Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Інститут електротехніки

(інститут)

Факультет інформаційних технологій

(факультет)

Кафедра системного аналізу і управління

(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеню магістра

(бакалавра, магістра)

студента Соловей Юлії Сергіївни

(ПІБ)

академічної групи 124м-21-1

(шифр)

спеціальності 124 - Системний аналіз

(код і назва спеціальності)

на тему *«Бізнес-аналіз додатка для ідентифікації кіберзлочинів у банківській сфері»*

(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
Кваліфікаційної роботи	*к.т.н., доц. Желдак Т.А.*			
розділів:				
Інформаційно-аналітичний	*к.т.н., доц. Желдак Т.А.*			
Спеціальній	*к.т.н., доц. Желдак Т.А.*			
Проектний	*к.т.н., доц. Желдак Т.А.*			
Рецензент	*к.т.н., доц. Приходенко С.Д.*			
Нормоконтролер	*к.ф.-м.н., доц. Хом'як Т.В*			

Дніпро

2022

ЗАТВЕРДЖЕНО:
завідувач кафедри

Системного аналізу та управління
(повна назва)

_____ *Желдак Т.А.*
(підпис) (прізвище, ініціали)

«_____»_____20___року

Завдання
на кваліфікаційну роботу
ступеня *магістра*
(бакалавра, магістра)

студенту *Соловей Ю. С.* **академічної групи** *124м-21-1*
(прізвище та ініціали) (шифр)

спеціальності 124 Системний аналіз

на тему *«Бізнес-аналіз додатка для ідентифікації кіберзлочинів у банківській сфері»*

Затверджена наказом ректора НТУ «Дніпровська політехніка» від *31 жовтня 2022 р.№ 1200-С*

Розділ	Зміст	Термін виконання
Інформаційно-аналітичний	Вивчити діяльність, основні завдання та методи боротьби з кіберзлочинами в Дніпропетровській області. Дослідити такі методи для вирішення проблеми задачі дослідження, як: збалансування незбалансованих даних методами undersampling та oversampling та використання методів XGBoost, AdaBoost, градієнтного бустингу та ансамблів відповідних моделей для побудови класифікатора банківських транзакцій.	
Спеціальний	Реалізувати обрані методи дослідження та зробити висновки про краще рішення поставленої задачі, яке було отримане в ході експериментів.	
Проектний	Збір і обробка вимог до додатка протидії кіберзлочинам, розробка логічної схеми БД додатка, моделювання елементів алгоритму роботи додатка, а також результати впровадження та оцінка ефективності додатка.	

Завдання видано _____ Желдак Т.А.
(підпис керівника) (прізвище, ініціали)

Дата видачі _____

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Соловей Ю.С.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: с.116, рис.44, табл.1, додатків 3, джерел 25.

Управління протидії кіберзлочинам в Дніпропетровській області ДКП НПУ є підрозділом кримінальної поліції, яка в свою чергу входить до складу апарату центрального органу управління поліції. Метою даної установи є захист населення від кіберзлочинів, запобігання неправомірних дій та розслідування і викриття вже скоєних злочинів. Спектр роботи даного відділку досить широкий: поліцїанти займаються боротьбою з вірусами, DdoS-атаками, спамом, шахрайством з банківськими системами і крадіжкою особистих даних та інше.

Об'єктом дослідження є діяльність управління протидії кіберзлочинам в Дніпропетровській області ДКП НПУ, а саме виявлення неправомірних дій в банківській сфері.

Предметом дослідження є використання методів бізнес-аналізу додатка для ідентифікації кіберзлочинів у банківській сфері управлінням протидії кіберзлочинам в Дніпропетровській області ДКП НПУ.

Метою роботи є підвищення ефективності роботи кіберполіціантів при розпізнаванні шахрайських дій та подальшого пошуку злочинців, можливого прогнозування неправомірних дій в банківських системах за допомогою розробленої Antifraud detection-системи.

В *інформаційно-аналітичному розділі* наведено аналіз об’єкту дослідження, основні задачі установи, та ключові проблеми в ньому. Розглянуто основні методи розв’язання поставленої задачі.

У *спеціальному розділі* сформульовано та виконано задачу машинного навчання, а саме побудову класифікатора на незбалансованих даних за допомогою методів undersampling та oversampling для збалансування незбалансованої вибірки, та методів XGBoost, AdaBoost, градієнтного бустингу та ансамблі відповідних моделей для побудови класифікатора банківських транзакцій, також виконано аналіз отриманих результатів та обрано найкращий з них.

У *проектному* розділі виконано бізнес-аналіз додатку, а саме проведено збір та обробку вимог до додатка, розроблено логічну схему БД, виконано моделювання елементів алгоритму роботи додатка, а також наведено результати застосування та оцінка ефективності додатка

Практична цінність результатів полягає у отриманні організаційно-технічних рішень, що дозволяють значно підвищити ефективність виявлення та розкриття злочинів у банківському секторі, захистити населення відповідно до задач правоохоронної установи.

Ключові слова: ПРАВООХОРОННА УСТАНОВА, СИСТЕМА ANTIFRAUD DETECTION, НЕЗБАЛАНСОВАНІ ДАНІ, МАШИННЕ НАВЧАННЯ, БУСТИНГ, БІЗНЕС-АНАЛІЗ, STAKEHOLDER, USER STORIES

THE ABSTRACT

Explanatory note: pages 116, figures 44, tables 1, annexes 3, sources 25.

The Department for Combating Cybercrime in the Dnipropetrovsk Region of the State Penitentiary NPU is a subdivision of the criminal police, which in turn is part of the staff of the central police department. The purpose of this institution is to protect the population from cybercrime, prevent wrongdoing and investigate and expose crimes already committed. The range of work of this department is quite wide: police officers are engaged in the fight against viruses, DdoS-attacks, spam, fraud with banking systems and theft of personal data. In addition, cyber police monitor the spread of pornography and neutralize pirated content.

The object of the study is the activity of the cybercrime countermeasures department in the Dnipropetrovsk region of the DKP NPU, namely the detection of illegal actions in the banking sector.

The subject of the study is the use of business analysis methods of the application for the identification of cybercrimes in the banking sector by the management of countering cybercrimes in the Dnipropetrovsk region of the DKP of the NPU.

The purpose of the work is to increase the effectiveness of cyber police officers in recognizing fraudulent actions and further searching for criminals, possible forecasting of illegal actions in banking systems with the help of the developed Antifraud detection system.

The informational and analytical section provides an analysis of the research object, the main tasks of the institution, and key problems in it. The main methods of solving the given problem are considered.

In a special section, the problem of machine learning is formulated and performed, namely, the construction of a classifier on unbalanced data using the methods of undersampling and oversampling to balance the unbalanced sample, and the methods of XGBoost, AdaBoost, gradient boosting, and an ensemble of relevant models to build a classifier of bank transactions, and the analysis of the obtained results and selected the best of them.

In the project section, a business analysis of the application was performed, namely, the collection and processing of requirements for the application was carried out, a logical database scheme was developed, simulation of the application algorithm elements was performed, and the results of application and evaluation of the application's effectiveness were also given

The practical value of the results lies in obtaining organizational and technical solutions that allow to significantly increase the effectiveness of detection and disclosure of crimes in the banking sector, to protect the population in accordance with the tasks of the law enforcement agency.

Keywords: LAW ENFORCEMENT AGENCY, ANTIFRAUD DETECTION SYSTEM, UNBALANCED DATA, MACHINE LEARNING, BOOSTING, BUSINESS ANALYSIS, STAKEHOLDER, USER STORIES

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ДКП НПУ	Департамент кіберполіції Національної поліції України
УПК	Управління протидії кіберзлочинам
«АРМОР»	Автоматизоване робоче місце оперативного робітника
МВС	Міністерство внутрішніх справ
DdoS	Distributed denial of service «розподілена відмова в обслуговуванні»
XGBoost	eXtreme Gradient Boosting, екстремальне градієнтне підсилювання
AdaBoost	Adaptive Boosting
SMOTE	Synthetic Minority Oversampling Technique
KNN	k-nearest neighbors

ВСТУП

Завдяки стрімкому прогресу в банківському секторі в напрямку діджіталізації і збільшення спектру банківських послуг, стає дедалі більше комфорту і розширюються можливості клієнта. Проте шахраї також стають все винахідливішими та вдаються до різних хитрощів, щоб нажитися чужими коштами. Часто зловмисники намагаються здійснити незаконні фінансові операції: оформити кредит на підставне ім'я, зняти кошти з чужого рахунку та інші потенційно можливі кіберзлочини. З кожним роком виникають та стрімко прогресують нові види та схеми кіберзлочинів у сфері використання платіжних систем, такі як скімінг (шимінг), фрод, кеш-трапінг, кардінг, а також несанкціоноване списання коштів з банківських рахунків за допомогою систем дистанційного банківського обслуговування та інші.

