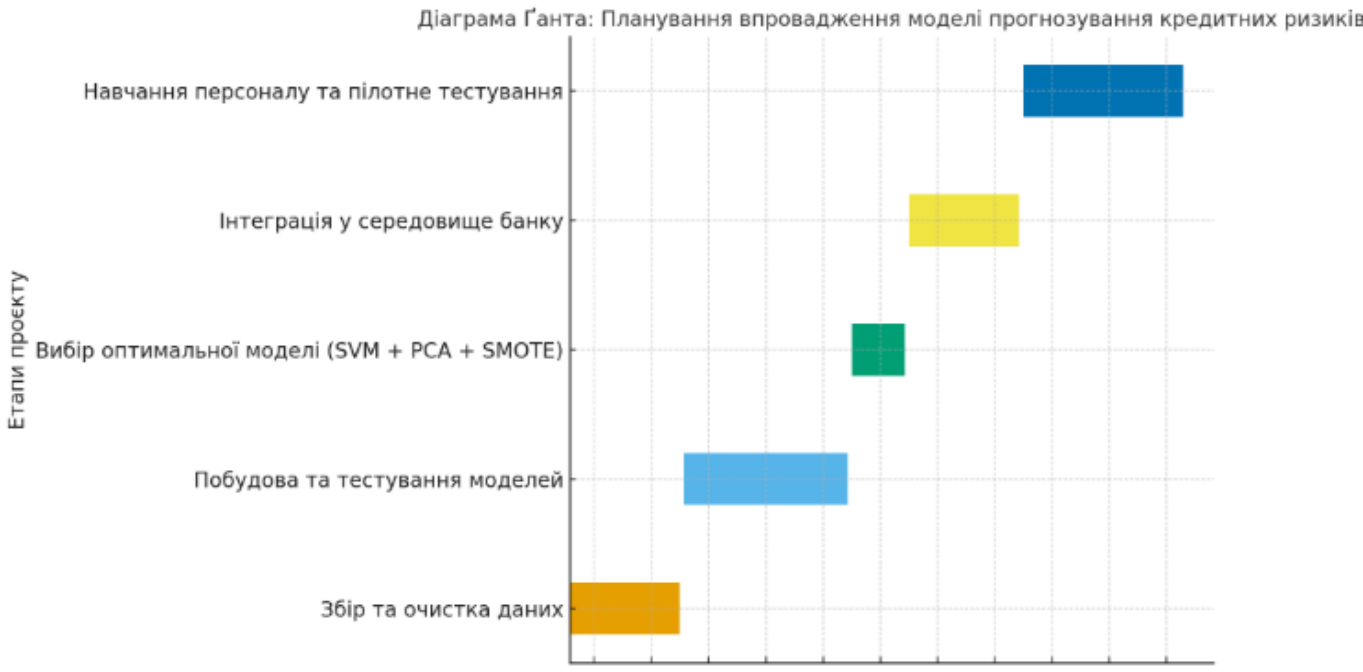


Рис. 3.1. Діаграма Ганта: Планування впровадження моделі прогнозування кредитних ризиків

3.3. Організація команди та розподіл ролей

Організація команди та ефективний розподіл ролей є центральним елементом у впровадженні моделі прогнозування кредитних ризиків, оскільки успіх проєкту залежить не лише від технічної досконалості обраної моделі, а й від злагодженої роботи міждисциплінарної команди. Банківські проєкти в сфері машинного навчання передбачають тісну співпрацю аналітиків, технічних спеціалістів і представників бізнес-підрозділів, що забезпечує цілісність підходу.

На початковому етапі формується проєктна команда, яка зазвичай складається з кількох ключових груп. До першої групи належать аналітики даних та дата-сайентісти, що відповідають за підготовку даних, побудову моделей і проведення експериментів. Вони здійснюють як базові завдання (збір, очистка та трансформація даних), так і складніші – тестування моделей, застосування методів зниження розмірності (PCA), балансування вибірки (SMOTE) та аналіз результатів за допомогою метрик якості (F1, Precision, Recall). Саме ця група визначає технічну основу впровадження.

Друга група – розробники програмного забезпечення та IT-інженери. Їхня роль полягає у створенні стабільного середовища для інтеграції моделі в банківську інфраструктуру. Вони реалізують API, налагоджують автоматизовані канали передачі даних, забезпечують сумісність із наявними інформаційними системами. Крім того, ця група відповідає за масштабування рішень та технічну підтримку після запуску.

Третя група – бізнес-аналітики та експерти з управління ризиками. Вони забезпечують зв’язок між технічною стороною та бізнес-цілями банку. Їхнє завдання полягає у формулюванні вимог до моделі, інтерпретації результатів прогнозування у фінансових термінах та оцінці економічної доцільності впровадження. Завдяки їм забезпечується узгодженість між математичними показниками та бізнес-ефектом, як-от скорочення частки проблемних кредитів чи оптимізація кредитного портфеля.

Четверта група – фахівці з інформаційної безпеки та комплаєнсу. Вони відповідають за дотримання законодавчих вимог і внутрішніх політик банку, забезпечення конфіденційності клієнтських даних та захист моделі від потенційних кібератак. Цей аспект є особливо важливим, оскільки моделі машинного навчання оперують великими масивами персональної інформації, яка має бути захищеною відповідно до міжнародних стандартів.

Нарешті, важливу роль відіграє менеджер проєкту, який координує діяльність усіх учасників, контролює строки виконання завдань, стежить за бюджетом і ризиками. У контексті використання Kanban менеджер відповідає за організацію робочих процесів на дошці завдань, визначення пріоритетів та усунення перешкод, що виникають у роботі команди.

Розподіл ролей у команді відображається також у комунікаційній структурі. Регулярні наради, щотижневі звіти про прогрес та візуалізація завдань у Kanban дозволяють уникати дублювання функцій та втрати інформації. Крім того, такий підхід сприяє підвищенню гнучкості команди: у випадку зміни пріоритетів чи виникнення непередбачуваних ситуацій завдання можуть бути оперативно перерозподілені без шкоди для загального прогресу.

Отже, організація команди та розподіл ролей у процесі впровадження моделі прогнозування кредитних ризиків – це складна, але необхідна управлінська задача. Вона базується на принципах міждисциплінарної взаємодії, чіткому розмежуванні відповідальності та використанні сучасних методів управління, таких як Kanban, які забезпечують прозорість і гнучкість робочих процесів.

3.4. Ризики та шляхи їх мінімізації у процесі впровадження

У процесі впровадження моделі прогнозування кредитних ризиків у банківську практику виникає низка факторів невизначеності, які можуть суттєво вплинути як на якість кінцевого результату, так і на строки та вартість проєкту. Важливою частиною управління цим процесом є ідентифікація можливих ризиків та побудова ефективної системи їх мінімізації.

Насамперед, значним ризиком є недостатня якість та повнота даних, які використовуються для побудови моделі. Банківські дані можуть містити прогалини, дублікати чи застарілу інформацію, що знижує точність прогнозів. Для зниження цього ризику доцільно запровадити попередній етап ретельного очищення даних, використання методів імпутації пропусків та контроль якості даних за допомогою внутрішніх аудиторських процедур.[29]

Другим суттєвим фактором є ризик неправильної вибірки та дисбалансу класів, що особливо актуально у випадках кредитних дефолтів, коли частка неповернень є значно меншою за кількість успішних кредитів. У цьому контексті застосування методів oversampling (наприклад, SMOTE) та зниження розмірності даних через PCA дає змогу підвищити репрезентативність вибірки. Саме ці методи було використано у побудові оптимальної моделі Support Vector Machine, яка показала найвищий коефіцієнт F1.

Не менш важливим є ризик технічних збоїв та проблем інтеграції моделі у наявні бізнес-процеси банку. Нові програмні рішення можуть конфліктувати з існуючими інформаційними системами, що призводить до затримок у роботі. Мінімізувати такі загрози можливо завдяки поетапному впровадженню за методологією Kanban, коли кожне технічне завдання проходить через стадії «заплановано», «у роботі» та «готово». Це забезпечує прозорість процесу та дозволяє швидко реагувати на виявлені проблеми.

Додатковим фактором ризику є можливий опір персоналу щодо використання нових моделей. Працівники банку, зокрема кредитні менеджери, можуть не довіряти результатам автоматизованих систем. Для зменшення цього ризику варто організувати навчання та воркшопи, які допоможуть пояснити переваги моделі та підвищити рівень довіри до неї.

Суттєву загрозу становить і ризик інформаційної безпеки. Використання великих масивів персональних даних клієнтів вимагає дотримання суворих стандартів конфіденційності та кіберзахисту. Для його мінімізації застосовуються методи шифрування даних, багаторівнева автентифікація доступу та регулярне проведення аудитів безпеки.

Останньою важливою групою ризиків є ризики управлінського та фінансового характеру. Недооцінка строків і бюджету може призвести до перевитрат та затримки впровадження. Тут ефективними стають практики гнучкого управління проєктами, коли бюджетування здійснюється поетапно, а кожен етап супроводжується моніторингом досягнутих результатів і переглядом планів.

Таким чином, комплексна робота з ризиками передбачає їх ідентифікацію, аналіз і запровадження запобіжних заходів ще на етапі планування. Це дозволяє підвищити надійність впровадження моделі, мінімізувати ймовірність негативних наслідків та забезпечити ефективну інтеграцію розробленого рішення у банківську практику.[30]

Таблиця 3.2

Реєстр ризиків у процесі впровадження моделі прогнозування кредитних ризиків

Категорія ризику	**Опис ризику**	**Потенційні наслідки**	**Стратегії мінімізації**
Якість даних	Використання неповних, зашумлених або застарілих клієнтських даних	Зниження точності моделі, викривлення результатів прогнозування	Попереднє очищення та стандартизація даних, імпутація пропусків, регулярні аудити якості
Дисбаланс вибірки	Нерівномірне співвідношення між випадками дефолтів і нормальних виплат	Недостатнє навчання моделі на рідкісних класах, хибна класифікація	Використання методів oversampling (SMOTE), undersampling, застосування PCA для підвищення стійкості
Технічні збої та інтеграція	Несумісність моделі з наявними IT-системами, перевантаження серверів	Затримки в роботі систем, зниження продуктивності банку	Поетапна інтеграція за Kanban, пілотне тестування, резервні серверні потужності

Організаційний опір	Недовіра співробітників до результатів моделі, страх втрати робочих функцій	Відмова персоналу використовувати нові інструменти, низький рівень прийняття	Навчання персоналу, воркшопи, поступове впровадження підходу «людина + машина»
Інформаційна безпека	Ризик витоку або несанкціонованого доступу до персональних даних клієнтів	Фінансові втрати, репутаційні ризики, юридичні санкції	Шифрування даних, багаторівнева автентифікація, сегментація доступу, регулярні тести безпеки
Фінансові та управлінські ризики	Недооцінка строків і витрат проєкту, перевищення бюджету	Перевитрати, затримки, зниження довіри керівництва	Agile-підхід з поетапним бюджетуванням, діаграми Ґанта для контролю строків, резервний бюджет

3.5. Етапи інтеграції моделі у бізнес-процеси банку

Процес інтеграції моделі прогнозування кредитних ризиків на основі Support Vector Machine з компонентами PCA та SMOTE oversampling у бізнес-процеси умовного банку представляє собою комплексну багатоетапну ініціативу, що потребує ретельного планування та координації між технічними та бізнес-підрозділами фінансової установи.

3.5.1. Підготовчий етап та аналіз поточного стану

На початковому етапі інтеграції критично важливим є проведення всебічного аудиту існуючих бізнес-процесів кредитування банку. Цей процес передбачає детальне картування поточних робочих потоків кредитного конвеєра, починаючи від первинної заявки клієнта до прийняття остаточного рішення про видачу кредиту. Дослідження виявило, що традиційні процеси оцінки кредитних ризиків у банківській сфері характеризуються значною залежністю від

експертних оцінок кредитних аналітиків, що створює потенційні точки неефективності та суб'єктивності в прийнятті рішень.

На цьому етапі формується міжфункціональна проєктна команда, до складу якої входять представники IT-департаменту, ризик-менеджменту, кредитного підрозділу та відділу комплаєнсу. Використання методології канбан на даному етапі дозволяє візуалізувати потік завдань через створення дошки з колонками "Заплановано", "Виконується", "На перевірці" та "Завершено", що забезпечує прозорість процесу та можливість швидкого реагування на зміни в пріоритетах.

3.5.2. Етап технічного планування та архітектурного проектування

Другий етап характеризується розробкою детальної технічної специфікації для інтеграції SVM-моделі в існуючу IT-архітектуру банку. Цей процес включає аналіз сумісності з наявними системами управління кредитними заявками, базами даних клієнтської інформації та системами звітності. Особливу увагу приділяється забезпеченню масштабованості рішення, оскільки обрана модель повинна обробляти значні обсяги транзакцій в режимі реального часу.

Використання принципів канбан на цьому етапі передбачає встановлення WIP (Work In Progress) лімітів для кожного типу завдань, що запобігає перевантаженню команди та забезпечує фокусування на якісному виконанні поточних задач. Кожна задача проходить через етапи аналізу вимог, технічного проектування, розробки прототипу та валідації з боку стейкхолдерів.

3.5.3. Етап розробки інтеграційного шару

Третій етап полягає у створенні програмного інтеграційного шару, який забезпечує взаємодію між оптимізованою SVM-моделлю та корпоративними системами банку. Цей компонент відповідає за обробку вхідних даних про кредитні заявки, їх попередню обробку відповідно до вимог моделі (включаючи застосування PCA для зменшення розмірності та SMOTE для балансування класів при необхідності), передачу до моделі та інтерпретацію результатів прогнозування.

На цьому етапі канбан-методологія демонструє особливу ефективність через можливість швидкої адаптації до змін у вимогах, які часто виникають під час тестування інтеграційних компонентів. Візуальне представлення прогресу дозволяє менеджменту проєкту оперативно ідентифікувати потенційні вузькі місця та перерозподіляти ресурси для забезпечення своєчасного виконання критичних завдань.

3.5.4. Етап пілотного впровадження та тестування

Четвертий етап передбачає поступове впровадження моделі в обмеженому масштабі з метою валідації її ефективності в реальних умовах банківського середовища. Пробне впровадження здійснюється на обмеженому сегменті кредитних продуктів або географічному регіоні, що дозволяє мінімізувати ризики та забезпечити можливість швидкого коригування в разі виявлення проблем.

Цей етап характеризується інтенсивним моніторингом ключових показників ефективності моделі, включаючи точність прогнозування, швидкість обробки заявок та вплив на загальну рентабельність кредитного портфеля. Використання канбан-дошки для відстеження процесу тестування дозволяє команді систематично обробляти виявлені проблеми та забезпечувати їх своєчасне вирішення.

3.5.5. Етап повномасштабного розгортання

П'ятий етап представляє собою розширення використання моделі на всі релевантні бізнес-процеси банку. Цей процес потребує координації між множинними департментами та забезпечення безперервності операційної діяльності під час переходу від старих до нових процедур оцінки кредитних ризиків.

3.5.6. Етап оптимізації та безперервного вдосконалення

Заключний етап фокусується на постійному моніторингу ефективності інтегрованої моделі та її адаптації до змінних умов ринкового середовища. Це

включає регулярне переналаштування параметрів SVM, оновлення компонентів PCA та SMOTE відповідно до нових даних, а також корегування бізнес-правил на основі накопичених показників ефективності.

Канбан-методологія на цьому етапі трансформується в інструмент управління безперервним вдосконаленням, де кожна ініціатива з оптимізації розглядається як окремий робочий елемент з власним життєвим циклом та метриками успішності.

3.6. Моніторинг ефективності та адаптація моделі після запуску

Моніторинг ефективності інтегрованої SVM-моделі з компонентами PCA та SMOTE oversampling представляє собою критично важливий процес, що забезпечує стабільність та надійність системи прогнозування кредитних ризиків у динамічному банківському середовищі. Цей процес характеризується багатовимірністю та потребує систематичного підходу до відстеження ключових показників ефективності.

Основою системи моніторингу є комплексний набір метрик, що охоплює як технічні, так і бізнес-аспекти функціонування моделі. Технічні метрики включають коефіцієнт F1, який виявився найвищим для обраної SVM-конфігурації під час первинного порівняння, а також показники точності (precision), повноти (recall) та площі під ROC-кривою (AUC-ROC). Важливо зазначити, що ці метрики розраховуються не лише для загального набору даних, але й для окремих сегментів кредитного портфеля, оскільки ефективність моделі може варіюватися залежно від типу позичальника, суми кредиту або географічного регіону.[31][14]

Бізнес-метрики фокусуються на фінансових результатах застосування моделі та включають зменшення рівня дефолтів, оптимізацію кредитного портфеля та зростання прибутковості від кредитних операцій. Критично важливим є моніторинг показника економічної ефективності моделі, який визначається як різниця між доходами від правильно схвалених кредитів та втратами від неправильно класифікованих заявок.