З метою протидії таким кіберзлочинам розробляються спеціальні Antifraud-системи, що оцінюють підозрілість фінансових транзакцій з точки зору шахрайства. Сервіс повідомляє про виявлення сумнівних дій і дає рекомендації щодо їх подальшої обробки. Такі системи є не в усіх банківських установах, а також є досить дорогими, тому не всі сайти підключаються до захищених платіжних систем, а тому велика кількість громадян знаходяться під ризиком натрапити на шахраїв.

З 2015 року було засновано окремий орган Національної поліції України (НПУ) - департамент кіберполіції НПУ. Департамент кіберполіції у Дніпропетровській області бере безпосередньо участь у формуванні та забезпеченні реалізації державної політики щодо попередження та протидії кримінальним правопорушенням, механізм підготовки, вчинення або приховування яких передбачає використання електронно-обчислювальних машин (комп'ютерів), систем та комп'ютерних мереж і мереж електрозв'язку.

Спектр діяльності підрозділу є досить широкий: поліцейські займаються боротьбою з вірусами, DdoS-атаками, спамом, шахрайством з банківськими системами і крадіжкою особистих даних. Крім того, кіберполіцейські відстежують поширення порнографії і нейтралізують піратський контент. Тому розробка і впровадження Antifraud-систем в правоохоронних установах є вкрай необхідною для забезпечення безпеки для усього населення та завчасного виявлення злочинців і попередження наступних злочинів.

Проте для грамотної розробки додатку Antifraud-системи, що задовольнятиме усі потреби замовника, подальшого впровадження та ефективного використання функціоналу необхідно використання методів бізнес-аналізу. Бізнес-аналіз – це набір методів, які допомагають зрозуміти структуру, особливості компанії клієнта (замовника), визначити її потреби і запропонувати варіанти рішення задачі. В процесі бізнес-аналізу спеціалісти вибирають оптимальне рішення, готують до нього вимоги, оцінюють, яка функціональність найбільш важлива для замовника, укладають та погоджують документацію для розробників. Тобто саме методи бізнес-аналізу дозволять розробити такий функціонал, який буде задовольняти усі вимоги замовника-кіберполіції, а сам додаток буде зручним та зрозумілим у використанні.

Відтак тема кваліфікаційної роботи є досить актуальною.

Об'єктом дослідження є діяльність управління протидії кіберзлочинам в Дніпропетровській області ДКП НПУ, а саме виявлення неправомірних дій в банківській сфері.

Предметом дослідження є використання методів бізнес-аналізу додатка для ідентифікації кіберзлочинів у банківській сфері управлінням протидії кіберзлочинам в Дніпропетровській області ДКП НПУ.

Метою магістерської роботи є підвищення ефективності роботи кіберполіцейських при розпізнаванні шахрайських дій в банківській сфері за допомогою розробленої та впровадженої Antifraud-системи.

Для досягнення поставленої мети в роботі визначені та вирішені наступні *задачі дослідження*:

- дослідити методи, за допомогою яких можна побудувати класифікатор банківських транзакцій;
- розробка та побудова класифікатора на незбалансованих даних;
- аналіз отриманих результатів та вибір оптимального методу;
- проведення бізнес-аналізу додатка для ідентифікації кіберзлочинів у банківській сфері, а саме збір та обробка вимог до додатка, розробка логічної схеми БД, моделювання елементів алгоритму роботи, а також результати застосування та оцінка ефективності додатка;

Для розв’язання поставлених задач в роботі запропоновано використовувати наступні *методи дослідження*:

- методи Undersampling та Oversampling для збалансування незбалансованої вибірки;
- методи XGBoost, AdaBoost та градієнтного бустингу та ансамблі відповідних моделей для побудови класифікатора банківських транзакцій з метою виявлення шахрайської діяльності
- різноманітні техніки виконання бізнес аналізу додатка, тощо.

Практична цінність отриманих результатів полягає як і у отриманні готового функціоналу класифікатора банківських транзакцій, який дозволятиме визначати до якого класу відноситься та чи інша транзакція, тобто чи є вона "хороша", або "погана", так і у формуванні результатів проведеного бізнес-аналізу додатку, що дозволять у подальшому реалізувати застосунок, який матиме зрозумілий та наочний інтерфейс користувача, матиме увесь необхідний функціонал, і дозволить у подальшому, використовуючи як зовнішні, так і

внутрішні банки даних, відшукувати злочинців або ж попереджати подальші неправомірні злочини.

Економічний ефект від отриманих результатів очікується позитивним завдяки тому, що зменшується щорічний збиток від фінансового шахрайства в сфері онлайн платежів.

Соціальний ефект від результатів роботи очікується позитивним через скорочення кількості злочинів в фінансовій сфері, захист населення від неправомірних транзакцій, тобто призводить до підвищення довіри населення до органів Національної поліції України. Також важливим є те, що правильно проведений бізнес-аналіз додатку дозволить у подальшому розробити зручний, практичний та зрозумілий у використанні для поліціантів функціонал.

1 ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

1.1 Діяльність управління протидії кіберзлочинам в Дніпропетровській області ДКП НПУ як об'єкт дослідження

Управління протидії кіберзлочинам в Дніпропетровській області ДКП НПУ є підрозділом кримінальної поліції, яка в свою чергу входить до складу апарату центрального органу управління поліції. Метою даної установи є захист населення від кіберзлочинів, запобігання неправомірних дій та розслідування і викриття вже скоєних злочинів.

З положення про Управління протидії кіберзлочинам в місті Дніпро Департаменту кіберполіції Національної поліції України :

1. Управління протидії кіберзлочинам в місті Дніпро Департаменту кіберполіції Національної поліції України (далі – УПК в Дніпропетровській області) є територіальним (відокремленим) підрозділом Департаменту кіберполіції Національної поліції України (далі – ДКП) та згідно із законодавством України здійснює оперативно-розшукову діяльність.

2. Повне найменування – Управління протидії кіберзлочинам в місті Дніпро Департаменту кіберполіції Національної поліції України, скорочена назва – УПК в Дніпропетровській області.

3. УПК в Дніпропетровській області відповідно до законодавства України забезпечує реалізацію державної політики у сфері протидії кіберзлочинності, здійснює інформаційно-аналітичне забезпечення ДКП про стан вирішення питань, віднесених до його компетенції.

4. До складу УПК в Дніпропетровській області входять різноманітні структурні підрозділи, які підпорядковані безпосередньо начальнику управління;

5. У своїй діяльності УПК в Дніпропетровській області керується Конституцією України та законами України, а також указами Президента України і постановами Верховної Ради України, актами Кабінету Міністрів України, міжнародними договорами України, іншими нормативними актами, Положенням про Департамент кіберполіції Національної поліції України та цим Положенням.

Діяльність УПК в Дніпропетровській області ґрунтується на принципах верховенства права, дотримання прав і свобод людини, гуманізму, поваги до особи, законності, відкритості та прозорості, політичної нейтральності, взаємодії з населенням на засадах партнерства, безперервності. Управління діє на основі поєднання єдиноначальності та колегіальності у вирішенні питань оперативно-службової діяльності, перспективного й поточного планування, а також своєчасного і якісного виконання планових заходів, наказів, рішень і доручень керівництва, з урахуванням наукових рекомендацій та передового досвіду, у тому числі міжнародного.

Завдання УПК в Дніпропетровській області:

1. Виявлення, запобігання та припинення кримінальних правопорушень, механізм підготовки, учинення або приховування яких передбачає використання електронно-обчислювальних машин (комп’ютерів), систем та комп’ютерних мереж і мереж електрозв’язку (далі – сфера протидії кіберзлочинності).

2. Установлення причин і умов, які сприяють учиненню кримінальних правопорушень у сфері протидії кіберзлочинності, та вжиття заходів щодо їх усунення.

3. Сприяння в порядку, передбаченому законодавством, іншим підрозділам Національної поліції України у попередженні, виявленні та припиненні кримінальних правопорушень.

Функції УПК в Дніпропетровській області.

УПК в Дніпропетровській області відповідно до покладених на нього завдань: визначає, розробляє та забезпечує реалізацію комплексу організаційних і практичних заходів, спрямованих на попередження та протидію кримінальним правопорушенням у сфері протидії кіберзлочинності на території обслуговування, у межах своїх повноважень уживає необхідних оперативно-розшукових та організаційно-практичних заходів щодо викриття причин і умов, які призводять до вчинення кримінальних правопорушень у сфері протидії кіберзлочинності, уживає передбачених законодавством організаційно-технічні заходи із збору та обліку інформації про кримінальні правопорушення у сфері протидії кіберзлочинності, узагальнює результати діяльності з протидії кіберзлочинності та надає таку інформацію керівництву ДКП, використовує сили, засоби та оперативно-пошукові обліки у протидії кримінальним правопорушенням, визначає основні напрями і тактику оперативно-розшукової діяльності, пов’язаної з виявленням кримінальних правопорушень у сфері протидії кіберзлочинності на території обслуговування та з урахуванням цього розробляє пропозиції та надає їх керівництву ДКП з метою підвищення ефективності оперативно-розшукової діяльності та інше.

1.2 Використання аналізу даних, комп’ютерних технологій та машинного навчання в роботі кіберполіціанта

Як було зазначено в попередніх розділах однією з важливих умов підвищення рівня протидії кіберзлочинності є широке використання сучасних досягнень науково-технічного прогресу, які останніми роками зробили прорив у сфері інформаційних технологій.

Найбільше всього працівники кіберполіції в ході рослідувань використовують спеціальні внутрішні засекречені бази даних, наприклад «АРМОР» (Автоматизоване Робоче Місце Оперативного Робітника), та інші

засекречені внутрішні пошукові бази даних. Також в ході роботи оперативними працівниками часто складаються схеми злочинних угруповань з використанням спеціалізованих аналітичних програмних продуктів, наприклад IBM i2 Analyst's Notebook.

Також, в ході практики мною було виявлено основні тенденції розвитку інформаційних технологій в правоохоронній сфері, а саме:

– розробка системи розпізнавання шахрайських дій з кредитними картками;

– удосконалення форм та методів управління системами інформаційного забезпечення;

– централізація та інтеграція комп'ютерних банків даних;

– впровадження новітніх інформаційних технологій для ведення та криміналістичних обліків;

– розбудова та широке використання ефективних та потужних комп'ютерних мереж;

– застосування спеціалізованих засобів захисту інформації;

– налагодження ефективного взаємообміну кримінологічною інформацією на міждержавному рівні.

Все це забезпечує суттєве підвищення рівня боротьби зі злочинністю.

Одну з вищезазначених задач було детально розглянуто в кваліфікаційній роботі, а саме розробка системи Antifraud detection, що буде дозволяти виявляти шахрайські дії під час проведення транзакцій.

1.3 Система Antifraud detection

В наш час, час розвитку інформаційних технологій, комп'ютеризації багатьох процесів, все більш частіше загальні повсякденні процеси ми виконуємо за допомогою Інтернет ресурсів та комп'ютерних приладь.

Прикладом слугує здійснення онлайн-покупок в інтернеті з використанням банківських карток. Поруч з розвитком йде і зростання шахрайських дій з кредитними картками, так званий фрод.

Фрод (Fraud) - це проведення шахрайських операцій, зокрема, за допомогою мережі Інтернет. Існує безліч видів шахрайства, більшість з них націлені на заволодіння даних банківської карти людини або самого пластику. Серед них є:

– фішинг (коли до власнику картки звертаються «співробітники банку» по пошті або телефону з проханням назвати дані карти, або відомий сайт копіюється зловмисниками),

– скімінг (копіювання даних карти через спеціальні пристрої, що встановлюються на банкомати),

– хакерські атаки і віруси , що надсилаються разом зі спамом по електронній пошті.

Antifraud detection - це система моніторингу та запобігання шахрайських операцій, яка в режимі реального часу перевіряє кожен платіж, проганяючи їх через десятки, а часом сотні фільтрів. Механізми антифрода працюють таким чином, щоб простежити, чи немає в платежі чогось «незвичайного». Завдання системи - перевірити кожну транзакцію, знайти в ній «підозрілі» моменти і винести рішення - відхилити платіж або пропустити його, або ж віднести його до певного класу, щоб потім можна було розслідувати і зібрати всю необхідну інформацію по власнику картки, IP-адреса та ін, що б допомогло знайти злочинця. Система антифрода складається з декількох компонентів: це автоматичний моніторинг транзакцій, а також моніторинг транзакцій в «ручному» режимі для крайніх випадків.

Сервіс антифрода включає типові та унікальні правила, фільтри і списки для перевірки кожної транзакції. Зокрема, найбільш популярні наступні обмеження:

- кількість покупок по одній банківській картці за певний період часу;
- максимальна сума разової покупки по одній карті в певний період;
- число карт, використовуваних одним користувачем в певний період часу;
- кількість користувачів, що використовують одну карту;
- облік історії покупок по банківських картах і користувачами («чорні» або «білі» списки).

Також часто використовуються наступні фільтри:

- валідатори, наприклад, перевірка реквізитів банківської картки на коректність;
- географія, коли IP-адреса, з якого користувач намагається зробити покупку, асоціюється з конкретною країною.
- відповідності параметрів, наприклад, країни IP-адреси платника і емітента банківської карти.
- стоп-листи, коли карта вже була помічена в шахрайських операціях або її власник заявив в банк-емітент про компрометацію даних.

Для реалізації таких типових правил виконується швидке розпізнавання користувача за різними параметрами і алгоритмам, в т.ч. за допомогою машинного навчання. Завдяки технологіям Big Data формується набір даних для автоматичної інтелектуальної оцінки споживчої поведінки. Інформація про самі транзакції і її метадані (відправник і одержувач платежу, сума, час, місце, додаткові відомості) агрегується і зіставляється з історією попередніх платежів. Як правило, правдива інформація про власника карти і сукупність параметрів користувача відповідає стандартним шаблонам поведінки добропорядних покупців.

Так, засоби Machine Learning формують шаблон користувача поведінки, за допомогою алгоритмів кластеризації визначаючи найбільш типову для цього клієнта суму зняття готівки або покупок. На відміну від вищерозглянутих

фільтрів і обмежень, самонавчаючі ML-моделі здатні розширювати раніше задані правила, підлаштовуючись під клієнта. Однак, така гнучкість не конфліктує з точністю. Якщо окрема транзакція не вкладається в раніше сформовані шаблони з урахуванням можливих припущень, вона вважається аномалією. Наприклад, людина раптово знімає або витрачає нехарактерну для нього суму. Інший підозрілий випадок, коли з одного рахунку йде відразу кілька платежів в різні місця з однаковою сумою. Ще ознакою шахрайства вважається перерахування дрібних сум на безліч різних рахунків.

Таким чином, алгоритми машинного навчання дозволяють Antifraud detection-системі вести безперервний моніторинг і виявлення підозрілих випадків, забезпечуючи гнучке налаштування параметрів фільтрації за рахунок інтерактивного формування поведінкових шаблонів. Крім того, засоби Machine Learning автоматизують прийняття рішень, відхиляючи аномальні операції і блокуючи скомпрометовані карти [1].

1.4 Класифікатор дерево рішень

Дерево рішень - класифікатор, побудований на основі вирішальних правил виду «якщо, то», упорядкованих в деревоподібну ієрархічну структуру.

В основі роботи дерева рішень лежить процес рекурсивного розбиття вихідної множини об'єктів на підмножини, асоційовані з попередньо заданими класами. Розбиття проводиться за допомогою вирішальних правил, в яких здійснюється перевірка значень атрибутів по заданій умові. Будуються на основі навчання з учителем. В якості навчального набору даних використовується безліч спостережень, для яких попередньо задана мітка класу.

1.4.1 Структура класифікатора дерево рішень

Структурно дерево рішень складається з об'єктів двох типів - вузлів (node) і листя (leaf). У вузлах розташовані вирішальні правила і підмножини

спостережень, які їм задовольняють. У листі містяться класифіковані деревом спостереження: кожен лист асоціюється з одним з класів, і об'єкту, який розподіляється в лист, присвоюється відповідна позначка класу.

Дерево рішень є лінійним класифікатором, тобто виробляє розбиття об'єктів в багатовимірному просторі площинами (в двовимірному випадку - лініями).

Широка популярність дерев рішень обумовлена наступними їх перевагами:

- правила в них формуються практично на природній мові, що робить пояснює здатність дерев рішень дуже високою;
- можуть працювати як з числовими, так і з категоріальним даними;
- вимагають відносно невелику попередню обробку даних, зокрема, не вимагають нормалізації, створення фіктивних змінних, можуть працювати з пропусками;
- можуть працювати з великими обсягами даних.
- Разом з тим, деревам рішень притаманний ряд недоліків:
- нестійкість - навіть невеликі зміни в даних можуть привести до значних змін результатів класифікації;
- оскільки алгоритми побудови дерев рішень є жадібними, вони не гарантують побудови оптимального дерева;
- схильність до перенавчання [6].