Одним з найбільш критичних аспектів моніторингу є раннє виявлення дрейфу даних (data drift), який може суттєво вплинути на ефективність моделі прогнозування кредитних ризиків. Дрейф даних виникає внаслідок змін у характеристиках вхідних даних порівняно з тренувальним набором, що може призвести до деградації прогнозної здатності моделі.[29][30]

Для виявлення дрейфу категоріальних змінних застосовуються статистичні тести, такі як хі-квадрат тест та точний тест Фішера, які дозволяють ідентифікувати статистично значущі зміни в розподілі категорій. Для числових змінних використовуються тест Колмогорова-Смірнова та тест Кузнецова-Уілкоксона, які порівнюють розподіли поточних даних з еталонними.

Особливу увагу приділяється моніторингу компонентів PCA, оскільки зміни в структурі кореляцій між вхідними змінними можуть призвести до зміщення головних компонент та, відповідно, до деградації якості моделі. Для цього розробляється система автоматичного порівняння матриць завантажень (loading matrices) та власних значень (eigenvalues) поточних даних з еталонними показниками.

Система моніторингу включає компонент реального часу для виявлення змін у вхідних даних моделі або поведінкових патернах позичальників, що сигналізують про потенційну нестабільність або дрейф [31]. Автоматизована система алертів працює на основі заздалегідь визначених порогових значень для кожної метрики та використовує багаторівневу схему ескалації.

Перший рівень попереджень активується при незначних відхиленнях від очікуваних показників та передбачає автоматичне сповіщення технічної команди для проведення поглибленого аналізу. Другий рівень ескалації включає залучення бізнес-аналітиків та ризик-менеджерів для оцінки потенційного впливу на кредитну політику банку. Критичний рівень попереджень може призвести до тимчасового відключення автоматичного прийняття рішень та переведення системи у режим асистованого прийняття рішень до вирішення виявлених проблем.

Процес адаптації моделі базується на результатах систематичного моніторингу та включає кілька рівнів втручання залежно від масштабу виявлених проблем. Мікро-адаптації передбачають коригування порогових значень класифікації без зміни архітектури моделі та здійснюються автоматично на основі поточної ефективності.

Макро-адаптації включають перетренування моделі на оновлених даних з урахуванням виявлених змін у поведінці позичальників або ринкових умовах. Цей процес передбачає повторне застосування процедури SMOTE oversampling для балансування класів у новому тренувальному наборі та перерахунок компонентів PCA для оптимального представлення оновлених даних.

Кожна ітерація адаптації моделі супроводжується комплексною процедурою валідації, що включає тестування на історичних даних та тестування на майбутніх даних на контрольованому сегменті поточних кредитних заявок. Особлива увага приділяється забезпеченню того, що покращення одних показників не призводить до погіршення інших критично важливих метрик.

Процес валідації також включає стрес-тестування адаптованої моделі в умовах екстремальних ринкових сценаріїв для забезпечення її стабільності в кризових ситуаціях. Це особливо важливо для банківського сектору, де моделі повинні демонструвати надійність навіть в умовах значної економічної невизначеності.

Всі процеси моніторингу та адаптації детально документуються для забезпечення відповідності регуляторним вимогам та внутрішнім стандартам ризик-менеджменту. Регулярні звіти включають аналіз трендів ключових метрик, опис виконаних адаптацій та їх впливу на загальну ефективність системи.

Використання канбан-методології для управління процесами моніторингу дозволяє візуалізувати потік завдань від виявлення потенційних проблем до їх вирішення та верифікації результатів. Це забезпечує прозорість процесу та можливість швидкого реагування на критичні ситуації, що можуть вплинути на стабільність кредитних операцій банку.

3.7. Рекомендації для практичного застосування

Практичне застосування моделі прогнозування кредитних ризиків на основі Support Vector Machine з компонентами PCA та SMOTE oversampling у реальному банківському середовищі потребує врахування численних факторів, які виходять за межі суто технічних аспектів машинного навчання. Розроблені рекомендації базуються на аналізі кращих практик індустрії та специфічних особливостях банківського регулювання.

Найбільш ефективним підходом до практичного застосування моделі є поетапне впровадження з поступовим розширенням сфери застосування. Рекомендується розпочинати з пробного проєкту, що охоплює обмежений сегмент кредитного портфеля, такий як споживчі кредити на суми до певного ліміту або кредитування фізичних осіб у конкретному географічному регіоні. Такий підхід дозволяє мінімізувати операційні ризики та забезпечити можливість швидкого коригування параметрів моделі на основі реальних результатів.

Перший етап впровадження повинен тривати не менше трьох місяців для накопичення достатнього обсягу даних для валідації ефективності моделі. Протягом цього періоду рекомендується використовувати паралельну систему прийняття рішень, де традиційні методи оцінки кредитних ризиків працюють поряд з новою моделлю, що дозволяє порівняти їх ефективність та ідентифікувати потенційні проблеми.

Критично важливою є забезпечення бездоганної інтеграції моделі з наявним робочим потоком кредитного підрозділу банку. Рекомендується розробити гнучкий інтерфейс, який дозволяє кредитним менеджерам легко інтерпретувати результати моделі та приймати обґрунтовані рішення. Інтерфейс повинен надавати не лише фінальну рекомендацію щодо схвалення або відхилення кредитної заявки, але й детальне розбиття факторів, які найбільше впливають на прогноз.

Особливу увагу слід приділити інтеграції з системами CRM та основними банківськими системами. Модель повинна автоматично отримувати необхідні дані про клієнта з різних джерел та консолідувати їх у форматі, придатному для обробки алгоритмом SVM. При цьому критично важливо забезпечити контроль якості даних для запобігання потраплянню некоректних або неповних даних до моделі.

Ефективність SVM-моделі з PCA та SMOTE значною мірою залежить від якості вхідних даних, тому рекомендується впровадити комплексну систему управління якістю даних. Ця система повинна включати автоматичні перевірки на наявність пропущених значень, викидів та аномальних патернів у даних клієнтів.

Для компоненту PCA рекомендується регулярно переглядати кількість головних компонент, що зберігаються для аналізу, оскільки структура кореляцій між змінними може змінюватися з часом під впливом ринкових умов. Оптимальна кількість компонент повинна пояснювати не менше 85-90% дисперсії вихідних даних, при цьому забезпечуючи значне зменшення розмірності для підвищення ефективності обчислень.

Щодо застосування методу SMOTE, рекомендується здійснювати цю процедуру виключно на тренувальному наборі даних, уникаючи її застосування до валідаційних або тестових вибірок. Параметри SMOTE, такі як кількість найближчих сусідів та ступінь оверсеймплінгу, повинні визначатися на основі специфічних характеристик кредитного портфеля банку та регулярно переглядатися.

Практичне застосування моделі неможливе без надійної системи моніторингу її ефективності в реальному часі. Рекомендується створити дашборд, який відображає ключові метрики ефективності моделі, включаючи коефіцієнт F1, точність прогнозів по різним сегментам позичальників та економічну ефективність прийнятих рішень.

Особливо важливим є моніторинг індексу стабільності, який вимірює ступінь зміни розподілу результатів моделі порівняно з базовим періодом.

Значення індексу стабільності вище 0,25 сигналізує про значний дрейф у характеристиках позичальників та потребу в переналаштуванні моделі.

У контексті банківського регулювання критично важливою є здатність моделі надавати пояснення для прийнятих рішень. Хоча SVM-моделі традиційно вважаються менш інтерпретованими порівняно з деревами рішень, рекомендується використовувати техніки post-hoc пояснюваності, такі як SHAP (SHapley Additive exPlanations) або LIME (Local Interpretable Model-agnostic Explanations).[51]

Ці інструменти дозволяють ідентифікувати найбільш впливові фактори для кожного конкретного рішення моделі, що є критично важливим для відповідності вимогам справедливого кредитування та захисту прав споживачів. Рекомендується документувати логіку прийняття рішень для кожної кредитної заявки та забезпечити можливість її аудиту регуляторними органами.

Практичне застосування автоматизованої системи прогнозування кредитних ризиків створює нові операційні ризики, які потребують проактивного управління. Рекомендується розробити комплексний план відновлення після збоїв, який включає процедури швидкого переключення на резервні системи у разі технічних проблем з основною моделлю.

Критично важливим є встановлення чітких лімітів на автономне прийняття рішень моделлю. Рекомендується, щоб автоматичне схвалення застосовувалося лише до заявок з високим рівнем впевненості моделі та в межах заздалегідь визначених параметрів ризику. Заявки з проміжними результатами або незвичайними характеристиками повинні направлятися на додатковий розгляд кредитними аналітиками.

Для забезпечення довгострокової ефективності моделі рекомендується запровадити систему безперервного навчання, яка дозволяє адаптувати параметри моделі на основі нових даних без повного перетренування. Це особливо важливо в умовах швидко мінливого економічного середовища, коли поведінкові патерни позичальників можуть істотно змінюватися.

Рекомендується проводити повне перетренування моделі щонайменше раз на квартал з використанням найновіших доступних даних. При цьому критично важливим є збереження історії всіх версій моделі та можливість швидкого відкату до попередньої версії у разі виявлення проблем з новою.

Використання agile-методологій, зокрема канбан, для управління процесами вдосконалення моделі дозволяє забезпечити гнучкість та швидкість реагування на зміни в бізнес-вимогах. Кожна ініціатива з удосконалення повинна розглядатися як окреме робоче завдання з чітко визначеними критеріями успішності та часу на виконання.

3.8. Висновки

Проведене дослідження процесу впровадження моделі прогнозування кредитних ризиків на базі Support Vector Machine з компонентами PCA та SMOTE oversampling у банківському середовищі дозволяє сформулювати ряд ключових висновків щодо ефективності та практичної доцільності такого рішення.

Результати експериментального дослідження переконливо демонструють технічну перевагу інтегрованого підходу, що поєднує SVM-алгоритм з методами зменшення розмірності та балансування вибірки. Застосування PCA забезпечує не лише зниження обчислювальної складності моделі через зменшення кількості ознак, але й підвищує її стійкість до шуму в даних, що є критично важливим в умовах банківського середовища з його високими вимогами до надійності прогнозів.

Використання SMOTE oversampling виявилося особливо ефективним для вирішення проблеми дисбалансу класів, характерної для кредитних даних, де частка дефолтних позичальників зазвичай становить незначний відсоток від загальної вибірки. Синтетичне генерування прикладів менш представленого класу дозволило підвищити чутливість моделі до виявлення потенційно проблемних позичальників, що має прямий позитивний вплив на фінансові результати банку.

Застосування канбан-підходу для управління процесом впровадження продемонструвало свою ефективність в умовах багатодисциплінарного банківського проєкту. Візуалізація робочих процесів через канбан-дошку забезпечила прозорість виконання завдань та дозволила оперативно ідентифікувати вузькі місця в процесі розробки. Особливо цінною виявилася можливість швидкого перерозподілу ресурсів між завданнями відповідно до зміни пріоритетів, що є типовим для динамічного банківського середовища.

Поєднання канбан з традиційними інструментами планування, такими як діаграми Ґанта, створило збалансовану систему управління проєктом, що забезпечує як гнучкість у реагуванні на зміни, так і контроль за дотриманням ключових етапів проєкту.

Аналіз реєстру ризиків та практичних рекомендацій виявив декілька критичних факторів, що визначають успішність впровадження моделі. Першочерговим є забезпечення високої якості вхідних даних, оскільки навіть найсучасніші алгоритми машинного навчання не можуть компенсувати систематичні проблеми з якістю даних. Рекомендована система контролю якості даних, що включає автоматичні перевірки та регулярні аудити, є необхідною умовою стабільної роботи моделі.

Другим критичним фактором є організаційна готовність банку до впровадження інновацій. Подолання опору персоналу та забезпечення його залучення до процесу через навчання та воркшопи виявляється не менш важливим, ніж технічна досконалість самої моделі. Стратегія поступового впровадження з паралельним функціонуванням традиційних та інноваційних процесів дозволяє мінімізувати організаційні ризики.

Дорожня карта проєкту з орієнтовною тривалістю 18-22 тижні та структурованим розподілом ресурсів демонструє реалістичність практичного впровадження. Поетапний підхід від підготовчого етапу до фінального пілотування забезпечує контрольовані витрати та можливість корегування стратегії на основі проміжних результатів.

Особливо важливим є врахування не лише прямих витрат на розробку та впровадження, але й непрямих ефектів, таких як підвищення швидкості прийняття рішень, зменшення операційних ризиків та покращення клієнтського досвіду. Ці фактори можуть мати довгострокові позитивні наслідки для конкурентоспроможності банку.

Успішне впровадження моделі прогнозування кредитних ризиків створює основу для подальшого розвитку аналітичних можливостей банку. Накопичений досвід роботи з моделями машинного навчання, налагоджені процеси управління якістю даних та сформована експертиза в області практичного застосування алгоритмів можуть бути використані для вирішення інших бізнес-завдань банку.

Зокрема, перспективними напрямами є розширення моделі для роботи з різними типами кредитних продуктів, впровадження систем скорингу в реальному часі та інтеграція з системами управління взаємовідносинами з клієнтами для персоналізації кредитних пропозицій.

Проведене дослідження окреслює кілька перспективних напрямів для подальших наукових досліджень. По-перше, доцільним є дослідження ефективності ансамблевих методів, що поєднують SVM з іншими алгоритмами машинного навчання для подальшого підвищення точності прогнозування. По-друге, актуальним є вивчення можливостей застосування підходів глибокого навчання для обробки неструктурованих даних, таких як текстові коментарі кредитних аналітиків або соціальні медіа-дані позичальників.

Загалом, результати дослідження підтверджують практичну цінність та економічну доцільність впровадження розглянутого підходу до прогнозування кредитних ризиків у банківському середовищі, при умові дотримання рекомендованих принципів управління проєктом та врахування специфічних особливостей банківської діяльності.

ВИСНОВКИ

Проведене комплексне дослідження застосування методів машинного навчання для прогнозування кредитних ризиків у фінансовій сфері продемонструвало як теоретичну обґрунтованість, так і практичну доцільність впровадження сучасних аналітичних підходів у банківську діяльність. Результати роботи підтверджують, що машинне навчання може суттєво підвищити ефективність управління кредитними ризиками при умові системного та виваженого підходу до впровадження.

Теоретична частина дослідження виявила еволюційний характер розвитку методів прогнозування кредитних ризиків. Традиційні статистичні підходи, зокрема логістична регресія, зберігають свою актуальність завдяки прозорості та відповідності регуляторним вимогам, однак їхні можливості обмежені при роботі зі складними нелінійними залежностями та великими обсягами різнорідних даних. Аналіз літературних джерел показав, що зарубіжні дослідження характеризуються ширшим спектром використаних методів та більшою емпіричною базою, тоді як вітчизняні роботи демонструють обережний підхід до впровадження нових технологій з акцентом на прозорості моделей.

Емпіричне дослідження на наборі даних кредитних карток з Тайваню розкрило складність та багатогранність процесу створення ефективних прогнозних моделей. Критичною виявилася роль якісної підготовки даних, включно з очищенням від некоректних значень та обробкою дисбалансу класів. Застосування методу головних компонент дозволило зменшити розмірність простору ознак удвічі, зберігши понад 99% дисперсії, що підтверджує наявність значної надмірності в початкових даних.

Проблема дисбалансу класів, характерна для кредитних даних де дефолтні позичальники становлять лише 22,1% вибірки, потребувала спеціальних методів вирішення. Порівняння підходів до балансування показало парадоксальний результат: моделі на незбалансованих даних демонстрували вищу загальну

точність, проте гіршу здатність виявляти ризикових клієнтів. Це підкреслює важливість вибору метрик оцінювання відповідно до бізнес-цілей.

Порівняльний аналіз чотирьох алгоритмів машинного навчання виявив суттєві відмінності в їх поведінці та ефективності. Метод опорних векторів з SMOTE-балансуванням показав найкращі результати з F1-показником 0,525, демонструючи ефективність ядрових методів для роботи з нелінійно роздільними даними. Random Forest забезпечив другий за якістю результат (F1 = 0,497) з додатковою перевагою інтерпретованості через ранжування важливості ознак. Логістична регресія зберегла актуальність завдяки стабільності та прозорості, хоча показала обмежену здатність до виявлення складних залежностей.

Дослідження показало відсутність універсального «найкращого» алгоритму для всіх сценаріїв. Вибір методу має базуватися на комплексному врахуванні специфіки даних, бізнес-цілей та операційних обмежень. Деревоподібні методи виявилися природно стійкими до дисбалансу класів та забезпечили цінну інформацію про важливість ознак, виявивши ключову роль історії платежів та кредитних лімітів у прогнозуванні дефолтів.

Практична частина дослідження продемонструвала реалістичність впровадження розробленого підходу в банківському середовищі. Застосування канбан-методології для управління проєктом забезпечило ефективну координацію міждисциплінарної команди та гнучкість у реагуванні на зміни. Розроблена дорожня карта з тривалістю 18-22 тижні та структурованим розподілом ресурсів підтверджує практичну здійсненність впровадження при контрольованих витратах.

Аналіз ризиків та розробка відповідних заходів мінімізації ризиків виявили критичні фактори успіху. Першочерговим є забезпечення високої якості вхідних даних через систему автоматичних перевірок та регулярних аудитів. Не менш важливою є організаційна готовність до впровадження інновацій, включно з подоланням опору персоналу через навчання та залучення до процесу розробки.