Основні етапи побудови дерева рішень:

В ході побудови дерева рішень потрібно вирішити кілька основних проблем, з кожною з яких пов'язаний відповідний крок процесу навчання:

1) Вибір атрибута, за яким буде проводитися розбиття в даному вузлі (атрибута розбиття).

2) Вибір критерію зупинки навчання.

3) Вибір методу відсікання гілок (спрощення).

4) Оцінка точності побудованого дерева.

Розглянемо кожен крок більш детально:

1. Вибір атрибуту розбиття.

Найбільш популярними критеріями є теоретико-інформаційний та статистичний.

Теоретико-інформаційний критерій - критерій заснований на поняттях теорії інформації, а саме - інформаційної ентропії.

Статистичний підхід – в основі лежить використання індексу Джині і показує, наскільки часто випадково обраний приклад навчальної вибірки буде розпізнано неправильно, за умови, що цільові значення в цій множині були взяті з певного статистичного розподілу.

2. Критерій зупинки алгоритму.

Фактично алгоритм навчання дерева рішень буде працювати до тих пір, поки в результаті не будуть отримані абсолютно «чисті» підмножини, в кожному з яких будуть приклади одного класу. Можливо при цьому буде побудовано дерево, в якому для кожного прикладу буде створено окремий лист. Очевидно, що таке дерево виявиться марним, оскільки воно буде перенавченим - кожному прикладу буде відповідати свій унікальний шлях в дереві, а отже, і набір правил, актуальний тільки для даного прикладу.

Рішенням проблеми є примусова зупинка побудови дерева, поки воно не стало перенавчання. Для цього розроблені такі підходи, як: рання зупинка, обмеження глибини дерева, завдання мінімально допустимого число прикладів у вузлі, відсікання гілок.

3. Вилучення правил та візуалізація результатів.

Іноді навіть спрощене дерево рішень все ще є занадто складним для візуального сприйняття і інтерпретації. У цьому випадку може виявитися корисним витягти з дерева вирішальні правила і організувати їх в набори, що описують класи. Для вилучення правил потрібно відстежити всі шляхи від кореневого вузла до листя дерева. Кожен такий шлях дасть правило, що складається з безлічі умов, що представляють собою перевірку в кожному вузлі шляху.

Візуалізація складних дерев рішень у вигляді вирішальних правил замість ієрархічної структури з вузлів і листя може виявитися більш зручною для візуального сприйняття [7]. В даний час дерева рішень продовжують розвиватися: створюються нові алгоритми (SHAID, MARS, Random Forest) і їх модифікації, вивчаються проблеми побудови ансамблів моделей на основі дерев рішень.

Як зазначалося вище бінарний класифікатор дерево рішень будується на основі навчання з учителем, а в якості навчального набору даних використовується безліч спостережень, для яких попередньо задана мітка класу.

При вирішенні багатьох практичних завдань методами машинного навчання дослідники стикаються з тим, що в навчальній вибірці присутня незбалансованість класів, тобто класи представлені нерівномірно (imbalanced dataset). Зокрема, ця проблема актуальна при побудові бінарного класифікатора при рішенні завдання розпізнавання шахрайських махінацій в банківських транзакціях, так як частка «поганих» транзакцій украй рідко перевищує 3-5%, а в більшості випадків знаходиться на рівні 1%. Наприклад, при побудові дерева класифікуючих правил на навчальній вибірці може виявитися, що результуюча модель містить вкрай мало правил або зовсім «порожнє» дерево, тобто побудований на таких наборах даних класифікатор може виявитися абсолютно неефективним.

1.4.2 Методи Undersampling та Oversampling для збалансування незбалансованої вибірки

Клас, що представлений в навчальних даних меншим числом прикладів, назвемо міноритарним (від англ. Minority - меншість), а представлений великим числом прикладів - мажоритарним (від англ. Majority - більшість) [2].

Відновлення балансу класів може проходити двома шляхами. У першому випадку видаляють деяку кількість прикладів мажоритарного класу (undersampling), у другому - збільшують кількість прикладів міноритарного (oversampling). Далі розглянемо ці методи більш детально.

1.Випадкове видалення прикладів мажоритарного класу (Random Undersampling)

Це найпростіша стратегія, яка полягає у тому, що розраховується число K - кількість мажоритарних прикладів, яке необхідно видалити для досягнення необхідного рівня співвідношення різних класів. Випадковим чином вибираються K мажоритарних прикладів і видаляються.

На рисунку 1.1. зображено приклад деякого набору даних в двовимірному просторі ознак до і після використання зазначеної стратегії семплінгу [3].

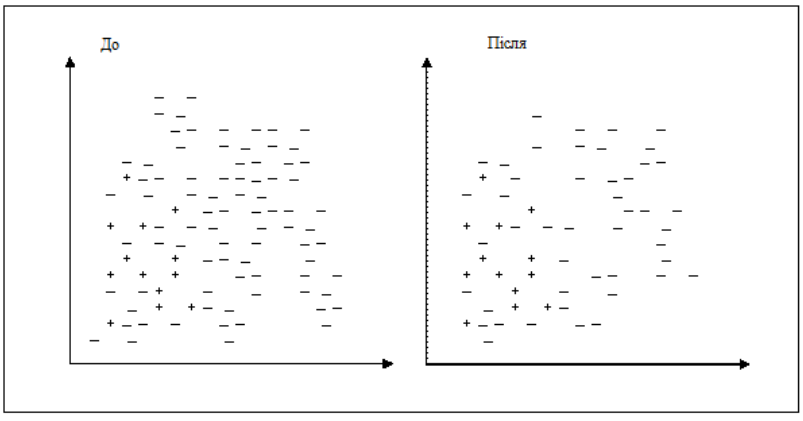

Рисунок 1.1 - Випадкове видалення прикладів мажоритарного класу

2. Збільшення міноритарного класу. Дублювання прикладів міноритарного класу (oversampling).

Найпростіший спосіб збільшити кількість прикладів міноритарного класу - випадковим чином вибрати спостереження з нього і додати їх в загальний набір даних, поки не буде досягнутий баланс між класами більшості і меншості. Залежно від того, яке співвідношення класів необхідно, вибирається число випадкових записів для дублювання.

До переваг такого підходу відносяться його простота, легкість реалізації та надається їм можливість змінити баланс в будь-яку потрібну сторону. Проте в свою чергу, застосування oversampling може привести до перенавчання.

Такий підхід до відновлення балансу не завжди ефективний, тому був запропонований спеціальний метод збільшення числа прикладів міноритарного класу - алгоритм SMOTE(англ. Synthetic Minority Oversampling Technique), що дозволяє задавати число записів, яке необхідно штучно згенерувати, при цьому який створює схожі на наявні записи в міноритарному класі, проте не дублює їх.

Для створення нового запису знаходять різницю $d = X_b - X_a$, де X_a, X_b- вектори ознак «сусідніх» прикладів a і b з міноритарного класу. Їх знаходять, використовуючи алгоритм найближчого сусіда KNN.

Алгоритм SMOTE дозволяє задавати кількість записів, яке необхідно штучно згенерувати. Ступінь подібності прикладів a і b можна регулювати шляхом зміни числа найближчих сусідів k.

На рисунку 1.2 схематично зображено те, як в двовимірному просторі ознак можуть розташовуватися штучно згенеровані приклади. Тобто в SMOTE ми синтезуємо елементи для класу меншини в безпосередній близькості від уже існуючих елементів [3].

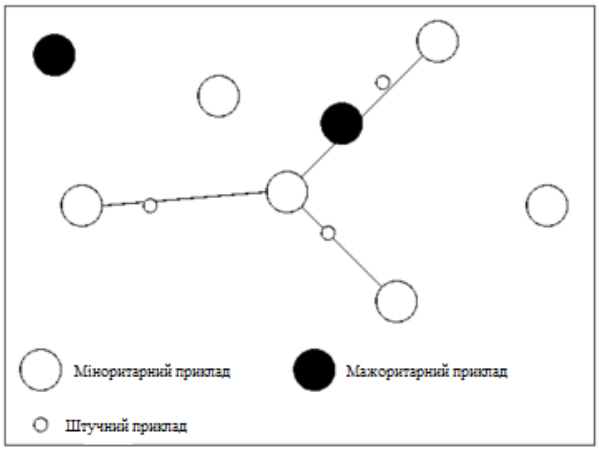

Рисунок 1.2 - Штучно створені нові приклади міноритарного класу

1.4.3 Критерії оцінки якості бінарного класифікатора

Введемо таке поняття, як матриця класифікації, або як її ще називають матриця помилок (confusion matrix). При цьому зазвичай міноритарний клас приймається за позитивний (1), а мажоритарний - за негативний (-1). Тоді матриця буде мати вигляд, показаний на рис. 1.3:

Передбачуваний клас			
Фактичний клас		$y = 1$	$y = -1$
	$a(x) = 1$	TP	FP
	$a(x) = -1$	FN	TN

Рисунок 1.3 - Матриця класифікації (випадок з двома класами)

Існує 4 величини, відповідні елементам матриці помилок (1.1-1.4) :

- True positive (TP):

$$TP = \sum_{i=0}^{n} [a(x_i) = +1][y_i = +1] \qquad (1.1)$$

- True negative (TN):

$$TN = \sum_{i=0}^{n} [a(x_i) = -1][y_i = -1] \quad (1.2)$$

- False positive (FP):

$$FP = \sum_{i=0}^{n} [a(x_i) = +1][y_i = -1] \quad (1.3)$$

- False negative (FN):

$$FN = \sum_{i=0}^{n} [a(x_i) = -1][y_i = +1] \quad (1.4)$$

P означає що класифікатор визначає клас об'єкта як позитивний, а N - негативний. T – означає, що клас передбачений правильно, відповідно F – неправильно [15].

В такій задачі бінарної класифікації використовуються наступні оцінки класифікаторів:

- частка правильно класифікованих об'єктів (accuracy),
- точність (Precision),
- повнота (Recall),
- F-міра та ін.

Розглянемо далі більш детально кожен з них, їх переваги та недоліки.

1. Частка правильно класифікованих об'єктів (accuracy).

Accuracy - найпростіша оцінка класифікації:

$$Accuracy(a) = \frac{\sum_{i=1}^{N} \mathbb{I}[a(x_i) = y_i]}{N} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1.5)$$

Фактично це ймовірність того, що клас буде передбачений правильно.

Якщо розглядати на прикладі класифікації кредитних транзакцій, то accuracy показує частку правильно розпізнаних підозрілих транзакцій.

2. Точність (Precision).

$$Precision(a) = \frac{TP}{TP + FP} \quad (1.6)$$

Точність показує яку частку об'єктів, розпізнаних як об'єкти позитивного класу, було передбачено вірно.

На прикладі: точність - це скільки зі розпізнаних класифікатором як підозрілі і віднесених до шахрайських транзакцій є дійсно такими.

3. Повнота (Recall).

$$Recall(a) = \frac{TP}{TP + FN} \quad (1.7)$$

Повнота показує, яку частку об'єктів, що реально належать до позитивного класу, передбачено вірно.

На прикладі: повнота - це скільки з шахрайських транзакцій, які було перевірено, було віднесено саме до підозрілих.

4. F-міра.

Точність і повнота оцінюють якість класифікатора достатньо добре для задач зі зміщеною апріорною ймовірністю, але якщо навчається модель з високою точністю, то може статися так, що повнота у такого класифікатора низька і навпаки. У зв’язку з цим, щоб зв'язати точність з повнотою вводять F-міру, як середнє гармонійне точності і повноти:

$$Fmeasure = \frac{2Precision \cdot Recall}{Precision + Recall} \quad (1.8)$$

У деяких завданнях одна метрика важливіше іншої. Наприклад це може залежати від замовника, його цілей. В випадку аналізу банківських транзакцій повнота може бути важливіше точності, адже краще, якщо система виявить якомога більше підозрілих транзакцій, які в подальшій роботі можна власноруч про аналізувати і відсіяти, а ось якщо класифікатор пропустить кілька таких транзакцій, то можна залишитися без якихось важливих деталей і випустити підозрілі транзакції із виду . Проте можливо і те, що для робити кіберполіції

важливо швидке реагування і якомога точна класифікація. В такому випадку точність матиме більшу вагу, ніж повнота.

Для встановлення важливості конкретної метрики розглядається параметрична F-міра:

$$Fmeasure_{\beta} = \frac{(1+\beta^2)Precision \cdot Recall}{\beta^2 Precision + Recall} \quad (1.9)$$

де $\beta\epsilon[0,\infty)$, при $\beta = 0$ буде отримано точність, при $\beta = 1$ – непараметричну F-міру, а при $\beta = \infty$ - повноту [14].

1.5 Методи XGBoost, AdaBoost та градієнтного бустингу для побудови класифікатора

У машинному навчанні, незалежно від того, стикаємося ми з проблемою класифікації або регресії, вибір моделі надзвичайно важливий, щоб мати будь-які шанси отримати хороші результати. Цей вибір може залежати від багатьох змінних задачі: кількості даних, розмірності простору, гіпотези розподілу і т.д.

Слабке зміщення і розкид моделі, хоча вони найчастіше змінюються в протилежних напрямках, є двома найбільш фундаментальними особливостями, очікуваними для моделі. Дійсно, щоб мати можливість «вирішити» проблему, ми хочемо, щоб в нашій моделі було досить ступенів свободи для вирішення базової складності даних, з якими ми працюємо, але ми також хочемо, щоб у неї було не забагато ступенів свободи, щоб уникнути розкиду і бути більш стійкою. Це так званий компроміс між зміщенням і розкидом.

В ансамблевій теорії навчання існує поняття слабких учнів (або базових моделей), яких можна використовувати в якості «будівельних блоків» для проектування більш складних моделей шляхом об'єднання кількох з них. Ідея ансамблевих методів полягає в тому, щоб спробувати зменшити зміщення і / або

розкид таких слабких учнів, об'єднуючи кілька з них разом, щоб створити сильного учня (або модель ансамблю), який досягає кращих результатів.

Існує три основних типи мета-алгоритмів, які спрямовані на об'єднання слабких учнів:

– *Бустинг*. Розглядають однорідних слабких учнів, навчають їх послідовно адаптивним способом і об'єднують їх, слідуючи детермінованій стратегії.

– *Беггінг.* Розглядають однорідних слабких учнів, навчають їх паралельно і незалежно один від одного, а потім об'єднують їх, слідуючи деякому детермінованому процесу усереднення.

– *Стекінг.* Враховують різнорідних слабких учнів, вивчають їх паралельно і об'єднують їх, навчаючи метамодель для виведення прогнозу, заснованого на прогнозах різних слабких моделей [4].

Далі розглянемо більш детально про бустинг.

Бустинг (англ. Boosting - посилення) - композиційний метаалгоритм машинного навчання, застосовується, головним чином, для зменшення зсуву (похибки оцінки), а також дисперсії в навчанні з учителем. Також визначається як сімейство алгоритмів машинного навчання, перетворюючи слабкі навчальні алгоритми в сильні [5]. Тобто це метод, який полягає в тому, щоб адаптувати послідовно декількох слабких учнів адаптивним способом: кожна модель в послідовності підбирається, що надає великої ваги об'єктів в датасеті, які погано оброблялися попередніми моделями в послідовності. Інтуїтивно, кожна нова модель фокусує свої зусилля на найбільш складних об'єктах вибірки при навчанні попередніх моделей, щоб ми отримали в кінці процесу сильного учня з більш низьким зсувом (навіть якщо вийде так, що бустинг буде при цьому зменшувати розкид). Бустинг, як і беггінг, може використовуватися як для задач регресії, так і для класифікації.

Розглянемо далі більш детально про 3 види бустингу: адаптивний бустинг (Adaboost), градієнтний бустинг та XGBoost.

1. Адаптивний бустинг (Adaboost)

AdaBoost (скорочення від Adaptive Boosting) - алгоритм машинного навчання, може використовуватися в поєднанні з декількома алгоритмами класифікації для поліпшення їх ефективності. Алгоритм підсилює класифікатори, об'єднуючи їх в «комітет». AdaBoost є адаптивним в тому сенсі, що кожен наступний комітет класифікаторів будується по об'єктах, невірно класифікованих попередніми комітетами.

Алгоритм AdaBoost для бінарної классификації:

Вихідні дані: вибірка розміром $N: z = \{z_1, z_2, \dots, z_N\}$, де $z_i = (x_{i,} y_i)$, $x_i \in X, y_i \in \{-1, +1\}$, M - максимальна кількість класифікаторів.

Вихід: Класифікатор $H: X \rightarrow \{-1, +1\}$.

Ініціалізуємо :

$$D_1(i) = \frac{1}{m}, i = 1, \dots, m. \tag{1.10}$$

Для кожного $t = 1, \dots, T$:

• Знаходимо класифікатор $h_t: X \rightarrow \{-1, +1\}$ який мінімізує зважену помилку класифікації:

$$h_t = \arg\min_{h_j \in H} \epsilon_j, \tag{1.11}$$

$$\text{де } \epsilon_j = \sum_{i=1}^{m} D_t(i)\,[y_i \neq h_j(x_i)] \tag{1.12}$$

• Якщо величина $\epsilon_j \geq 0.5$, то зупиняємося.