Економічна ефективність впровадження виявляється не лише через прямі фінансові результати, а й через непрямі ефекти: підвищення швидкості прийняття рішень, зменшення операційних ризиків та покращення клієнтського досвіду. Ці фактори створюють довгострокові конкурентні переваги для банку.

Побудована фінансова модель показала, що підвищення чутливості системи до ризикових клієнтів дозволяє банку виявляти близько 1 866 додаткових випадків потенційного дефолту. За умови застосування превентивних заходів, здатних зменшити очікувані втрати на 50%, економія на скороченні кредитних збитків може становити близько 559 тис. доларів США. Додатково враховано операційну економію, пов’язану зі зниженням навантаження на співробітників завдяки автоматизації процесів ухвалення рішень. Скорочення ручної обробки заявок на 20% дає змогу зекономити ще близько 200 тис. доларів. Навіть з урахуванням витрат на впровадження моделі, оцінених у 150 тис. доларів, чистий економічний ефект перевищує 600 тис. доларів, демонструючи високий рівень рентабельності інвестицій.

Результати дослідження мають важливі теоретичні та практичні імплікації для розвитку фінансової аналітики. На теоретичному рівні робота розширює розуміння можливостей та обмежень різних алгоритмів машинного навчання в контексті кредитного скорингу. Практичне значення полягає в демонстрації шляхів успішного впровадження технологій машинного навчання з урахуванням специфіки банківського середовища.

Дослідження також окреслює перспективні напрями подальшого розвитку. Актуальним є вивчення ансамблевих методів, що поєднують переваги різних алгоритмів, розробка систем реального часу з автоматичним переналаштуванням при зміні умов, та інтеграція альтернативних джерел даних для підвищення точності прогнозування. Перспективним також є застосування глибинного навчання для обробки неструктурованих даних, що може відкрити нові можливості для аналізу кредитних ризиків.

Успішне впровадження створює основу для розширення аналітичних можливостей банку на інші сфери діяльності. Накопичений досвід, налагоджені

процеси управління якістю даних та сформована експертиза можуть бути використані для персоналізації кредитних пропозицій, управління портфельними ризиками та оптимізації інших бізнес-процесів.

У підсумку дослідження підтверджує, що машинне навчання має значний потенціал для підвищення ефективності управління кредитними ризиками. Водночас реалізація цього потенціалу вимагає комплексного підходу, що поєднує технічну досконалість з глибоким розумінням бізнес-процесів, регуляторних вимог та організаційних особливостей банківської діяльності. Лише така інтегрована парадигма може забезпечити успішне впровадження інноваційних методів прогнозування та створення довгострокових конкурентних переваг у фінансовому секторі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mahesh B. Machine learning algorithms-a review // International Journal of Science and Research (IJSR). 2020. Vol. 9, No. 1. P. 381–386.
2. Borrelli S., Ianneli A. та ін. Machine learning-driven credit risk: a systemic review. URL: https://link.springer.com/article/10.1007/s00521-022-07472-2 (Дата звернення: 24.08.2025).
3. MDPI. Credit Risk Prediction Using Machine Learning and Deep Learning (огляд/дослідження). URL: https://www.mdpi.com/2227-9091/12/11/174 (Дата звернення: 24.08.2025).
4. ScienceDirect. Interpretable machine learning for imbalanced credit scoring datasets. URL: https://www.sciencedirect.com/science/article/pii/S0377221723005088 (Дата звернення: 24.08.2025).
5. European Central Bank. Revised Guide to Internal Models (ECB – guide to internal models, 2025 update). URL: https://www.bankingsupervision.europa.eu/ecb/pub/pdf/ssm.supervisory_guide202507.en.pdf (Дата звернення: 24.08.2025).
6. A Data-Driven Machine Learning Framework Proposal for Selecting Project Management Methods // Applied Sciences. URL: https://www.mdpi.com/2076-3417/15/13/7263 (Дата звернення: 24.08.2025).
7. Crook J., Edelman D., Thomas L. Recent developments in Consumer Credit Risk assessment // European Journal of Operational Research. 2007. Vol. 183. P. 1447–1465. DOI: 10.1016/j.ejor.2006.09.100. URL: https://www.researchgate.net/publication/222829893_Recent_developments_in_Consumer_Credit_Risk_assessment
8. Baesens B., Van Gestel T., Viaene S., Stepanova M., Suykens J., Vanthienen J. Benchmarking state-of-the-art classification algorithms for credit scoring // Journal of the Operational Research Society. 2003. Vol. 54. DOI: 10.1057/palgrave.jors.2601545. URL: https://www.researchgate.net/publication/32029800_Benchmarking_state-of-the-art_classification_algorithms_for_credit_scoring
9. Національний банк України. Економетрична модель НБУ для оцінки кредитного ризику // Журнал НБУ. 2015. URL: https://journal.bank.gov.ua/ua/article/2015/234/03 (Дата звернення: 24.08.2025).
10. Johnson B. Credit Scoring Models 101: Types and Examples for a Stronger Financial Future. 2023. URL: https://www.highradius.com/resources/Blog/credit-scoring-models-types-andexamples/ (Дата звернення: 24.08.2025).
11. Investopedia. Is Credit Scoring? Purpose, Factors, and Role In Lending. 2023. URL: https://www.investopedia.com/terms/c/credit_scoring.asp (Дата звернення: 24.08.2025).

12. Datsko M., Druschak N. Credit scoring modelling using machine learning methods // Visnyk of the Lviv University. Series Economics. 2022. Vol. 63. P. 56–66. DOI: http://dx.doi.org/10.30970/ves.2022.63.0.6305
13. Liashenko O., Kravets T., Kostovetskyi Y. Machine Learning and Data Balancing Methods for Bankruptcy Prediction // Ekonomika. 2023. Vol. 102(2). P. 28–46. DOI: 10.15388/
14. Google Developers. Classification: ROC Curve and AUC. URL: https://developers.google.com/machine-learning/crash-course/classification/rocand-auc (Дата звернення: 24.08.2025).
15. Ляшенко О. І., Кравець Т. В., Мордатенко О. К. Моделювання кредитного ризику з використанням гібридної моделі RNN-LSTM для цифрових фінансових установ // Ефективна економіка. 2025. DOI: http://doi.org/10.32702/2307-2105.2025.4.26
16. Scikit-learn. Machine learning in Python. URL: https://scikit-learn.org/stable/ (Дата звернення: 24.08.2025).
17. Imbalanced-learn documentation. URL: https://imbalanced-learn.org/stable/ (Дата звернення: 24.08.2025).
18. D. Dua and C. Graff, UCI Machine Learning Repository: default of credit card clients Data Set [https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients], Irvine, CA: University of California, School of Information and Computer Science, 2019.
19. T. M. Alam et al., "An Investigation of Credit Card Default Prediction in the Imbalanced Datasets," in IEEE Access, vol. 8, pp. 201173-201198, 2020. https://doi.org/10.1109/ACCESS.2020.3033784
20. Show-Jane Yen, Yue-Shi Lee, "Cluster-based under-sampling approaches for imbalanced data distributions", Expert Systems with Applications, Volume 36, Issue 3, Part 1, 2009, Pages 5718-5727, ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2008.06.108
21. G. Lemaitre, et al., "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning", 2016. https://arxiv.org/abs/1609.06570
22. N. V. Chawla, et al., "SMOTE: synthetic minority over-sampling technique", Journal of artificial intelligence research, 321-357, 2002. https://arxiv.org/abs/1106.1813
23. F. Last, et al., "Oversampling for Imbalanced Learning Based on K-Means and SMOTE", Information Sciences, Volume 465, 2018. https://arxiv.org/abs/1711.00837
24. Ron Kohavi, «A study of cross-validation and bootstrap for accuracy estimation and model selection», IJCAI'95, Volume 2, 1137–1143, 1995. https://dl.acm.org/doi/10.5555/1643031.1643047
25. N. Japkowicz, «Assessment metrics for imbalanced learning», Imbalanced learning, 187-206, 2013. https://doi.org/10.1002/9781118646106.ch8
26. D.P. Kroese, et al., “Data Science and Machine Learning: Mathematical and

Statistical Methods", Chapman & Hall, 2019. https://people.smp.uq.edu.au/DirkKroese/DSML/
27. F. Pedregosa, et al., «Scikit-learn: Machine Learning in Python», Journal of Machine Learning Research, Volume 12, 2825-2830, 2011. https://scikit-learn.org/stable/
28. I-Cheng Yeh, Che-hui Lien, «The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients», Expert Systems with Applications, Volume 36, Issue 2, Part 1, 2009, Pages 2473-2480, ISSN 0957-4174. https://doi.org/10.1016/j.eswa.2007.12.020
29. Brij Bhushan Singh. "Solving Data Drift Issues in Credit Risk Models: A Practical Guide with Real-World AI and ML Examples". URL: https://medium.com/@mjprub/solving-data-drift-issues-in-credit-risk-models-a-practical-guide-with-real-world-ai-and-ml-3f253fff87f2 (Дата звернення: 21.09.2025)
30. Harry Shi. "Model Degradation in Credit Risk". URL: https://2os.medium.com/model-degradation-309d96295f7f (Дата звернення: 21.09.2025)
31. A Guide to Model Monitoring: 8 Essential Steps to Get Started. URL: https://www.citrusx.ai/post/a-guide-to-model-monitoring (Дата звернення: 21.09.2025)
32. Akash Takyar. "How to build credit risk models using machine learning?". URL: https://www.leewayhertz.com/build-credit-risk-models-using-machine-learning/ (Дата звернення: 21.09.2025)
33. Marcos R. Machado, Daniel Tianfu Chen, Joerg R. Osterrieder, "An analytical approach to credit risk assessment using machine learning models", Decision Analytics Journal, Volume 16, 2025, 100605, ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2025.100605. URL: https://www.sciencedirect.com/science/article/pii/S277266222500061X (Дата звернення: 21.09.2025)
34. Kruppa, J., Schwarz, A., Arminger, G., & Ziegler, A. (2013). Consumer credit risk: Individual probability estimates using machine learning. Expert Systems with Applications, 40(13), 5125-5131. https://doi.org/10.1016/j.eswa.2013.03.019
35. Bellotti, T., & Crook, J. (2009). Support vector machines for credit scoring and discovery of significant features. Expert Systems with Applications, 36(2), 3302-3308. https://doi.org/10.1016/j.eswa.2008.01.005
36. Kvamme, H., Sellereite, N., Aas, K., & Sjursen, S. (2018). Predicting mortgage default using convolutional neural networks. Expert Systems with Applications, 102, 207-217. https://doi.org/10.1016/j.eswa.2018.02.036
37. Zhu, L., Qiu, D., Ergu, D., Ying, C., & Liu, K. (2019). A study on predicting loan default based on the random forest algorithm. Procedia Computer Science, 162, 503-513. https://doi.org/10.1016/j.procs.2019.12.017
38. Wang, G., Hao, J., Ma, J., & Jiang, H. (2011). A comparative assessment of ensemble learning for credit scoring. Expert Systems with Applications, 38(1),

223-230. https://doi.org/10.1016/j.eswa.2010.06.048
39. Paleologo, G., Elisseeff, A., & Antonini, G. (2010). Subagging for credit scoring models. European Journal of Operational Research, 201(2), 490-499. https://doi.org/10.1016/j.ejor.2009.03.008
40. Louzada, F., Ara, A., & Fernandes, G. B. (2016). Classification methods applied to credit scoring: Systematic review and overall comparison. Surveys in Operations Research and Management Science, 21(2), 117-134. https://doi.org/10.1016/j.sorms.2016.10.001
41. Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. Expert Systems with Applications, 39(3), 3446-3453. https://doi.org/10.1016/j.eswa.2011.09.033
42. Barocas, S., Hardt, M., & Narayanan, A. (2019). Fairness and Machine Learning. URL: http://www.fairmlbook.org (Дата звернення: 21.09.2025)
43. Molnar, C. (2020). Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. URL: https://christophm.github.io/interpretable-ml-book/ (Дата звернення: 21.09.2025)
44. Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. https://doi.org/10.48550/arXiv.1702.08608
45. Huang, C. L., Chen, M. C., & Wang, C. J. (2007). Credit scoring with a data mining approach based on support vector machines. Expert Systems with Applications, 33(4), 847-856. https://doi.org/10.1016/j.eswa.2006.07.007
46. Zhou, L., Lai, K. K., & Yu, L. (2010). Least squares support vector machines ensemble models for credit scoring. Expert Systems with Applications, 37(1), 127-133. https://doi.org/10.1016/j.eswa.2009.05.024
47. Harris, T. (2015). Credit scoring using the clustered support vector machine. Expert Systems with Applications, 42(2), 741-750. https://doi.org/10.1016/j.eswa.2014.07.021
48. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority oversampling technique. Journal of Artificial Intelligence Research, 16, 321-357. https://doi.org/10.1613/jair.953
49. Douzas, G., & Bação, F. (2018). Effective data generation for imbalanced learning using conditional generative adversarial networks. Expert Systems with Applications, 91, 464-471. https://doi.org/10.1016/j.eswa.2017.09.030
50. Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from Imbalanced Data Sets. Springer. https://doi.org/10.1007/978-3-319-98074-4
51. Gramegna, A., & Giudici, P. (2021). SHAP and LIME: An evaluation of discriminative power in credit risk. Frontiers in Artificial Intelligence, 4, 752558. https://doi.org/10.3389/frai.2021.752558
52. Bussmann, N., Giudici, P., Marinelli, D., & Papenbrock, J. (2021). Explainable machine learning in credit risk management. Computational Economics, 57(1), 203-216. https://doi.org/10.1007/s10614-020-10042-0
53. Maldonado, S., López, J., & Vairetti, C. (2019). An alternative SMOTE oversampling strategy for high-dimensional datasets. Applied Soft Computing,

76, 380-389. https://doi.org/10.1016/j.asoc.2018.12.024
54. Hurley, M., & Adebayo, J. (2016). Credit scoring in the era of big data. Yale Journal of Law and Technology, 18, 148-216. Available at: https://yjolt.org/sites/default/files/18_yale_j.l._tech_148.pdf
55. Bartlett, R., Morse, A., Stanton, R., & Wallace, N. (2022). Consumer-lending discrimination in the fintech era. Journal of Financial Economics, 143(1), 30-56. https://doi.org/10.1016/j.jfineco.2021.05.047
56. Logistic Regression in Machine Learning. URL: https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/ (Дата звернення: 10.10.2025)
57. Taraqur Rahman. Classification – ISLR Series: Chapter 4 – Part I. URL: https://medium.com/kaggle-nyc/classification-islr-series-chapter-4-part-i-20422520bea3 (Дата звернення: 10.10.2025)
58. Μηχανική Μάθηση. Kernel Methods. URL: https://helios.ntua.gr/2021-22/pluginfile.php/107231/mod_folder/content/0/14.%20Kernel%20Methods/14.%20Kernel%20Methods.pdf?forcedownload=1 (Дата звернення: 10.10.2025)
59. Anubha Singh. Cross-Validation Techniques. URL: https://medium.com/analytics-vidhya/cross-validation-techniques-a925782517e8 (Дата звернення: 10.10.2025)
60. Sunita Rawat. Principal Component Analysis (PCA). URL: https://medium.com/@sunita_rawat/principal-component-analysis-pca-a10132f9c78a (Дата звернення: 10.10.2025)

ДОДАТКИ

ДОДАТОК А

Лістинг 1. Підключення python бібліотек необхідних у дослідженні

```
import pandas as pd
import numpy as np

# Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA
from imblearn.under_sampling import ClusterCentroids
from imblearn.over_sampling import KMeansSMOTE, SMOTE

# Classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn import tree
from pydotplus import graph_from_dot_data
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Metrics
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, recall_score, precision_score,
f1_score, auc, precision_recall_curve, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay, PrecisionRecallDisplay

# Visualization
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

%matplotlib inline
sns.set_style('darkgrid')
random_state = 42

import warnings
warnings.filterwarnings("ignore")
```

Лістинг 2. Зчитування та створення датафрейму

```
# read and creation of dataframe
df = pd.read_csv('C:/Users/topto/OneDrive/Рабочий стол/Credit
default/data/UCI_Credit_Card.csv', index_col='ID')
df.rename(columns={'DEFAULT.PAYMENT.NEXT.MONTH':'DEFAULT'}
, inplace=True)
df.rename(columns={'PAY_0': 'PAY_1'}, inplace=True)
df.rename(columns=lambda x: x.upper(), inplace=True)
df.head()
```

Лістинг 3. Перегляд інформації про датафрейм

```
df.info()
```

Лістинг 4. Перегляд описової інформації про фактори датасету

```
df[['LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE']].describe()
df['EDUCATION'].value_counts().sort_index()
df['MARRIAGE'].value_counts().sort_index()
```

Лістинг 5. Очистка безкорисною та неправильної інформації

```
# remove useless and incorrect information
print(f"Dataset size before:\t{df.shape[0]}")
df = df.drop(df[df['MARRIAGE']==0].index)
df = df.drop(df[df['EDUCATION']==0].index)
df = df.drop(df[df['EDUCATION']==5].index)
df = df.drop(df[df['EDUCATION']==6].index)
print(f"Dataset size after:\t{df.shape[0]}")
```

Лістинг 6. Огляд описової інформації про історію попередніх платежів в датасеті

```
df[['PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']].describe()
```

Лістинг 7. Нормалізація даних про історію попередніх платежів

```
pay_features = ['PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']
for p in pay_features:
    df.loc[df[p]<0, p] = -1
    df.loc[df[p]>=0, p] = df.loc[df[p]>=0, p] + 1
    df[p] = df[p].astype('int64')
```

Лістинг 8. Перегляд описової інформації про суму виписки

```
df[['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4',
'BILL_AMT5', 'BILL_AMT6']].describe()
```

Лістинг 9. Перегляд описової інформації про суму попереднього платежу

```
df[['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4',
'PAY_AMT5', 'PAY_AMT6']].describe()
```

Лістинг 10. Огляд кількості дефолтних оплат на наступний місяць

```
df['DEFAULT.PAYMENT.NEXT.MONTH'].value_counts()
```

Лістинг 11. Графік розподілу класів

```
# Class distribution
plt.figure(figsize = (4,4))
graph_target = sns.countplot(x="DEFAULT.PAYMENT.NEXT.MONTH",
data=df, palette=['#77DD76','#FF6962'])

i=0
for p in graph_target.patches:
    height = p.get_height()

    percentage = round(100 *
df["DEFAULT.PAYMENT.NEXT.MONTH"].value_counts()[i] / len(df),2)

    str_plot = f'{percentage} %'

    graph_target.text(p.get_x()+p.get_width()/2., height - 100, str_plot,
ha="center")
    i += 1

plt.title('Class Distribution')
plt.xticks([0,1],['Non-defaulters', 'Defaulters'])
plt.xlabel('Default payment next month', fontsize=12)
plt.ylabel('Number of clients')
plt.tight_layout()
```

Лістинг 12. Графік щільності розподілу даних

```
# Kernel density distribution
sns.set_palette(palette=['#77DD76','#FF6962'])
fig, axs = plt.subplots(1, 2, figsize=(15,4))

class_0 = df.loc[df['DEFAULT.PAYMENT.NEXT.MONTH'] ==
0]['LIMIT_BAL']
class_1 = df.loc[df['DEFAULT.PAYMENT.NEXT.MONTH'] ==
1]['LIMIT_BAL']
axs[0].set_title('Density plot of LIMIT_BAL by default type')
sns.kdeplot(class_1, fill=True, label='Defaulters', color='darkorange',
```

```
ax=axs[0])
sns.kdeplot(class_0, fill=True, label='Non-defaulters', ax=axs[0])
axs[0].axvline(0.14*1e6, 0, ls='--')
axs[0].legend()

class_0 = df.loc[df['DEFAULT.PAYMENT.NEXT.MONTH'] == 0]['AGE']
class_1 = df.loc[df['DEFAULT.PAYMENT.NEXT.MONTH'] == 1]['AGE']
axs[1].set_title('Density plot of AGE by default type')
sns.kdeplot(class_1, fill=True, label='Defaulters', color='darkorange',
ax=axs[1])
sns.kdeplot(class_0, fill=True, label='Non-defaulters', ax=axs[1])
axs[1].axvline(26, 0, ls='--')
axs[1].axvline(40, 0, ls='--')
axs[1].legend()

fig.tight_layout()
plt.show()
```

Лістинг 13. Графік розподілу значень факторів статі, рівня навчання та сімейного статусу

```
sns.set_palette(palette=['#77DD76','#FF6962'])
fig, axs = plt.subplots(1, 3, figsize=(20,4))

axs[0].set_title('SEX distribution')
sns.countplot(x='SEX', hue='DEFAULT.PAYMENT.NEXT.MONTH',
data=df, ax=axs[0])
axs[0].set_xticklabels(['Male', 'Female'])
axs[0].set_xlabel('')
axs[0].legend()

axs[1].set_title('EDUCATION distribution')
sns.countplot(x='EDUCATION',
hue='DEFAULT.PAYMENT.NEXT.MONTH', data=df, ax=axs[1])
axs[1].set_xticklabels(['Graduate School', 'University', 'High School',
'Others'])
axs[1].set_xlabel('')
axs[1].legend()

axs[2].set_title('MARRIAGE status distribution')
sns.countplot(x='MARRIAGE',
hue='DEFAULT.PAYMENT.NEXT.MONTH', data=df, ax=axs[2])
axs[2].set_xticklabels(['Married', 'Single', 'Others'])
axs[2].set_xlabel('')
```

```
axs[2].legend()

fig.tight_layout()
plt.show()
```

Лістинг 14. Графік розподілу оплат

```
PAY_cols = ['PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']
plt.figure(figsize=(15,8))

for i, col in enumerate(PAY_cols):
    plt.subplot(2,3,i + 1)
    ax = sns.countplot(df.loc[:,col], palette = 'muted')
    plt.ylim(0,20000)
    plt.xlabel(col,fontsize=15)
    plt.ylabel('')
    plt.tight_layout()

    for p in ax.patches:
        ax.annotate((p.get_height()), (p.get_x(), p.get_height()+500), fontsize =
11)

plt.show()
```

Лістинг 15. Графік розподілу рахунків

```
BILL_AMT_cols = ['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3',
'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6']
plt.figure(figsize=(12,6))

for i,col in enumerate(BILL_AMT_cols):
    plt.subplot(2,3,i + 1)
    sns.kdeplot(df.loc[(df['DEFAULT'] == 0), col], label = 'No
Default',color='#77DD76', shade=True)
    sns.kdeplot(df.loc[(df['DEFAULT'] == 1), col], label = 'Default',
color='#FF6962', shade=True)
    plt.xlim(-40000,200000)
    plt.ylabel('')
    plt.xlabel(col,fontsize=15)
    plt.legend()
    plt.tight_layout()

plt.show()
```

Лістинг 16. Графік розподілу оплат

```
PAY_AMT_cols = ['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3',
'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']
plt.figure(figsize=(12,6))

for i,col in enumerate(PAY_AMT_cols):
    plt.subplot(2,3,i + 1)
    sns.kdeplot(df.loc[(df['DEFAULT'] == 0), col], label = 'No Default', shade
= True,color='#77DD76')
    sns.kdeplot(df.loc[(df['DEFAULT'] == 1), col], label = 'Default', shade =
True, color='#FF6962')
    plt.xlim(-10000,60000)
    plt.xlabel(col,fontsize=15)
    plt.ylabel('')
    plt.legend()
    plt.tight_layout()

plt.show()
```

Лістинг 17. Графік статусу оплати

```
repayment = df[['PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6',
'DEFAULT']]
repayment = pd.melt(repayment,
                    id_vars="DEFAULT",
                    var_name="Payment Status",
                    value_name='Delay (months)')


plt.figure(figsize=(12,4))
sns.boxplot(y="Delay (months)", x="Payment Status", hue="DEFAULT",
data=repayment, palette=sns.color_palette(('#77DD76','#FF6962')))
plt.xlim([-1.5,5.5])
plt.ylim([-1.5,9.5])
plt.yticks(np.arange(-1,9))
plt.title('PAYMENT STATUS - BOXPLOT')
plt.legend()
plt.box(False)
```

Лістинг 18. Графік кореляційної матриці

```
numeric =
['LIMIT_BAL','AGE','PAY_1','PAY_2','PAY_3','PAY_4','PAY_5','PAY_6','B
ILL_AMT1','BILL_AMT2','BILL_AMT3','BILL_AMT4','BILL_AMT5','BIL
L_AMT6','PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_A
MT5','PAY_AMT6', 'DEFAULT']
corr = df[numeric].corr()

mask = np.triu(np.ones_like(corr, dtype=bool))
f, ax = plt.subplots(figsize=(18,12))
ax = sns.heatmap(corr, mask=mask, vmax=1, vmin=-1, center=0,
square=True, linewidths=.5, cmap='Blues', cbar_kws={'shrink': .5},
annot=True, annot_kws={'size': 10})
cbar = ax.collections[0].colorbar
```

Лістинг 19. Графік діаграми розсіювання найбільш корельованих значень

```
pair_plot =
sns.pairplot(df[['BILL_AMT1','BILL_AMT2','BILL_AMT3','BILL_AMT4','
BILL_AMT5','BILL_AMT6','DEFAULT']], hue='DEFAULT',
diag_kind='kde', corner=True)
pair_plot._legend.remove()
```

Лістинг 20. Перевірка нормальності розподілу даних у датасеті

```
#Normality check
# Check gaussian distribution
sns.set_color_codes('pastel')
fig, axs = plt.subplots(5, 3, figsize=(18,18))

numeric =
['LIMIT_BAL','AGE','BILL_AMT1','BILL_AMT2','BILL_AMT3','BILL_A
MT4','BILL_AMT5','BILL_AMT6','PAY_AMT1','PAY_AMT2','PAY_AMT
3','PAY_AMT4','PAY_AMT5','PAY_AMT6']
i, j = 0, 0
for f in numeric:
    if j == 3:
        j = 0
        i = i+1
    stats.probplot(df[f],
                   dist='norm',
                   sparams=(df[f].mean(), df[f].std()),
                   plot=axs[i,j])
    axs[i,j].get_lines()[0].set_marker('.')
```

```
    axs[i,j].grid()
    axs[i,j].get_lines()[1].set_linewidth(3.0)
    j = j+1

fig.tight_layout()
axs[4,2].set_visible(False)

plt.show()
```

Лістинг 21. Підготовка даних до моделювання - категоріювання даних

```
#Data preprocesing
df['GRAD_SCHOOL'] = (df['EDUCATION'] == 1).astype('category')
df['UNIVERSITY'] = (df['EDUCATION'] == 2).astype('category')
df['HIGH_SCHOOL'] = (df['EDUCATION'] == 3).astype('category')
df.drop('EDUCATION', axis=1, inplace=True)

df['MALE'] = (df['SEX'] == 1).astype('category')
df.drop('SEX', axis=1, inplace=True)

df['MARRIED'] = (df['MARRIAGE'] == 1).astype('category')
df.drop('MARRIAGE', axis=1, inplace=True)

df.head()
```

Лістинг 22. Розділ датасету на тренуваньну та тестову вибірки

```
#Test/traim datasets

y = df['DEFAULT']
X = df.drop('DEFAULT', axis=1, inplace=False)

X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y,
random_state=24, stratify=y)
print('Training set shape: ', np.shape(X_train_raw))
print(f'- Defaulters:\t {len(y_train[y_train==1])}')
print(f'- Non-defaulters: {len(y_train[y_train==0])}')
print('Test set shape: ', np.shape(X_test_raw))
print(f'- Defaulters:\t {len(y_test[y_test==1])}')
print(f'- Non-defaulters: {len(y_test[y_test==0])}')
```

Лістинг 23. Скейлінг фіч - нормалізація

```
#Feature scaling
```

```
scaler = MinMaxScaler()
X_train_norm = X_train_raw.copy()
X_test_norm = X_test_raw.copy()

X_train_norm['LIMIT_BAL'] =
scaler.fit_transform(X_train_raw['LIMIT_BAL'].values.reshape(-1, 1))
X_test_norm['LIMIT_BAL'] =
scaler.transform(X_test_raw['LIMIT_BAL'].values.reshape(-1, 1))
X_train_norm['AGE'] =
scaler.fit_transform(X_train_raw['AGE'].values.reshape(-1, 1))
X_test_norm['AGE'] = scaler.transform(X_test_raw['AGE'].values.reshape(-1,
1))
for i in range(1,7):
    X_train_norm['PAY_' + str(i)] = scaler.fit_transform(X_train_raw['PAY_' +
str(i)].values.reshape(-1, 1))
    X_test_norm['PAY_' + str(i)] = scaler.transform(X_test_raw['PAY_' +
str(i)].values.reshape(-1, 1))
    X_train_norm['BILL_AMT' + str(i)] =
scaler.fit_transform(X_train_raw['BILL_AMT' + str(i)].values.reshape(-1, 1))
    X_test_norm['BILL_AMT' + str(i)] =
scaler.transform(X_test_raw['BILL_AMT' + str(i)].values.reshape(-1, 1))
    X_train_norm['PAY_AMT' + str(i)] =
scaler.fit_transform(X_train_raw['PAY_AMT' + str(i)].values.reshape(-1, 1))
    X_test_norm['PAY_AMT' + str(i)] =
scaler.transform(X_test_raw['PAY_AMT' + str(i)].values.reshape(-1, 1))
```

Лістинг 24. Скейлінг фіч - стандартизація

```
scaler = StandardScaler()
X_train_std = X_train_raw.copy()
X_test_std = X_test_raw.copy()

X_train_std['LIMIT_BAL'] =
scaler.fit_transform(X_train_raw['LIMIT_BAL'].values.reshape(-1, 1))
X_test_std['LIMIT_BAL'] =
scaler.transform(X_test_raw['LIMIT_BAL'].values.reshape(-1, 1))
X_train_std['AGE'] =
scaler.fit_transform(X_train_raw['AGE'].values.reshape(-1, 1))
X_test_std['AGE'] = scaler.transform(X_test_raw['AGE'].values.reshape(-1,
1))
for i in range(1,7):
    X_train_std['PAY_' + str(i)] = scaler.fit_transform(X_train_raw['PAY_' +
str(i)].values.reshape(-1, 1))
    X_test_std['PAY_' + str(i)] = scaler.transform(X_test_raw['PAY_' +
```

```
str(i)].values.reshape(-1, 1))
    X_train_std['BILL_AMT' + str(i)] =
scaler.fit_transform(X_train_raw['BILL_AMT' + str(i)].values.reshape(-1, 1))
    X_test_std['BILL_AMT' + str(i)] =
scaler.transform(X_test_raw['BILL_AMT' + str(i)].values.reshape(-1, 1))
    X_train_std['PAY_AMT' + str(i)] =
scaler.fit_transform(X_train_raw['PAY_AMT' + str(i)].values.reshape(-1, 1))
    X_test_std['PAY_AMT' + str(i)] =
scaler.transform(X_test_raw['PAY_AMT' + str(i)].values.reshape(-1, 1))
```

Лістинг 25. Порівняння розкиду даних до та після скейлінгу

```
sns.set_color_codes('deep')
numeric =
['LIMIT_BAL','AGE','BILL_AMT1','BILL_AMT2','BILL_AMT3','BILL_A
MT4','BILL_AMT5','BILL_AMT6','PAY_AMT1','PAY_AMT2','PAY_AMT
3','PAY_AMT4','PAY_AMT5','PAY_AMT6']
fig, axs = plt.subplots(1, 2, figsize=(24,6))

sns.boxplot(data=X_train_norm[numeric], ax=axs[0])
axs[0].set_title('Boxplot of normalized numeric features')
axs[0].set_xticklabels(labels=numeric, rotation=25)
axs[0].set_xlabel(' ')

sns.boxplot(data=X_train_std[numeric], ax=axs[1])
axs[1].set_title('Boxplot of standardized numeric features')
axs[1].set_xticklabels(labels=numeric, rotation=25)
axs[1].set_xlabel(' ')

fig.tight_layout()
plt.show()
```

Лістинг 26. Графік часток варіації даних, що пояснюються головними факторами

```
#Explained variance

pc = len(X_train_norm.columns.values) #number columns = 25
pca = PCA(n_components=pc)
pca.fit(X_train_norm)

sns.reset_orig()
sns.set_color_codes('pastel')
plt.figure(figsize = (8,4))
plt.grid()
```

```
plt.title('Explained Variance of Principal Components')
plt.plot(pca.explained_variance_ratio_, marker='o')
plt.plot(np.cumsum(pca.explained_variance_ratio_), marker='o')
plt.legend(["Individual Explained Variance", "Cumulative Explained
Variance"])
plt.xlabel('Principal Component Indexes')
plt.ylabel('Explained Variance Ratio')
plt.tight_layout()
plt.axvline(12, 0, ls='--')
plt.show()
```

Лістинг 27. Виведення відсотковості вибірки яку пояснює кожна із 15 компонент

```
cumsum = np.cumsum(pca.explained_variance_ratio_)
indexes = ['PC' + str(i) for i in range(1, pc+1)]
cumsum_df = pd.DataFrame(data=cumsum, index=indexes, columns=['var1'])
cumsum_df['var2'] = pd.Series([round(val, 4) for val in cumsum_df['var1']],
index = cumsum_df.index)
cumsum_df['Cumulative Explained Variance'] =
pd.Series(["{0:.2f}%".format(val * 100) for val in cumsum_df['var2']], index
= cumsum_df.index)
cumsum_df = cumsum_df.drop(['var1','var2'], axis=1, inplace=False)
cumsum_df.T.iloc[:,:15]
```

Лістинг 28. Виведення матриці коваріації тренувального датасету після підбору 12 найважливіших факторів

```
pc = 12
pca = PCA(n_components=pc)
pca.fit(X_train_norm)
X_train = pd.DataFrame(pca.transform(X_train_norm))
X_test = pd.DataFrame(pca.transform(X_test_norm))
X_train.columns = ['PC' + str(i) for i in range(1, pc+1)]
X_test.columns = ['PC' + str(i) for i in range(1, pc+1)]
X_train.head()
```