• Вибираємо $\alpha_t \in R$

$$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}, \tag{1.13}$$

де ϵ_t зважена помилка класифікатора h_t.

• Оновлюємо розподіл :

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}, \quad (1.14)$$

де Z_t є нормалізуючим параметром (обраним так, щоб D_{t+1} було розподілом ймовірностей, тобто

$$\sum_{i=1}^{m} D_{t+1}(i) = 1). \quad (1.15)$$

Будуємо результуючий класифікатор:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \quad (1.16)$$

Вираз для поновлення розподілу D_t має бути сконструйоване таким чином, щоб виконувалася умова :

$$e^{-\alpha_t y_i h_t(x_i)} \begin{cases} < 1, y(i) = h_t(x_i) \\ > 1, y(i) \neq h_t(x_i) \end{cases} \quad (1.17)$$

Таким чином, після вибору оптимального класифікатора h_t для розподілу D_t, об'єкти x_i, які класифікатор h_t ідентифікує коректно, мають ваги менші, ніж ті, які ідентифікуються некоректно. Отже, коли алгоритм тестує класифікатори на розподілі D_{t+1}, він буде вибирати класифікатор, який краще ідентифікує об'єкти невірно розпізнаються попереднім класифікатором [8].

2. Градієнтний бустинг

Градієнтний бустинг - окремий випадок бустинга, в якому помилка мінімізується алгоритмом градієнтного спуску. Тобто, найменш кваліфіковані кандидати відсіюються якомога раніше.

В основі алгоритму лежить послідовне уточнення функції, що представляє собою лінійну комбінацію базових класифікаторів, з тим щоб мінімізувати диференційовану функцію втрат. Градієнтний бустинг - це один з найбільш універсальних і сильних методів машинного навчання, відомих на

сьогоднішній день. Зокрема, на градієнтному бустингу над деревами рішень заснований алгоритм ранжирування видачі компанії Яндекс.

Наведемо кроки реалізації градієнтного бустингу:

1. Модель будується по вибірці даних.

2. Ця модель робить передбачення для всього набору даних.

3. За прогнозами і істинним значенням обчислюються помилки.

4. Нова модель будується з урахуванням помилок як цільових змінних. При цьому за мету стоїть знайти краще розділення для мінімізації помилки.

5. Передбачення, зроблені за допомогою цієї нової моделі, поєднуються з передбаченнями попередніх.

6. Знову обчислюються помилки з використанням цих передбачених значень і істинних значень.

7. Цей процес повторюється, поки функція помилки не перестане змінюватися або поки не буде досягнуто максимальної кількості предикторів [10].

Алгоритм градієнтного бустингу:

Вхід: навчальний набір $\{(x_i, y_i)\}_{i=1}^{n}$, диференційна функція втрат $L\big(y, F(x)\big)$, число ітерацій *M*.

Алгоритм:

3. Іінцілізуємо модель з постійним значенням:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{1.18}$$

2. Для *m* = від 1 до *M*:

1. Обчислити так звані псевдо-залишки:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \tag{1.19}$$

для $i = 1, \dots, n$.

2. Підібрати базового учня (або слабкого учня, наприклад, дерево) $h_m(x)$ до псевдо-залишкам, тобто навчити його за допомогою навчальної вибірки $\{(x_i, y_i)\}_{i=1}^n$.

3. Обчислити множник γ_m шляхом вирішення наступної одновимірної задачі оптимізації:

$$\gamma_m = \arg\min_\gamma \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \quad (1.20)$$

4. Оновлення моделі:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \quad (1.21)$$

5. Вихід $F_M(x)$. [9]

3. XGBoost

Екстремальний градієнтний бустинг (XGBoost - Extreme Gradient Boosting) - це просунута реалізація градієнтного бустинга. Цей алгоритм має високу передбачальну здатністю і в десять разів швидше будь-яких інших методів градієнтного бустінга. Крім того, включає в себе різні регуляризації, що зменшує перенавчання і покращує загальну продуктивність.

Переваги:

– Здійснює регуляризацію, що допомагає боротися з перенавчанням.

– Можливе розпаралелювання, що робить його набагато швидше градієнтного бустингу.

– Дозволяє користувачеві визначити власні цілі оптимізації та критерії оцінки, додаючи вимірювання в модель.

– Має вбудовану підпрограму для обробки пропущених значень.

– Знаходить поділи до заданої максимальної глибини, а потім починає обрізати дерево і видаляти поділи, після яких немає позитивних висновків.

– Дозволяє робити крос-валідацію на кожній ітерації бустингу і отже полегшує обчислення оптимального числа ітерацій бустингу [10].

Алгоритм XGBoost:

Функція для оптимізації градієнтного бустингу:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i) + \mho(f)) \quad (1.22)$$

де:

- l - функція втрат;
- $y_i, \hat{y_i}^{(t)}$- значення *i*-го елемента навчальної вибірки і сума передбачень перших *t* дерев відповідно;
- x_i - набір ознак *i*-го елемента навчальної вибірки;
- f_t- функція (у нашому випадку дерево), яку ми хочемо навчити на кроці *t*;
- $f_t(x_i)$ - передбачення на *i*-му елементі навчальної вибірки;
- $\mho(f)$ - регуляризація функції f. При цьому $\mho(f) = \gamma T + \frac{1}{2}\lambda||w||^2$, де T - кількість вершин в дереві, *w* - значення в листі, а γ та λ – параметри регуляризації.

Далі за допомогою розкладання Тейлора до другого члена можемо наблизити оптимізацію функції $\mathcal{L}^{(t)}$ виразами:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y_i}^{(t-1)} + g_i f_t(x_i) + 0.5\, h_i {f_i}^2(x_i)\right) + \mho(f_t), \quad (1.23)$$

де:
$$g_i = \frac{\partial l(y_i, \hat{y_i}^{(t-1)})}{\partial\, \hat{y_i}^{(t-1)}}, \quad (1.24)$$

$$h_i = \frac{\partial^2 l(y_i, \hat{y_i}^{(t-1)})}{\partial^2 \hat{y_i}^{(t-1)}}. \quad (1.25)$$

Оскільки ми хочемо мінімізувати помилку моделі на навчальній вибірці, нам потрібно знайти мінімум $\mathcal{L}^{(t)}$ для кожного *t*.

Мінімум цього виразу щодо $f_t(x_i)$ знаходиться в точці $f_t(x_i) = \frac{-g_i}{h_i}$. Кожне окреме дерево ансамблю $f_t(x_i)$ навчається стандартним алгоритмом [11].

Як можна побачити на рис.1.4, модель з XGBoost може похвалитися кращою комбінацією "продуктивність-час навчання" серед інших алгоритмів. Інші наукові методи вимірювання продуктивності показали схожий результат [12].

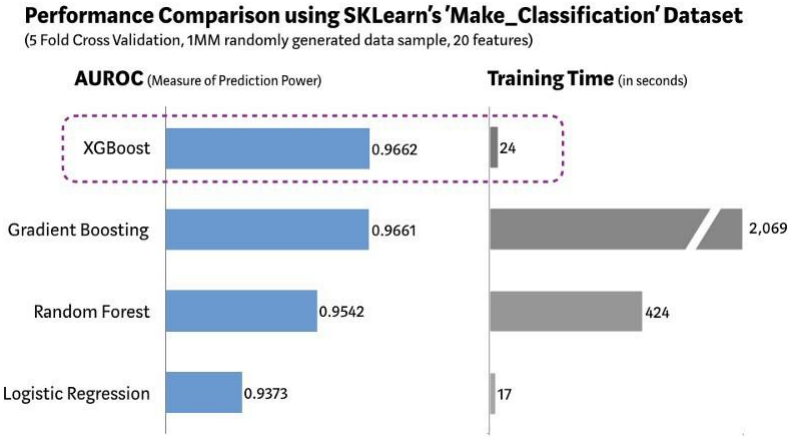

Рисунок 1.4 - Порівняння продуктивності XGBoost і інших алгоритмів на одному датасеті

1.6 Можливості мови програмування Python

За останні кілька років зростання машинного навчання набирає стрімких темпів. Це пов'язано з випуском бібліотек машинного навчання або глибокого навчання, які абстрагуються від складності або реалізації моделі. Мова програмування Python широко використовується в додатках машинного навчання, і ця мова стала загальним вибором для галузей і наукових кіл. Бібліотеки машинного навчання Python є пакетами коду, які скомпільовані разом, щоб служити спільній меті. Далі подано деякі основні бібліотеки, які використано в практичній частині.

1. Pandas.

Pandas - це високорівнева Python бібліотека для аналізу даних. Вона є високорівневою, тому що побудована на більш низькорівневій бібліотеці

NumPy, що є великим плюсом в продуктивності. В екосистемі Python, pandas - є найбільш просунутою і прогресуючою бібліотекою для обробки і аналізу даних.

В основі Pandas лежить клас DataFrame, що надає можливості роботи з двовимірними масивами неоднотипних даних. Об'єкт DataFrame складений з об'єктів Series - одновимірних масивів, об'єднаних під однією назвою. Тому в DataFrame кожен стовпець може мати свій тип даних. Бібліотека Pandas не вимагає знань лінійної алгебри або умінь працювати з багатовимірними масивами. DataFrame можна уявити як таблицю, де кожен рядок являє одиничне спостереження. А з рядками і стовпцями зручно працювати. Python-бібліотека Pandas підтримує такі ж операції, як і NumPy: індексація масивів, статистичні дослідження, роботу з порожніми (Nan) значеннями, зміна форми масивів, витяг окремих елементів, конкатенацію. Крім цього, Pandas також дозволяє читання і запис даних, наприклад, csv або excel таблиця, угруповання даних, засоби візуалізації даних, роботу з датами, роботу зі строковими значеннями і т.д.

2. Scikit-learn.

Scikit-learn - це багатопрофільна бібліотека машинного навчання Python, основна мета якої- запропонувати ефективні інструменти для аналізу даних, при цьому бібліотека побудована на інших потужних бібліотеках, таких як NumPy, SciPy і matplotlib, з підтримкою plotly, pandas і багатьох інших. SciKit пропонує функціональні можливості, які стосуються класифікації та структуруваннях даних. Вона також включає в себе регресію, кластеризації, зменшення розмірності, вибір моделі і попередню обробку даних. На додаток до цього в бібліотеці також є багато зазвичай використовуваних алгоритмів машинного навчання, таких як машини опорних векторів, випадковий ліс і підвищення градієнта. Бібліотека пропонує багато функціональних можливостей, особливо якщо розглядати їх в поєднанні з корпоративними варіантами використання алгоритмів машинного навчання[16].

Також у scikit-learn можна скористатися вбудованими метриками для відбору моделі. Це можна зробити за допомогою аргументу scoring. Також в sklearn.metrics можна отримати confusion_matrix. Це дозволяє побудувати матрицю помилок, яку ми розглядали в попередньому підрозділі, як для бінарної, так і для мультикласової класифікації.

3. Imblearn

Бібліотека Imblearn спеціально розроблена для роботи з незбалансованими наборами даних. Він надає різні методи, такі як недостатня вибірка, передискретизація і SMOTE, для обробки і видалення дисбалансу з набору даних. Ця бібліотека складається з різних методів ансамблю, таких як класифікатори упаковки, випадковий ліс і класифікатори підвищення, які можна використовувати для навчання моделей для незбалансованих наборів даних з дуже ефективною точністю.

Розглянемо більш детально, які ж саме основні фічі були використані для налаштування бібліотек бустингу, семплінгу та класифікаторів.

Параметри для моделі Xgboost:

– Objective - Цей параметр визначає функцію втрат, яку потрібно мінімізувати. Двійковий код: логістика - логістична регресія двох класифікацій, що повертає прогнозовану ймовірність;

– eval_metric - методи вимірювання для достовірних даних, в нашому випадку auc - площа під кривою;

– eta - аналогічно параметру швидкості навчання в GBM, зазвичай встановлюється значення 0,01-0,2;

– gamma - визначає мінімальну втрату функції втрат, необхідну для поділу вузла. Чим більше значення цього параметра, тим більш консервативний алгоритм;

– min_child_weight - сума мінімальних ваг вибірок, використовується, щоб уникнути перевантаження. Коли його значення велике, це може перешкодити моделі вивчати місцеві спеціальні зразки, але якщо це значення занадто високе, це призведе до недостатньої підгонки;

– max_depth – є максимальною глибиною дерева. Це значення також використовується, щоб уникнути перевантаження. Чим більше max_depth, тим модель буде вивчати більш конкретні і локальні вибірки. Типове значення: 3-10;

– colsample_bytree - використовується для контролю частки числа стовпців, обраних випадковим чином (кожен стовпець є елементом). Типове значення: 0,5-1;

– reg_alpha - його можна застосовувати в разі дуже великих розмірів, що прискорює алгоритм;

– reg_lambda - зважений член регуляризації, рідко застосовується, проте може знайти більше застосування для скорочення переоснащення;

– n_estimators - число дерев в "лісі". Чим більше його значення, тим більш глибшим буде навчання.

– random_state - початкове значення для генерації випадкових чисел;

– n_jobs - кількість ядер для побудови моделі і передбачень, та інші.

Це не всі можливі параметри, які можна застосувати при побудові класифікатора методом Xgboost, проте таких достатньо аби отримати результати машинного навчання.

Параметри для моделі градієнтного бустингу.

Деякі параметри аналогічні тим, що використовуються в моделі Xgboost, тому нижче описано тільки ті, що не було перераховано вище:

– loss - функція втрат, яку слід оптимізувати. " deviance" означає девіацію (логістичну регресію) для класифікації з імовірнісними результатами;

– learning_rate - коефіцієнт швидкості навчання;

– subsample - частка зразків, яка буде використовуватися для підбору окремих базових учнів. Якщо <1.0 призводить до зменшення дисперсії і збільшення систематичної помилки.

– criterion - функція вимірювання якості розбиття, де «friedman_mse» для середньоквадратичної помилки з оцінкою покращення за Фрідманом;

– min_samples_split - мінімальна кількість зразків, необхідних для розділення внутрішнього вузла;

– min_samples_leaf - мінімальна кількість вибірок, яка вимагається для кінцевого вузла;

– min_weight_fraction_leaf - мінімальна зважена доля від загальної суми вагів (усіх вхідних зразків), необхідних для находження в вузлі листку, та інші.

Для моделі Adaboost в цілому усі параметри були описані вище.

1.7 Висновки до розділу

В інформаційно аналітичному розділі об’єктом дослідження виступає діяльність управління протидії кіберзлочинам в Дніпропетровській області ДКП НПУ, а саме виявлення неправомірних дій в банківській сфері. Відповідно до законодавства України УПК в Дніпропетровській області забезпечує реалізацію державної політики у галузі протидії кіберзлочинності, здійснює інформаційно-аналітичне забезпечення керівництва Національної поліції України та органів державної влади про стан вирішення питань, віднесених до його компетенції, бере участь у формуванні та забезпеченні реалізації державної політики щодо попередження та протидії кримінальним правопорушенням, механізм підготовки, вчинення або приховування яких передбачає використання електронно-обчислювальних машин (комп’ютерів), систем та комп’ютерних мереж і мереж електрозв’язку.

Метою роботи даної установи є захист населення від кіберзлочинів, запобігання неправомірних дій та розслідування і викриття вже скоєних злочинів.

Однією з можливих та найбільш поширених задач кіберполіції є боротьба з шахрайством з банківськими системами. Тобто проводиться протидія кіберзлочинам у сфері використання платіжних систем, а саме проти фроду, скімінгу, кеш-трапінгу, кардінгу, несанкціонованих списань коштів з банківських рахунків за допомогою систем дистанційного банківського обслуговування та ін.

Існують спеіальні системи Antifraud detection - системи моніторингу та запобігання шахрайських операцій, які в режимі реального часу перевіряють кожен платіж, проганяючи їх через десятки, а часом сотні фільтрів, проте на даний момент вони застосовуються не у всіх банківських установах, а отже не всі платіжні канали захищені від шахрайських дій. В даному розділі було детально розглянуто які ж бувають Antifraud detection – системи, а також загальні теоретичні відомості.

Також, в ході вивчення теоретичного матеріалу, було виявлено, що досить часто при машинному навчанні вибірок стикаються з проблемою незбалансованих даних, тобто класи представлені нерівномірно (imbalanced dataset). Зокрема, ця проблема актуальна при побудові бінарного класифікатора при рішенні завдання розпізнавання шахрайських махінацій в банківських транзакціях, так як частка «поганих» транзакцій украй рідко перевищує 3-5%, а в більшості випадків знаходиться на рівні <1%. В даному розділі було розглянуто методи відновлення балансу класів – методи семплінгу. Семплінг може проводитися двома шляхами: шляхом видалення певної кількості прикладів мажоритарного класу (undersampling), або зменшенням кількість прикладів міноритарного (oversampling).