Лістинг 29. Пропорції розподілу тренувального датасету

```
#Balancing data

class_count = [y_train[y_train == 0].count(), y_train[y_train == 1].count()]
class_count_df = pd.DataFrame(data=class_count, index=['Non-defaulters',
'Defaulters'], columns=['Number'])
class_count_df['var2'] =
pd.Series([round(val/class_count_df['Number'].sum(axis=0), 4) for val in
```

```
class_count_df['Number']], index = class_count_df.index)
class_count_df['Percentage'] = pd.Series(["{0:.2f}%".format(val * 100) for val
in class_count_df['var2']], index = class_count_df.index)
class_count_df = class_count_df.drop(['var2'], axis=1, inplace=False)
print('Training set class proportion')
class_count_df
```

Лістинг 30. Балансування даних - Андерсеймплінг

```
#Undersampling

oversample = ClusterCentroids(random_state=24)
X_train_cc, y_train_cc = oversample.fit_resample(X_train, y_train)

class_count = [y_train_cc[y_train_cc == 0].count(), y_train_cc[y_train_cc ==
1].count()]
class_count_df = pd.DataFrame(data=class_count, index=['Non-defaulters',
'Defaulters'], columns=['Number'])
class_count_df['var2'] =
pd.Series([round(val/class_count_df['Number'].sum(axis=0), 4) for val in
class_count_df['Number']], index = class_count_df.index)
class_count_df['Percentage'] = pd.Series(["{0:.2f}%".format(val * 100) for val
in class_count_df['var2']], index = class_count_df.index)
class_count_df = class_count_df.drop(['var2'], axis=1, inplace=False)
print('Training set class proportion after Cluster Centroid Undersampling')
class_count_df

# non-defaulters 4954
# defaulters 4954
```

Лістинг 31. Балансування даних - SMOTE

```
#SMOTE

oversample = SMOTE(random_state=24)
X_train_smote, y_train_smote = oversample.fit_resample(X_train, y_train)

class_count = [y_train_smote[y_train_smote == 0].count(),
y_train_smote[y_train_smote == 1].count()]
class_count_df = pd.DataFrame(data=class_count, index=['Non-defaulters',
'Defaulters'], columns=['Number'])
class_count_df['var2'] =
pd.Series([round(val/class_count_df['Number'].sum(axis=0), 4) for val in
class_count_df['Number']], index = class_count_df.index)
```

```
class_count_df['Percentage'] = pd.Series(["{0:.2f}%".format(val * 100) for val
in class_count_df['var2']], index = class_count_df.index)
class_count_df = class_count_df.drop(['var2'], axis=1, inplace=False)
print('Training set class proportion after SMOTE Oversampling')
class_count_df

# non-defaulters 17246
# defaulters 17246
```

Лістинг 32. Балансування даних - K-means SMOTE

```
#K-means SMOTE



oversample = KMeansSMOTE(cluster_balance_threshold=0.00001,
random_state=24)
X_train_ksmote, y_train_ksmote = oversample.fit_resample(X_train, y_train)

class_count = [y_train_ksmote[y_train_ksmote == 0].count(),
y_train_ksmote[y_train_ksmote == 1].count()]
class_count_df = pd.DataFrame(data=class_count, index=['Non-defaulters',
'Defaulters'], columns=['Number'])
class_count_df['var2'] =
pd.Series([round(val/class_count_df['Number'].sum(axis=0), 4) for val in
class_count_df['Number']], index = class_count_df.index)
class_count_df['Percentage'] = pd.Series(["{0:.2f}%".format(val * 100) for val
in class_count_df['var2']], index = class_count_df.index)
class_count_df = class_count_df.drop(['var2'], axis=1, inplace=False)
print('Training set class proportion after K-means SMOTE Oversampling')
class_count_df
```

Лістинг 33. Графік порівняння результатів балансування

```
y_list = [y_train, y_train_cc, y_train_smote, y_train_ksmote]
y_list_labels = ['No resampling', 'Cluster Centroids', 'SMOTE', 'K-means
SMOTE']

plt.figure(figsize=(14,4))

i = 1
for value, label in zip(y_list,y_list_labels):
    plt.subplot(1,4,i)
    graph_target = sns.countplot(value, palette=['#77DD76','#FF6962'])
    plt.xlabel(label, fontdict= {"size":15})
```

```
    plt.ylabel('')
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)

    j=0
    for p in graph_target.patches:
        height = p.get_height()

        str_plot = f'{round(100 * value.value_counts()[j]/len(value),2)} %'

        graph_target.text(p.get_x()+p.get_width()/2., height + 100 , str_plot,
ha="center", fontsize = 13)
        j += 1

    i += 1
    graph_target.set_ylim(0, 20000)

plt.tight_layout()
plt.show()
```

Лістинг 34. Побудова логістичної регресії

```
#LOGISTIC REGRESSION

# Plots a barchart with the F1 results and a confusion matrix on the best
estimator with the best preprocessing technique
def plot_summary(result, resampling_names, y_pred, y_test):

    fig = plt.figure(constrained_layout=False,figsize=(14,4))
    gs1 = fig.add_gridspec(nrows=1, ncols=6, left=0.1, right=0.90, wspace=0.5,
hspace=0.5)
    ax1 = fig.add_subplot(gs1[0,0:3])
    ax2 = fig.add_subplot(gs1[0,3:5])

    plot_x = []
    plot_y = []

    for i in range(len(result)):
        plot_x.append(result[i])
        plot_y.append(resampling_names[i])

    temp_df = pd.DataFrame({'x':plot_x, 'y':plot_y}).sort_values(['x'],
ascending=False)
```

```
    # plot barchart
    sns.barplot(data=temp_df, x='x',y='y',
                palette='Greens_r',
                ax=ax1)
    ax1.set_xlabel("F1-score", fontsize="14")
    ax1.set_ylabel("")
    ax1.set_yticklabels(temp_df.y,fontsize=13)

    # confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot = True,
                cmap=plt.cm.Greens,
                annot_kws={"size": 14},
                linecolor = 'w',
                linewidth = 4,ax=ax2,
                fmt='d'
               )

    ax2.set_xlabel("Predicted labels",fontsize="14")
    ax2.set_ylabel("True labels",fontsize="14")

    return plt


def plot_test_scores(model, oversample, plot):

    if oversample == 'raw':
        X_test_s = X_test_norm.copy()
    else:
        X_test_s = X_test.copy()

    # predict
    y_pred = model.predict(X_test_s)

    # various tests
    test_acc = accuracy_score(y_test, y_pred)
    test_precision = precision_score(y_test, y_pred)
    test_recall = recall_score(y_test, y_pred)
    precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
    test_auc = auc(recall, precision)
    test_f1 = f1_score(y_test, y_pred)

    #print on console
```

```
    print("Test Scores:")
    print(f"- Accuracy:\t{test_acc}")
    print(f"- Recall:\t{test_recall}")
    print(f"- Precision:\t{test_precision}")
    print(f"- F1-score:\t{test_f1}")
    print(f"- AUC:\t\t{test_auc}")

    #plot results
    if plot:
        fig, axs = plt.subplots(1, 3, figsize=(15,4))

        tmp = pd.DataFrame({'Feature': X_test_s.columns,
                            'Feature importance':
model.best_estimator_.feature_importances_})
        tmp = tmp.sort_values(by='Feature importance', ascending=False)
        s = sns.barplot(x='Feature', y='Feature importance', data=tmp, ax=axs[0])
        s.set_xticklabels(s.get_xticklabels(), rotation=90)

        plot_confusion_matrix(model, X_test_s, y_test, cmap=plt.cm.Greens,
normalize='true', ax=axs[1])
        no_skill = len(y_test[y_test==1]) / len(y_test)
        plot_precision_recall_curve(model, X_test_s, y_test, ax=axs[2])
        plt.axhline(no_skill, 0, ls='--', label='No Skill')
        plt.ylim([-0.05, 1.05])
        axs[2].legend(loc = 'upper right')
        plt.show()
    else:
        fig, axs = plt.subplots(1, 2, figsize=(10,4))
        plot_confusion_matrix(model, X_test_s, y_test, cmap=plt.cm.Greens,
normalize='true', ax=axs[0])
        no_skill = len(y_test[y_test==1]) / len(y_test)
        plot_precision_recall_curve(model, X_test_s, y_test, ax=axs[1])
        plt.axhline(no_skill, 0, ls='--', label='No Skill')
        plt.ylim([-0.05, 1.05])
        axs[1].legend(loc = 'upper right')
        plt.show()

    return test_acc, test_recall, test_precision, test_f1, test_auc


def pipeline(model, params, oversample=None, plot=False):

    max_f1 = 0
```

```
y_pred_max = 0
y_test_max = 0

if oversample:
    if oversample == 'raw': #without PCA
        X_train_s = X_train_norm.copy() #original data scaled
        y_train_s = y_train.copy()
        X_test_f1 = X_test_raw.copy()
    else:
        X_train_s, y_train_s = oversample.fit_resample(X_train, y_train)
        X_test_f1 = X_test.copy()
else:
    X_train_s = X_train.copy() #with PCA without resampling
    y_train_s = y_train.copy()
    X_test_f1 = X_test.copy()

#gridsearch best parameter
gridsearch = GridSearchCV(estimator = model,
                  param_grid = params,
                  scoring = 'f1',
                  cv = 5,
                  n_jobs = -1,
                  verbose = True)
gridsearch.fit(X_train_s, y_train_s)
print(f"Best parameters:\t{gridsearch.best_params_}")
print(f"Best validation score:\t{gridsearch.best_score_}")

test_scores = plot_test_scores(gridsearch, oversample, plot)

best_estimator = gridsearch.best_estimator_

# save best config
y_pred = best_estimator.predict(X_test_f1)
f1 = round(f1_score(y_test, y_pred),2)

if f1 > max_f1:
    y_pred_max = y_pred
    y_test_max = y_test
    max_f1 = f1

return test_scores, y_pred_max, y_test_max
```

```
oversample_method = ['raw',
                     None,
                     SMOTE(random_state=24),
                     KMeansSMOTE(cluster_balance_threshold=1e-5,
random_state=24),
                     ClusterCentroids(random_state=24)]
oversample_names = ['Raw data',
                    'PCA',
                    'PCA + SMOTE oversampling',
                    'PCA + KMeansSMOTE oversampling',
                    'PCA + ClusterCentroids oversampling']
scores_lr = {'Accuracy' : [float for i in range(len(oversample_method))],
             'Recall' : [float for i in range(len(oversample_method))],
             'Precision' : [float for i in range(len(oversample_method))],
             'F1-score' : [float for i in range(len(oversample_method))],
             'AUC' : [float for i in range(len(oversample_method))],}



params_lr = {'C': [1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2]}
for idx, oversample in enumerate(oversample_method):
    print(f"Logistic Regression with {oversample_names[idx]}")
    scores, y_pred_max, y_test_max =
pipeline(LogisticRegression(random_state=24, max_iter=1000, n_jobs=-1),
                                        params_lr,
                                        oversample)
    scores_lr['Accuracy'][idx] = scores[0]
    scores_lr['Recall'][idx] = scores[1]
    scores_lr['Precision'][idx] = scores[2]
    scores_lr['F1-score'][idx] = scores[3]
    scores_lr['AUC'][idx] = scores[4]

scores_lr_df = pd.DataFrame(data=scores_lr, index=oversample_names)
plot = plot_summary(scores_lr_df["F1-score"], oversample_names,
y_pred_max, y_test_max)

scores_lr_df
```

Лістинг 35. Побудова моделі SVM

```
#SUPPORT VECTOR MACHINES (SVM)
```

```
oversample_method = ['raw',
                     None,
                     SMOTE(random_state=24),
                     KMeansSMOTE(cluster_balance_threshold=1e-5,
random_state=24),
                     ClusterCentroids(random_state=24)]
oversample_names = ['Raw data',
                    'PCA',
                    'PCA + SMOTE oversampling',
                    'PCA + KMeansSMOTE oversampling',
                    'PCA + ClusterCentroids oversampling']
scores_svm = {'Accuracy' : [float for i in range(len(oversample_method))],
              'Recall' : [float for i in range(len(oversample_method))],
              'Precision' : [float for i in range(len(oversample_method))],
              'F1-score' : [float for i in range(len(oversample_method))],
              'AUC' : [float for i in range(len(oversample_method))],}



params_svm = {'C': [1e-1, 1e0, 1e1, 1e2],
              'kernel': ['rbf', 'poly'],
              'gamma': [1e-1, 'scale']}
for idx, oversample in enumerate(oversample_method):
    print(f"Support Vector Machine with {oversample_names[idx]}")
    scores, y_pred_max, y_test_max = pipeline(SVC(),
                        params_svm,
                        oversample)
    scores_svm['Accuracy'][idx] = scores[0]
    scores_svm['Recall'][idx] = scores[1]
    scores_svm['Precision'][idx] = scores[2]
    scores_svm['F1-score'][idx] = scores[3]
    scores_svm['AUC'][idx] = scores[4]

scores_svm_df = pd.DataFrame(data=scores_svm, index=oversample_names)
plot = plot_summary(scores_svm_df["F1-score"], oversample_names,
y_pred_max, y_test_max)
scores_svm_df
```

Лістинг 36. Побудова моделі за допомогою дерев рішень

```
#Decision trees


clf = DecisionTreeClassifier(max_depth=3,
```

```
                    criterion='entropy',
                    max_features=None)
clf.fit(X_train_raw, y_train)
plt.figure(figsize=(24,12))
tree.plot_tree(decision_tree=clf,
          feature_names=X_train_raw.columns,
          class_names=['Non-Default','Default'],
          filled=True, proportion=True)
plt.show()




def plot_tree_test_scores(model, oversample, plot):

    if oversample == 'raw':
        X_test_s = X_test_raw.copy()
    else:
        X_test_s = X_test.copy()

    # predict
    y_pred = model.predict(X_test_s)

    # various tests
    test_acc = accuracy_score(y_test, y_pred)
    test_precision = precision_score(y_test, y_pred)
    test_recall = recall_score(y_test, y_pred)
    precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
    test_auc = auc(recall, precision)
    test_f1 = f1_score(y_test, y_pred)

    #print on console
    print("Test Scores:")
    print(f"- Accuracy:\t{test_acc}")
    print(f"- Recall:\t{test_recall}")
    print(f"- Precision:\t{test_precision}")
    print(f"- F1-score:\t{test_f1}")
    print(f"- AUC:\t\t{test_auc}")

    #plot results
    if plot:
        fig, axs = plt.subplots(1, 3, figsize=(15,4))
```

```
        tmp = pd.DataFrame({'Feature': X_test_s.columns,
                            'Feature importance':
model.best_estimator_.feature_importances_})
        tmp = tmp.sort_values(by='Feature importance', ascending=False)
        s = sns.barplot(x='Feature', y='Feature importance', data=tmp, ax=axs[0])
        s.set_xticklabels(s.get_xticklabels(), rotation=90)

        plot_confusion_matrix(model, X_test_s, y_test, cmap=plt.cm.Greens,
normalize='true', ax=axs[1])
        no_skill = len(y_test[y_test==1]) / len(y_test)
        plot_precision_recall_curve(model, X_test_s, y_test, ax=axs[2])
        plt.axhline(no_skill, 0, ls='--', label='No Skill')
        plt.ylim([-0.05, 1.05])
        axs[2].legend(loc = 'upper right')
        plt.show()
    else:
        fig, axs = plt.subplots(1, 2, figsize=(10,4))
        plot_confusion_matrix(model, X_test_s, y_test, cmap=plt.cm.Greens,
normalize='true', ax=axs[0])
        no_skill = len(y_test[y_test==1]) / len(y_test)
        plot_precision_recall_curve(model, X_test_s, y_test, ax=axs[1])
        plt.axhline(no_skill, 0, ls='--', label='No Skill')
        plt.ylim([-0.05, 1.05])
        axs[1].legend(loc = 'upper right')
        plt.show()

    return test_acc, test_recall, test_precision, test_f1, test_auc

def tree_pipeline(model, params, oversample=None, plot=False):

    max_f1 = 0
    y_pred_max = 0
    y_test_max = 0

    if oversample:
        if oversample == 'raw': #without PCA
            X_train_s = X_train_raw.copy() #original data scaled
            y_train_s = y_train.copy()
            X_test_f1 = X_test_raw.copy()
        else:
            X_train_s, y_train_s = oversample.fit_resample(X_train, y_train)
            X_test_f1 = X_test.copy()
```

```
    else:
        X_train_s = X_train.copy() #with PCA without resampling
        y_train_s = y_train.copy()
        X_test_f1 = X_test.copy()

    #gridsearch best parameter
    gridsearch = GridSearchCV(estimator = model,
                      param_grid = params,
                      scoring = 'f1',
                      cv = 5,
                      n_jobs = -1,
                      verbose = True)
    gridsearch.fit(X_train_s, y_train_s)
    print(f"Best parameters:\t{gridsearch.best_params_}")
    print(f"Best validation score:\t{gridsearch.best_score_}")

    test_scores = plot_tree_test_scores(gridsearch, oversample, plot)

    best_estimator = gridsearch.best_estimator_

    # save best config
    y_pred = best_estimator.predict(X_test_f1)
    f1 = round(f1_score(y_test, y_pred),2)

    if f1 > max_f1:
        y_pred_max = y_pred
        y_test_max = y_test
        max_f1 = f1

    return test_scores, y_pred_max, y_test_max




oversample_method = ['raw',
                     None,
                     SMOTE(random_state=24),
                     KMeansSMOTE(cluster_balance_threshold=1e-5,
random_state=24),
                     ClusterCentroids(random_state=24)]
oversample_names = ['Raw data',
                    'PCA',
                    'PCA + SMOTE oversampling',
```

```
                'PCA + KMeansSMOTE oversampling',
                'PCA + ClusterCentroids oversampling']
scores_tree = {'Accuracy' : [float for i in range(len(oversample_method))],
          'Recall' : [float for i in range(len(oversample_method))],
          'Precision' : [float for i in range(len(oversample_method))],
          'F1-score' : [float for i in range(len(oversample_method))],
          'AUC' : [float for i in range(len(oversample_method))],}



params_tree = {'max_depth': [5, 10, 20, 30, 50]}
for idx, oversample in enumerate(oversample_method):
   print(f"Decision Tree with {oversample_names[idx]}")
   scores, y_pred_max, y_test_max =
tree_pipeline(DecisionTreeClassifier(criterion='entropy', max_features=None,
random_state=24),
                    params_tree,
                    oversample)
   scores_tree['Accuracy'][idx] = scores[0]
   scores_tree['Recall'][idx] = scores[1]
   scores_tree['Precision'][idx] = scores[2]
   scores_tree['F1-score'][idx] = scores[3]
   scores_tree['AUC'][idx] = scores[4]

scores_rf_tree = pd.DataFrame(data=scores_tree, index=oversample_names)
plot = plot_summary(scores_rf_tree["F1-score"], oversample_names,
y_pred_max, y_test_max)
scores_rf_tree
```

Лістинг 37. Побудова моделі за допомогою рандом форесту

```
#RANDOM FOREST

oversample_method = ['raw',
                None,
                SMOTE(random_state=24),
                KMeansSMOTE(cluster_balance_threshold=1e-5,
random_state=24),
                ClusterCentroids(random_state=24)]
oversample_names = ['Raw data',
                'PCA',
                'PCA + SMOTE oversampling',
                'PCA + KMeansSMOTE oversampling',
                'PCA + ClusterCentroids oversampling']
```

```
scores_rf = {'Accuracy' : [float for i in range(len(oversample_method))],
             'Recall' : [float for i in range(len(oversample_method))],
             'Precision' : [float for i in range(len(oversample_method))],
             'F1-score' : [float for i in range(len(oversample_method))],
             'AUC' : [float for i in range(len(oversample_method))],}




params_rf = {'n_estimators': [10, 50, 100, 200],
             'max_features': [None, 'sqrt']}
for idx, oversample in enumerate(oversample_method):
    print(f"Random Forest with {oversample_names[idx]}")
    scores, y_pred_max, y_test_max =
tree_pipeline(RandomForestClassifier(criterion='entropy', random_state=24,
n_jobs=-1),
                          params_rf,
                          oversample,
                          True)
    scores_rf['Accuracy'][idx] = scores[0]
    scores_rf['Recall'][idx] = scores[1]
    scores_rf['Precision'][idx] = scores[2]
    scores_rf['F1-score'][idx] = scores[3]
    scores_rf['AUC'][idx] = scores[4]

scores_rf_df = pd.DataFrame(data=scores_rf, index=oversample_names)
plot = plot_summary(scores_rf_df["F1-score"], oversample_names,
y_pred_max, y_test_max)

scores_rf_df
```

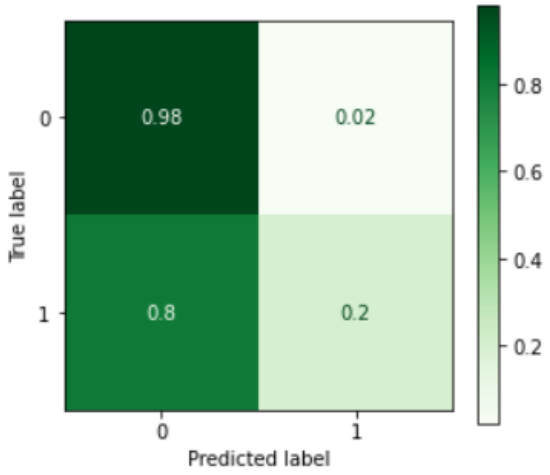
Лістинг 38. Графік порівняння результатів моделювання

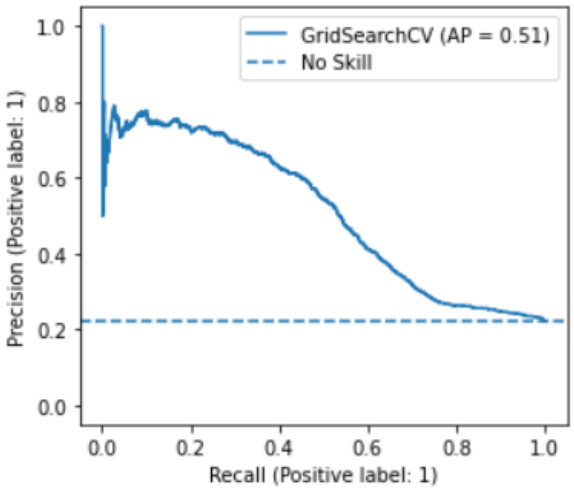
```
#Results

frames = [scores_lr_df, scores_svm_df, scores_rf_tree, scores_rf_df]
scores_df = pd.concat(frames, keys=['Logistic Regression', 'Support Vector
Machine', 'Decision Tree', 'Random Forest'])
scores_df



scores_lr_df['Classifier'] = 'Logistic Regression'
scores_lr_df['Technique'] = scores_lr_df.index
#scores_lr_df['Technique'][scores_lr_df['Technique']=='Normalized data'] =
```

```
'Raw data'
scores_svm_df['Classifier'] = 'SVM'
scores_svm_df['Technique'] = scores_svm_df.index
#scores_svm_df['Technique'][scores_svm_df['Technique']=='Normalized
data'] = 'Raw data'
scores_rf_tree['Classifier'] = 'Decision Tree'
scores_rf_tree['Technique'] = scores_rf_tree.index
scores_rf_df['Classifier'] = 'Random Forest'
scores_rf_df['Technique'] = scores_rf_df.index

frames2 = [scores_lr_df, scores_svm_df, scores_rf_tree, scores_rf_df]
scores = pd.concat(frames2, ignore_index=True, keys=None)

scores.set_index(['Technique', 'Classifier']).unstack(0).plot(kind='bar',
                                                    y='F1-score',
                                                    ylabel='F1-score',
                                                    figsize=(12,6),
                                                    fontsize=12,
                                                    rot=0).legend(bbox_to_anchor=(1.02,
1))
```

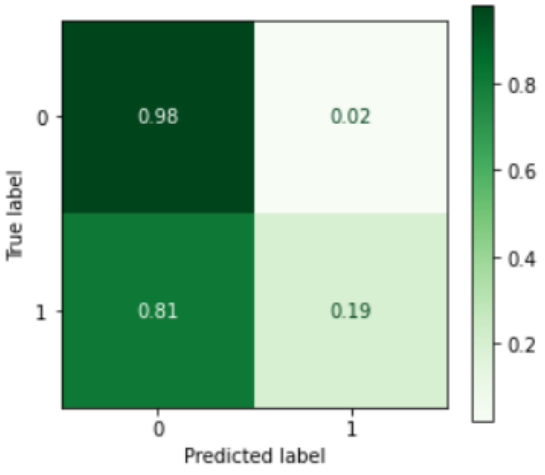
ДОДАТОК Б

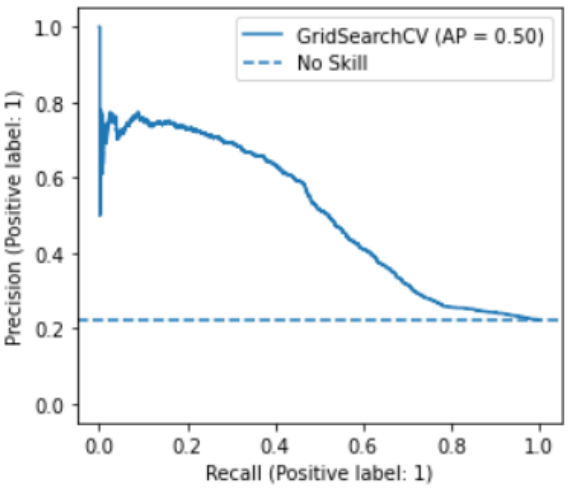
```
Logistic Regression with Raw data
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters:        {'C': 100.0}
Best validation score:  0.27946732816436864
Test Scores:
- Accuracy:     0.8048912309147412
- Recall:       0.19624470018170806
- Precision:    0.7346938775510204
- F1-score:     0.30975143403441685
- AUC:          0.5551193361572708
```





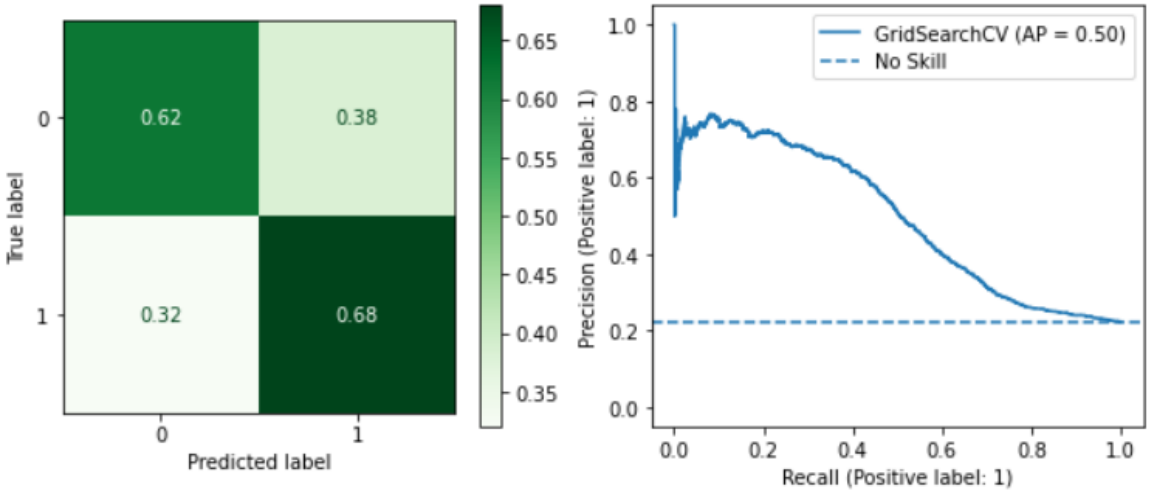

```
Logistic Regression with PCA
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters:        {'C': 100.0}
Best validation score:  0.27924371052935976
Test Scores:
- Accuracy:     0.8042156465342521
- Recall:       0.19200484554815264
- Precision:    0.7337962962962963
- F1-score:     0.3043686989918387
- AUC:          0.5530235272794735
```



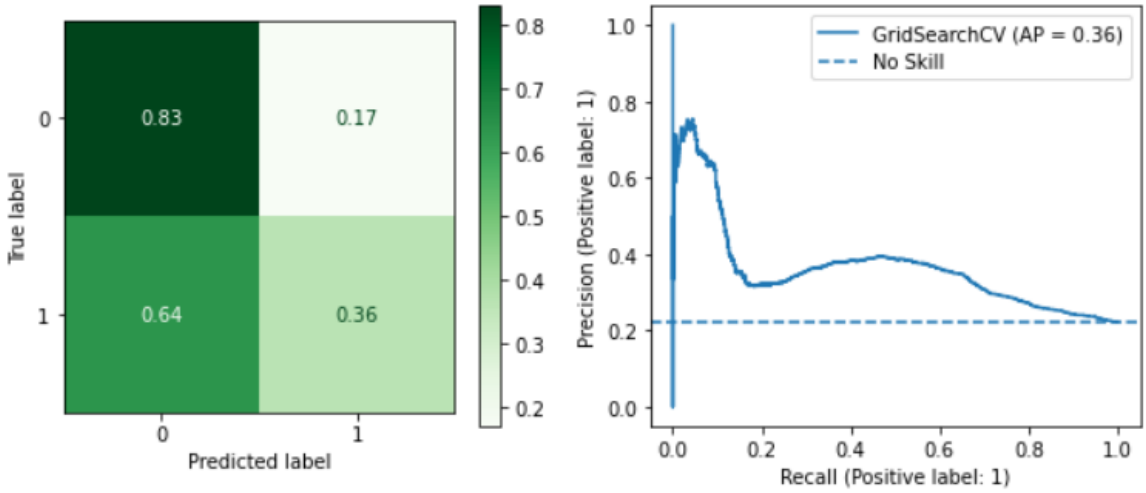


ДОДАТОК Б

```
Logistic Regression with PCA + SMOTE oversampling
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters:        {'C': 1.0}
Best validation score:  0.663015565520895
Test Scores:
- Accuracy:     0.6319416295095257
- Recall:       0.6795881284070261
- Precision:    0.3382574615616521
- F1-score:     0.45169082125603865
- AUC:          0.5446612087122137
```

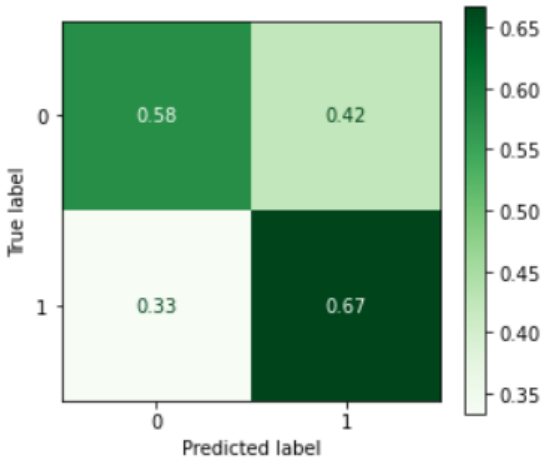


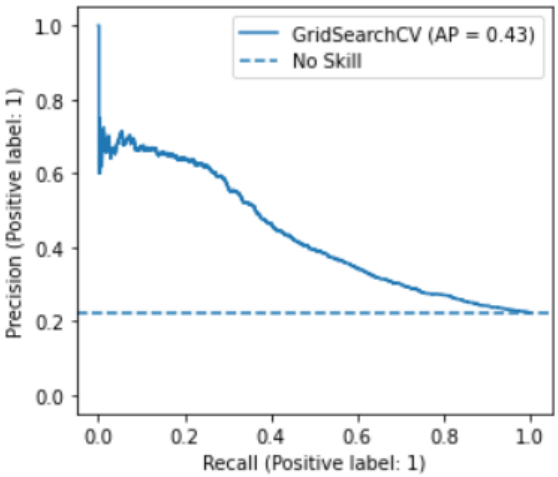

```
Logistic Regression with PCA + KMeansSMOTE oversampling
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters:        {'C': 10.0}
Best validation score:  0.7325059123562209
Test Scores:
- Accuracy:     0.7247669233887313
- Recall:       0.3634161114476075
- Precision:    0.37831021437578816
- F1-score:     0.3707136237256719
- AUC:          0.44186708130110464
```

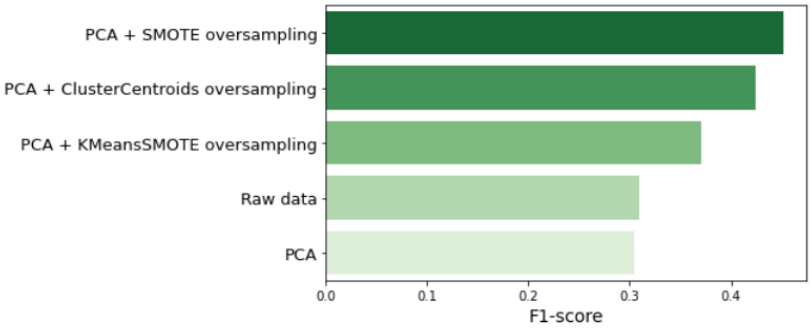


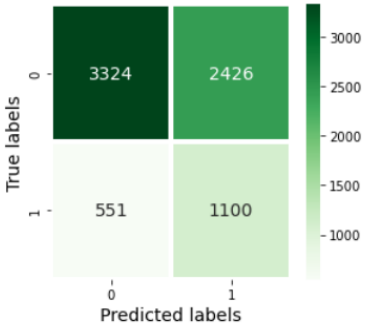
ДОДАТОК Б

```
Logistic Regression with PCA + ClusterCentroids oversampling
Fitting 5 folds for each of 6 candidates, totalling 30 fits
Best parameters:        {'C': 10.0}
Best validation score:  0.6527932511996625
Test Scores:
- Accuracy:     0.5977570598567761
- Recall:       0.6662628709872804
- Precision:    0.31196823596142936
- F1-score:     0.4249565385358316
- AUC:          0.5263402528393056
```



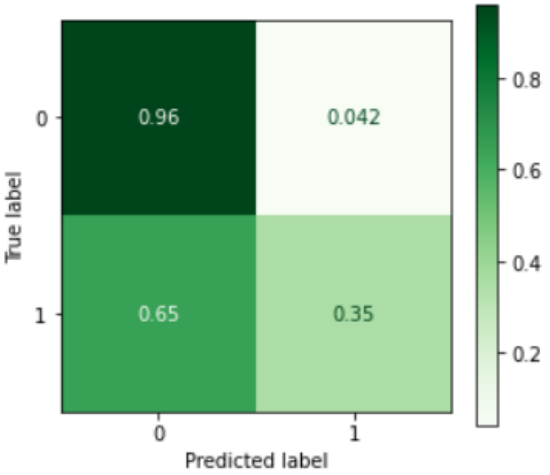


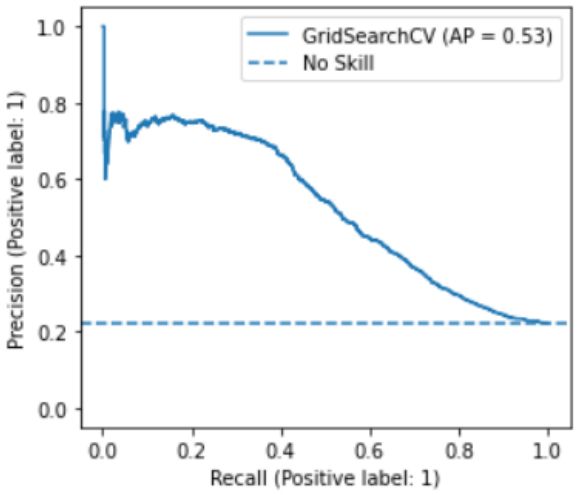




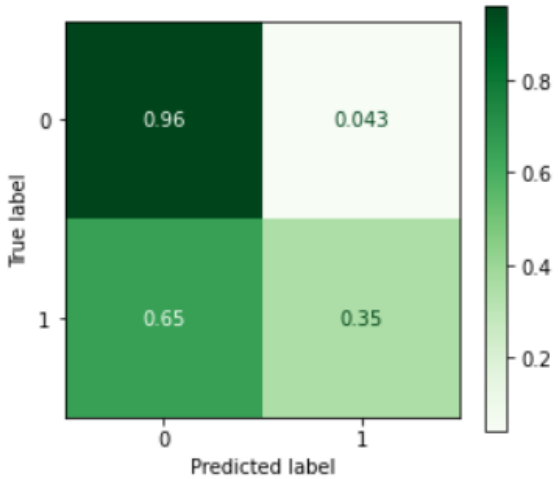
ДОДАТОК В

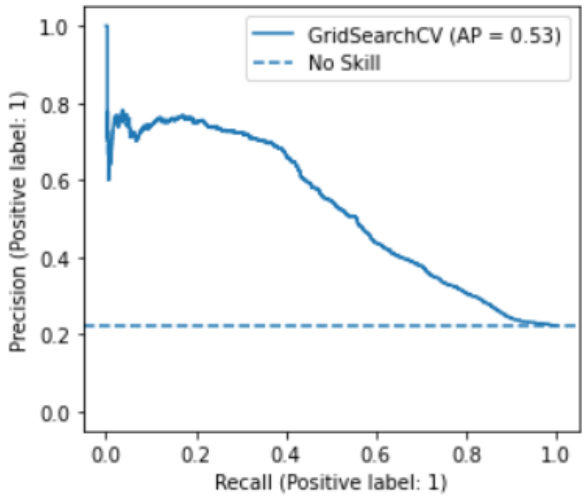
```
Support Vector Machine with Raw data
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters:        {'C': 100.0, 'gamma': 0.1, 'kernel': 'rbf'}
Best validation score:  0.44927163508004886
Test Scores:
- Accuracy:     0.8221861910552628
- Recall:       0.35069654754694124
- Precision:    0.7035236938031592
- F1-score:     0.4680679062247373
- AUC:          0.5995327662634842
```





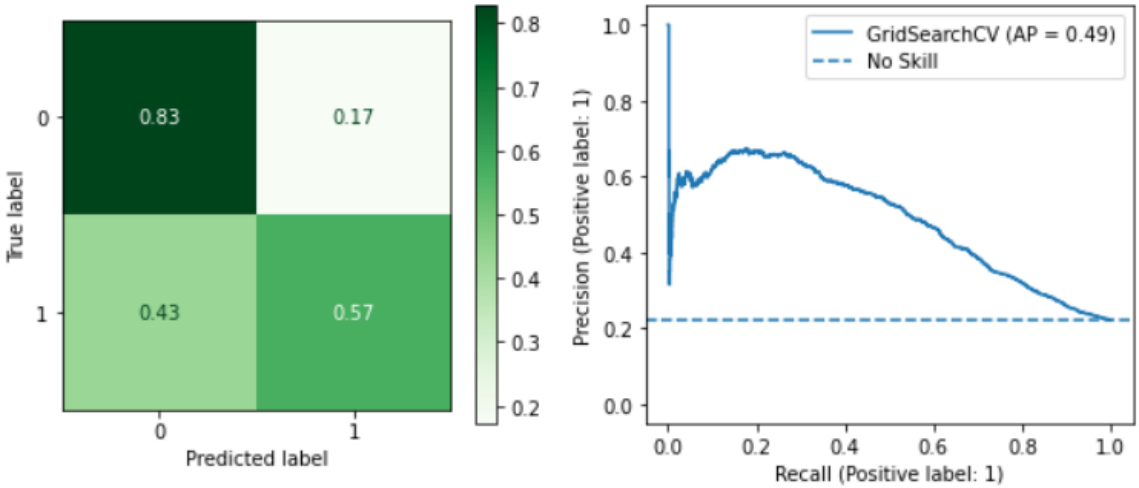

```
Support Vector Machine with PCA
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters:        {'C': 100.0, 'gamma': 0.1, 'kernel': 'rbf'}
Best validation score:  0.452377656956216
Test Scores:
- Accuracy:     0.8216457235508715
- Recall:       0.34948516050878253
- Precision:    0.701093560145808
- F1-score:     0.4664510913500404
- AUC:          0.5978471227918272
```



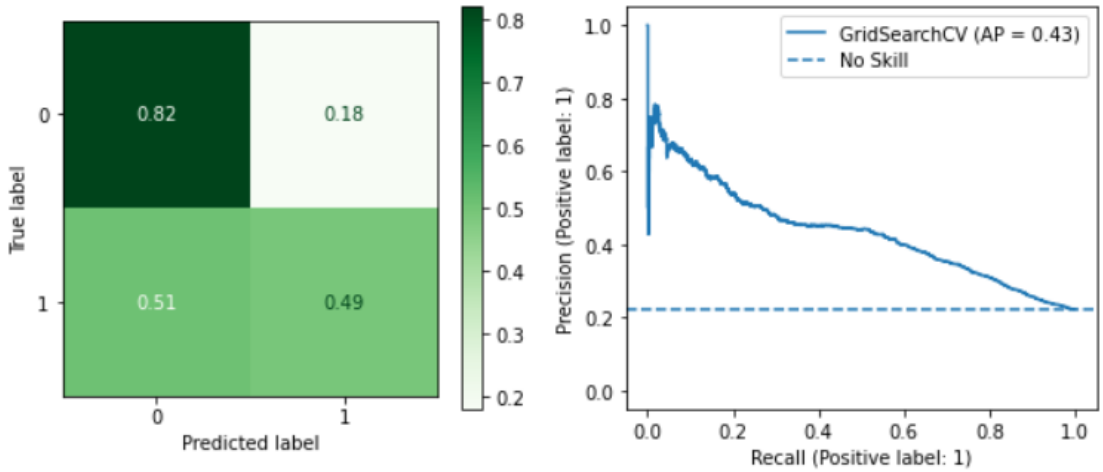


ДОДАТОК В

```
Support Vector Machine with PCA + SMOTE oversampling
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters:        {'C': 100.0, 'gamma': 'scale', 'kernel': 'rbf'}
Best validation score:  0.6755793952290514
Test Scores:
- Accuracy:     0.7698959600054047
- Recall:       0.5693519079345851
- Precision:    0.4865424430641822
- F1-score:     0.5246999720904271
- AUC:          0.5759812249521603
```

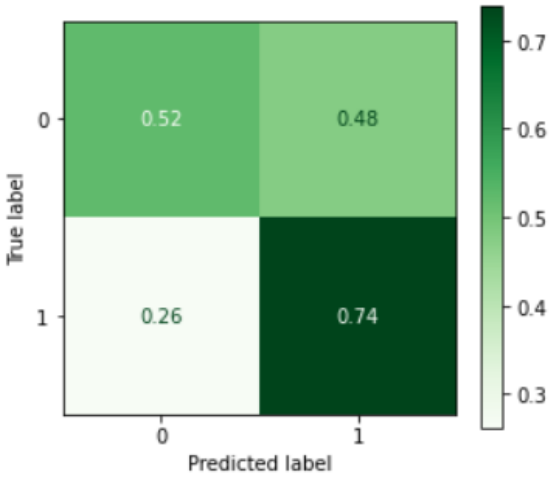


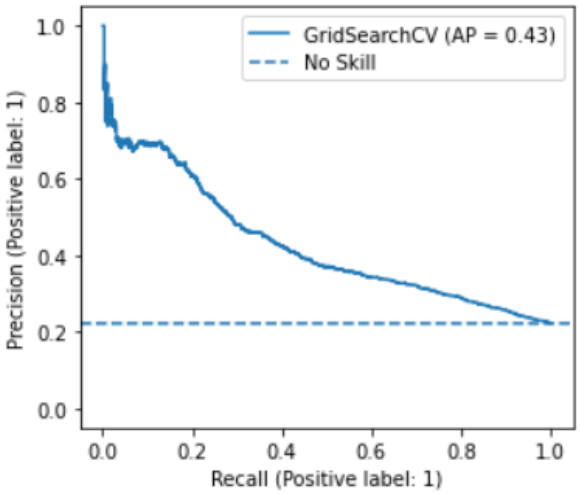

```
Support Vector Machine with PCA + KMeansSMOTE oversampling
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters:        {'C': 100.0, 'gamma': 'scale', 'kernel': 'rbf'}
Best validation score:  0.7622860213760343
Test Scores:
- Accuracy:     0.7470612079448723
- Recall:       0.49061175045427013
- Precision:    0.43997827267789247
- F1-score:     0.46391752577319584
- AUC:          0.5221116579652165
```

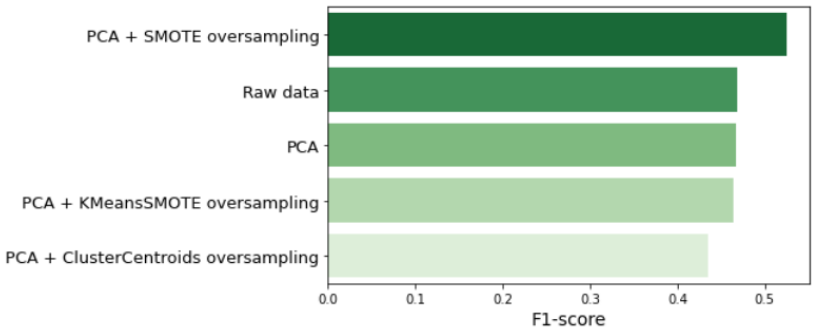


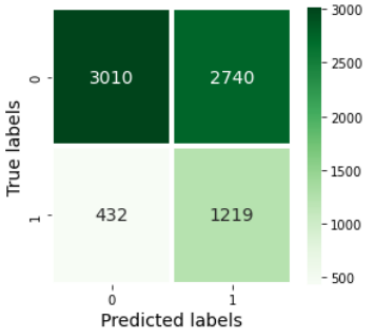
ДОДАТОК В

```
Support Vector Machine with PCA + ClusterCentroids oversampling
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters:        {'C': 100.0, 'gamma': 'scale', 'kernel': 'rbf'}
Best validation score:  0.6921628171993855
Test Scores:
- Accuracy:     0.5714092690177003
- Recall:       0.7383403997577226
- Precision:    0.3079060368779995
- F1-score:     0.43458110516934045
- AUC:          0.5523084635549912
```



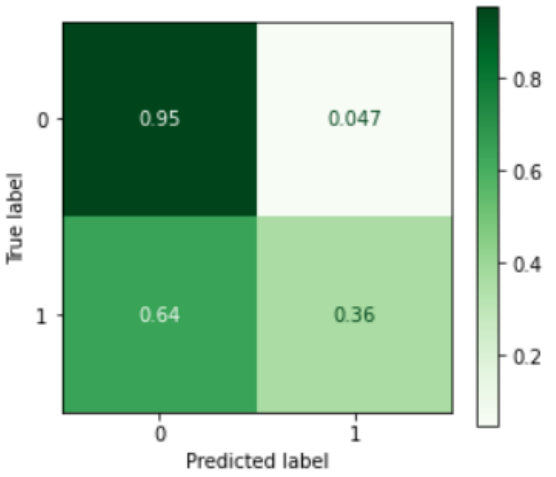


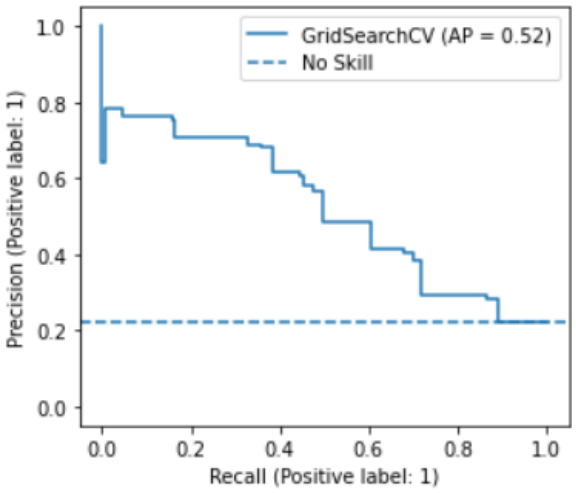




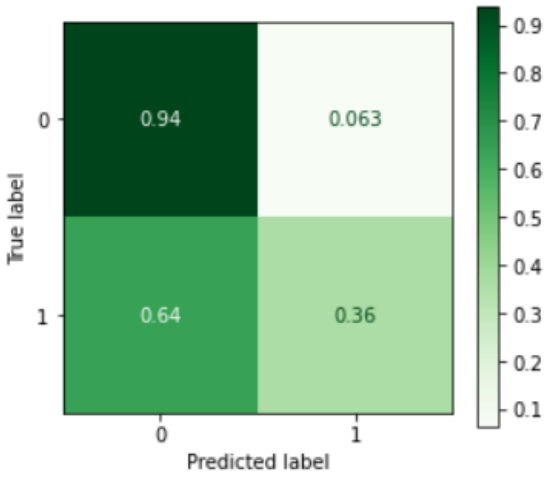
ДОДАТОК Г

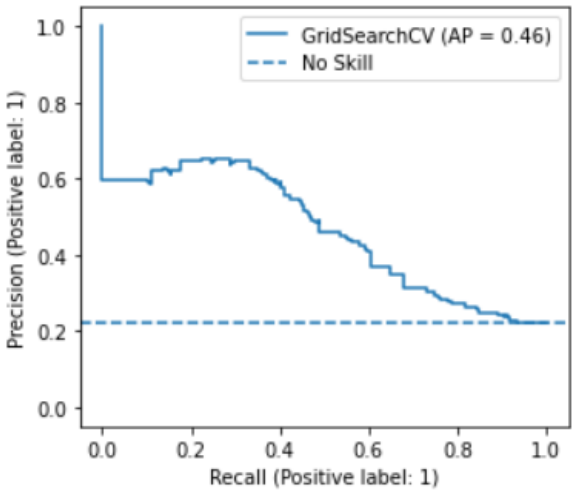
```
Decision Tree with Raw data
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Best parameters:        {'max_depth': 5}
Best validation score:  0.45813216221308634
Test Scores:
- Accuracy:     0.8205647885420889
- Recall:       0.35978195033313143
- Precision:    0.6867052023121387
- F1-score:     0.47217806041335453
- AUC:          0.5946528453403355
```





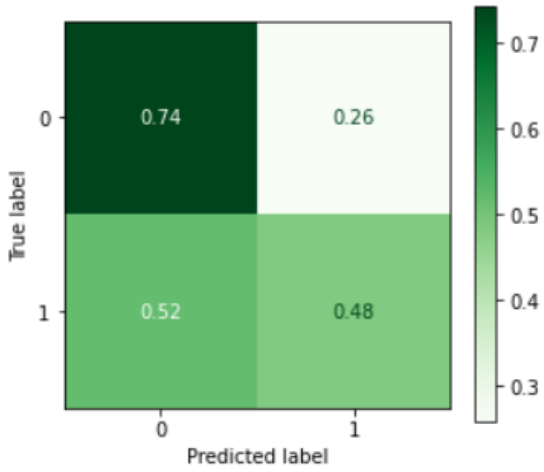

```
Decision Tree with PCA
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Best parameters:        {'max_depth': 10}
Best validation score:  0.430074611221572
Test Scores:
- Accuracy:     0.8078638021888934
- Recall:       0.3579648697758934
- Precision:    0.6201469045120671
- F1-score:     0.4539170506912442
- AUC:          0.5606678314758273
```

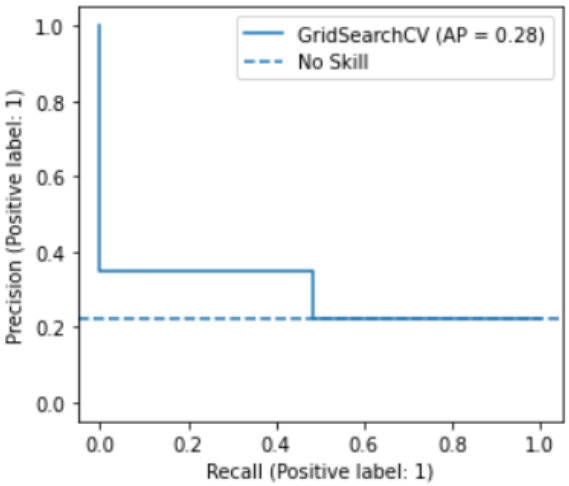




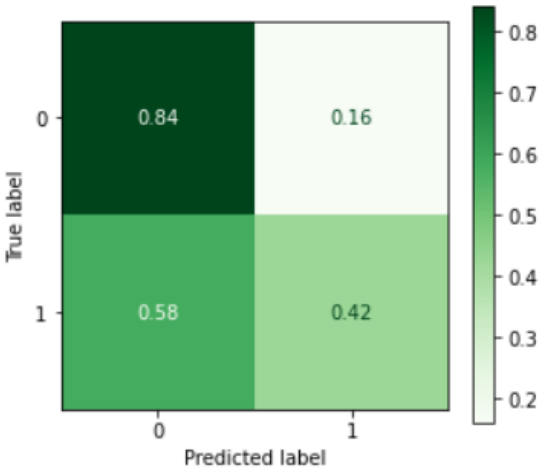
ДОДАТОК Г

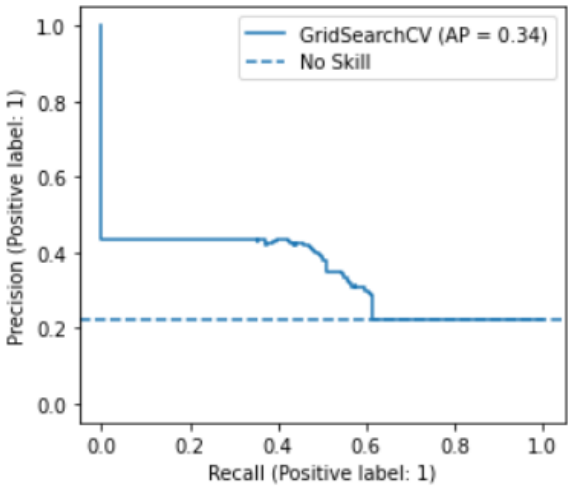
```
Decision Tree with PCA + SMOTE oversampling
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Best parameters:        {'max_depth': 50}
Best validation score:  0.762111126627109
Test Scores:
- Accuracy:     0.6842318605593839
- Recall:       0.48152634766807995
- Precision:    0.3492970123022847
- F1-score:     0.40488922841864017
- AUC:          0.47324170295505125
```





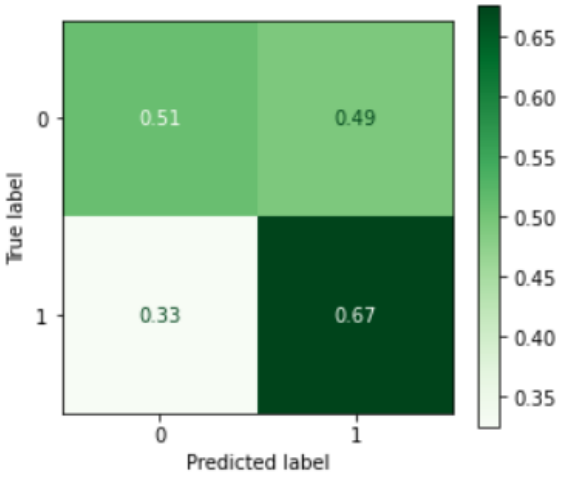

```
Decision Tree with PCA + KMeansSMOTE oversampling
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Best parameters:        {'max_depth': 20}
Best validation score:  0.7834135816908295
Test Scores:
- Accuracy:     0.7459802729360897
- Recall:       0.42216838279830404
- Precision:    0.42945163277880466
- F1-score:     0.42577886377519847
- AUC:          0.4902607576872167
```

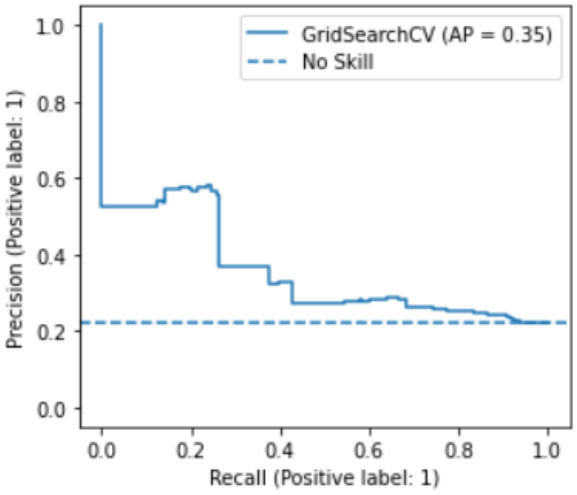


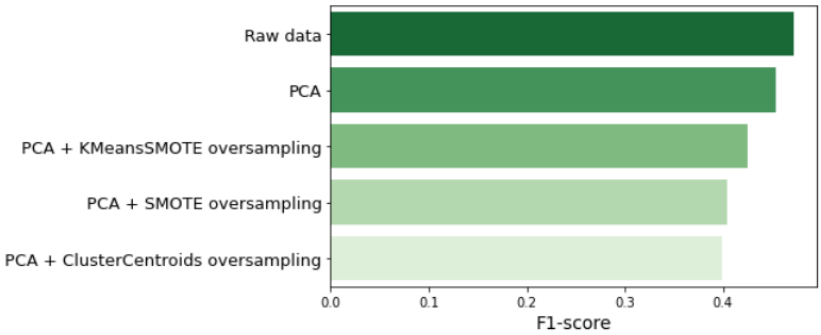


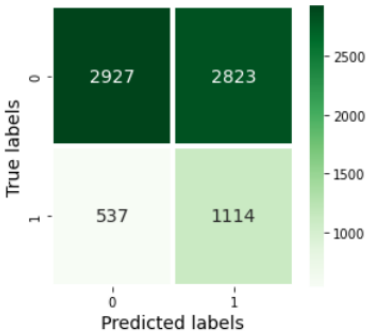
ДОДАТОК Г

```
Decision Tree with PCA + ClusterCentroids oversampling
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Best parameters:        {'max_depth': 10}
Best validation score:  0.656376360481173
Test Scores:
- Accuracy:     0.5460072963113093
- Recall:       0.6747425802543913
- Precision:    0.28295656591313184
- F1-score:     0.3987115246957767
- AUC:          0.5151284543160275
```



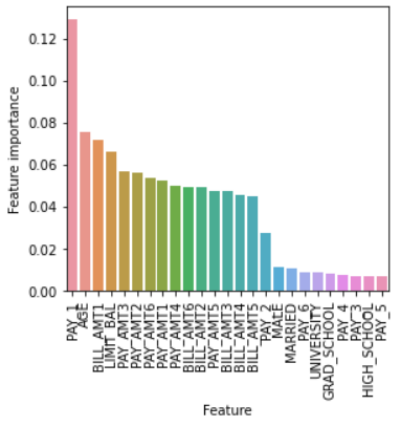


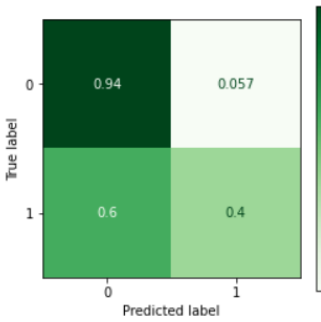


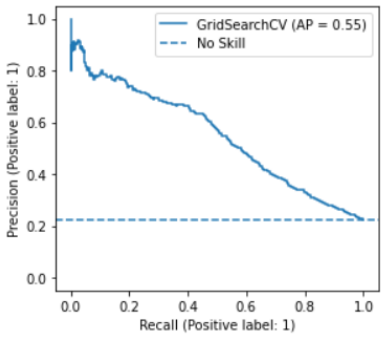


ДОДАТОК Д

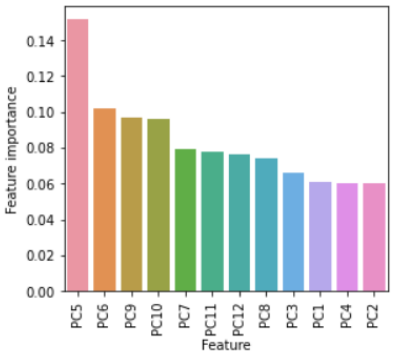
```
Random Forest with Raw data
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best parameters:        {'max_features': None, 'n_estimators': 200}
Best validation score:  0.4695030676559674
Test Scores:
- Accuracy:     0.8209701391703824
- Recall:       0.39672925499697154
- Precision:    0.665650406504065
- F1-score:     0.49715370018975336
- AUC:          0.598478035047235
```

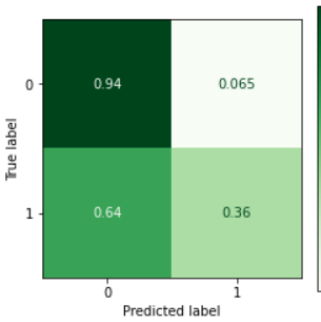


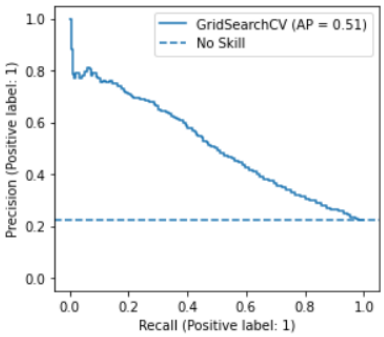





```
Random Forest with PCA
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best parameters:        {'max_features': None, 'n_estimators': 100}
Best validation score:  0.4419014087953347
Test Scores:
- Accuracy:     0.8074584515605999
- Recall:       0.36281041792852814
- Precision:    0.6162551440329218
- F1-score:     0.45672893633244377
- AUC:          0.5606042578081807
```

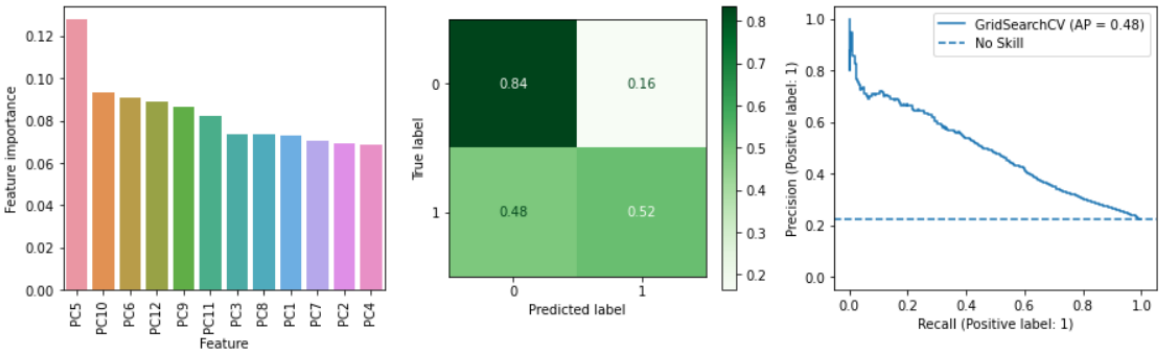




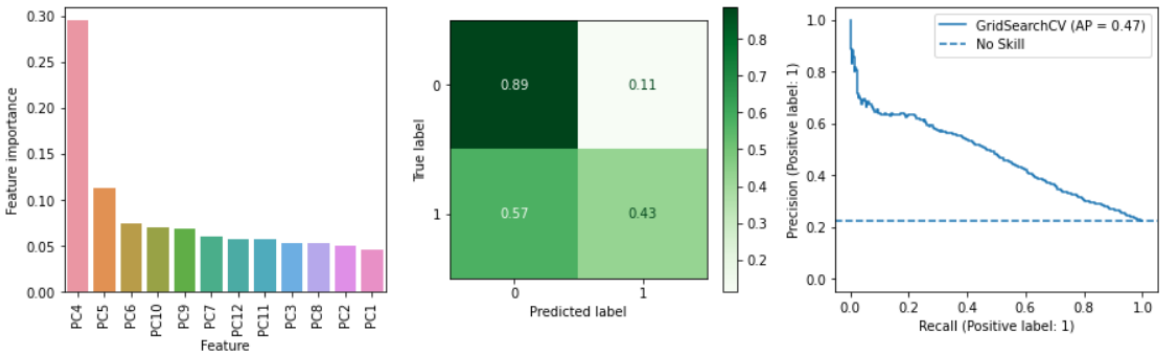


ДОДАТОК Д

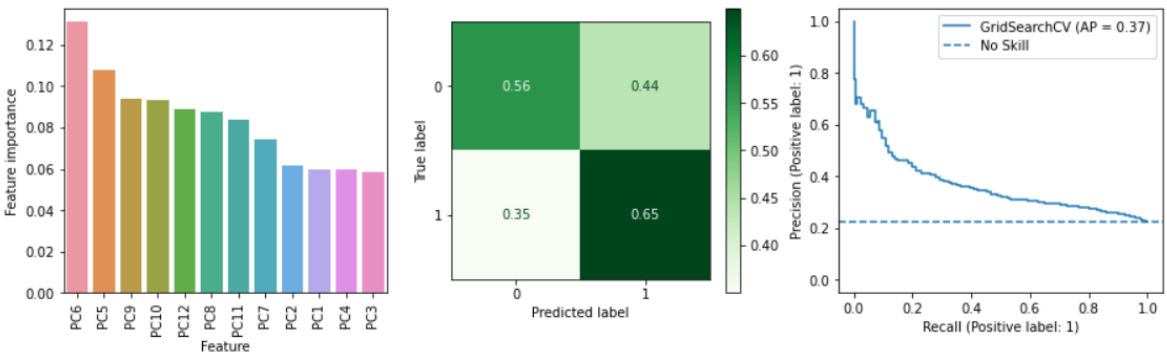
```
Random Forest with PCA + SMOTE oversampling
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best parameters:        {'max_features': 'sqrt', 'n_estimators': 200}
Best validation score:  0.8350874582586624
Test Scores:
- Accuracy:     0.7647615187136874
- Recall:       0.5160508782556027
- Precision:    0.47491638795986624
- F1-score:     0.4946298984034833
- AUC:          0.5494628251088154
```



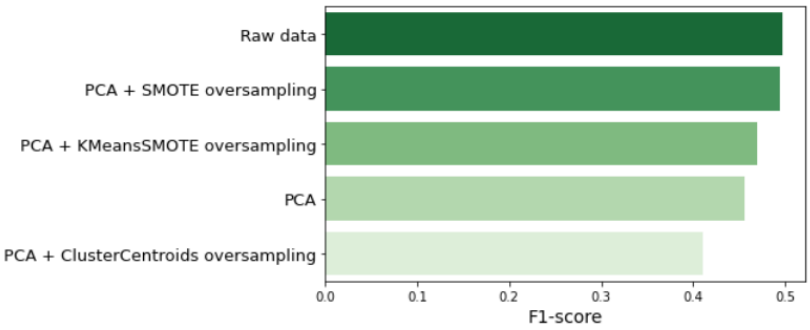

```
Random Forest with PCA + KMeansSMOTE oversampling
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best parameters:        {'max_features': None, 'n_estimators': 200}
Best validation score:  0.8111537515961622
Test Scores:
- Accuracy:     0.7844885826239697
- Recall:       0.4270139309509388
- Precision:    0.5206794682422452
- F1-score:     0.46921797004991683
- AUC:          0.537756981990863
```

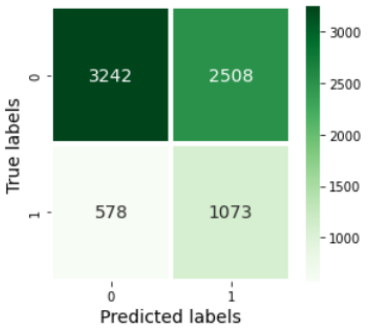



```
Random Forest with PCA + ClusterCentroids oversampling
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best parameters:        {'max_features': None, 'n_estimators': 100}
Best validation score:  0.6674024856384421
Test Scores:
- Accuracy:     0.5830293203621132
- Recall:       0.6499091459721381
- Precision:    0.2996369729125942
- F1-score:     0.4101681957186544
- AUC:          0.5138218366346375
```



ДОДАТОК Д


Raw data
PCA + SMOTE oversampling
PCA + KMeansSMOTE oversampling
PCA
PCA + ClusterCentroids oversampling
0.0
0.1
0.2
0.3
0.4
0.5
F1-score


3242
2508
578
1073
0
1
True labels
0
1
Predicted labels
3000
2500
2000
1500
1000