Для побудови класифікатора було обрано алгоритм дерево рішень. Досліджено, що Дерево рішень – це класифікатор, побудований на основі вирішальних правил виду «якщо, то», упорядкованих в деревоподібну ієрархічну структуру. В основі роботи дерева рішень лежить процес рекурсивного розбиття вихідної множини об'єктів на підмножини, асоційовані з попередньо заданими класами. Розбиття проводиться за допомогою вирішальних правил, в яких здійснюється перевірка значень атрибутів по заданій умові.

Також було зазначено, що в ансамблевій теорії навчання існує поняття слабких учнів (або базових моделей), яких можна використовувати в якості «будівельних блоків» для проектування більш складних моделей шляхом об'єднання кількох з них. Одним з таких і є класифікатор дерево рішень. Для зменшення зміщення і / або розкиду таких слабких учнів, шляхом об'єднання кількох з них разом, існує декілька мета-алгоритмів, одним з яких є бустинг. Бустинг - це метод, який полягає в тому, щоб адаптувати послідовно декількох слабких учнів адаптивним способом: кожна модель в послідовності підбирається, що надає великої ваги об'єктам в датасеті, які погано оброблялися попередніми моделями в послідовності. Більш детально було розглянуто 3 алгоритми бустингу, а саме XGBoost, AdaBoost та градієнтного бустингу. Було виявлено, що XGBoost є кращою комбінацією "продуктивність-час навчання" серед інших алгоритмів, проте не завжди використання цього методу може бути оптимальним, тому задача аналітика полягає в тому, щоб спробувати різні методи і визначити, який з них є більш швидким і точним.

В розділі було розглянуто критерії оцінки якості бінарного класифікатора. Було виявлено, що в задачі бінарної класифікації використовуються наступні оцінки класифікаторів: частка правильно класифікованих об'єктів (accuracy), точність (Precision), повнота (Recall), F-міра

та ін. Також було виявлено, що в таких задачах оцінка якості accuracy – не є основною, так як вона погано працює при високій апріорній ймовірності у одного з класів, що не підходить для задачі виявлення шахрайських транзакцій. А от вибір спиратися на показник Precision або Recall вже залежить від бажання замовника. В нашому випадку це 2 варіанти: або чітко відокремити хороші транзакції, при цьому є можливість того, що підозрілі транзакції будуть віднесені до «хороших» і їх буде втрачено з виду, в такому випадку варто спиратися на показник Precision, або визначити якомога повніше всі підозрілі транзакції, і в крайньому випадку перевірити їх вручну. В такому випадку це може зайняти більше часу і зросте кіль-ть транзакцій віднесених до підозрілих, проте не буде випущено з виду підозрілі транзакції, тоді слід звертати увагу на показник Recall.

Також було розглянуто основні бібліотеки, які буде використано в спеціальній частині кваліфікаційної роботи. А саме бібліотека Pandas (завдяки цій бібліотеці можна буде зчитувати дані з .csv файлу, аналізувати та впорядковувати дані, працювати, як з таблицями, завдяки DataFrame), бібліотека scikit-learn (допоможе в отримані оцінок матриці помилок), та бібліотека Imblearn (для роботи з незбалансованими наборами даних).

В результаті було розглянуто і зібрано всі данні, та матеріали, необхідні для вирішення технічного аспекту задачі магістерської роботи, а саме побудови класифікатора банківських транзакцій з метою виявлення шахрайської діяльності, з використанням методів undersampling та oversampling для збалансування незбалансованої вибірки, та методів XGBoost, AdaBoost та градієнтного бустингу для побудови класифікатора .

2 СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Постановка задачі

Для постановки задачі введемо наступні поняття. Клас, представлений в навчальних даних меншим числом прикладів, назвемо міноритарним, а представлений великим числом прикладів - мажоритарним. Ефективність бінарного класифікатора опишемо матрицею класифікації (confusion matrix). При цьому зазвичай міноритарний клас приймається за позитивний (1), а мажоритарний - за негативний (-1). Тоді матриця буде мати вигляд, показаний на рис. 2.1:

Передбачуваний клас			
Фактичний клас		$y = 1$	$y = -1$
	$a(x) = 1$	TP	FP
	$a(x) = -1$	FN	TN

Рисунок 2.1 - Матриця класифікації (випадок з двома класами)

Де P означає, що класифікатор визначає клас об'єкта як позитивний, а N - негативний. T – клас передбачений правильно, відповідно F - неправильно.

Формальна постановка задачі класифікації.

Нехай маємо задачу побудови бінарного класифікатора на множині навчальних прикладів (X_i, y), де $i = 1, \dots, n$, X_i – вектор ознак, y – мітка класу з множини $Y = \{1,2, \dots, J\}$. Крім цього, припустимо, що навчальна вибірка була отримана з множини, розподіленої по деякому ймовірнісному закону $P\ (X, y)$. Тоді метою алгоритму навчання буде побудова класифікатора h, який уможливлює правильне розпізнавання довільних прикладів, розподілених по

тим же законам з досить високою ймовірністю. Аналогічно, якщо неправильне розпізнавання веде до втрат, то метою навчання буде мінімізація повних очікуваних втрат C_t за формулою 2.1:

$$C_t = \sum_{(X,y)} P\ (X,y) C(h(X),y) \qquad (2.1)$$

де $C(h(X),y)$ – функція втрат, що виражає питомі втрати на приклад (X,y). Таким чином, повні витрати C_t є сумою втрат для всіх класифікованих спостережень.

Нехай для класифікатора h відома ймовірність $P_h(i,j)$ того, зо випадково обраний приклад відноситься до класу j, проте буде розпізнана як i. Тоді очікувані втрати класифікатора h будуть дорівнювати :

$$C_t = \sum_{i=1}^{m} \sum_{j=1}^{m} P_h(i,j) C(i,j). \qquad (2.2)$$

Варто відзначити, що $P_h(i,j) = P(i|j)P(j)$, де $P(j)$ – ймовірність того, що окремий приклад відноситься до класу j, $P_h(i,j)$ – умовна ймовірність помилкового віднесення прикладів класу j до класу i.

Таким чином, метою задачі класифікації з врахуванням втрат є знаходження класифікатору, який мінімізує повні витрати на основі рівняння 2.2.

Слід зазначити, що вхідні дані є незбалансованими, тому не всі методи, які можна застосовувати в машинному навчанні при побудові класифікатора є ефективними або дають точні результати. Також ймовірні випадки перенавчання, або класифікатор просто буде відносити всі ймовірні випадки до мажоритарного класу. Підхід зміни репрезентативності класів передбачає використання методів семплінгу, тобто відбувається додавання/видалення елементів вибірки, з метою отримання збалансованої множини. До основних методів семплінгу належать oversampling та undersampling та їх модифікації.

Далі, застосувавши методи XGBoost, AdaBoost, градієнтного бустингу та ансамблі відповідних моделей, буде побудовано класифікатор банківських транзакцій. По результатам необхідно буде про аналізувати використані моделі та методики та зробити відповідні висновки про ефективність того чи іншого методу.

2.2 Загальне представлення даних та реалізація методів XGBoost, AdaBoost, градієнтного бустингу та ансамблів відповідних моделей

Програмна реалізація методів, визначених в постановці задачі була виконана в Jupyter Notebook (командна оболонка для інтерактивних обчислень), мовою програмування Python.

Вхідні дані, які були використані в подальшій роботі були взяті з відкритого джерела Kaggle [13], так як отримання реальної поточної інформації з фін.установ є неможливим з причин конфіденційності та захисту персональної інформації.

Фрагмент даних, розділених по стовпцях в програмі Excel подано нижче на рис.2.2:

	A	B	C	D	E	F	G	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
1	Time	V1	V2	V3	V4	V5	V6	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
2	0	-1.35981	-0.07278	2.53635	1.37816	-0.33832	0.46239	0.25141	-0.01831	0.27784	-0.11047	0.06693	0.12854	-0.18911	0.13356	-0.02105	149.62	0
3	0	1.19186	0.26615	0.16648	0.44815	0.06002	-0.08236	-0.06908	-0.22578	-0.63867	0.10129	-0.33985	0.16717	0.12589	-0.00898	0.01472	2.69	0
4	1	-1.35835	-1.34016	1.77321	0.37978	-0.5032	1.8005	0.52498	0.248	0.77168	0.90941	-0.68928	-0.32764	-0.1391	-0.05535	-0.05975	378.66	0
5	1	-0.96627	-0.18523	1.79299	-0.86329	-0.01031	1.2472	-0.20804	-0.1083	0.00527	-0.19032	-1.17558	0.64738	-0.22193	0.06272	0.06146	123.5	0
6	2	-1.15823	0.87774	1.54872	0.40303	-0.40719	0.09592	0.40854	-0.00943	0.79828	-0.13746	0.14127	-0.20601	0.50229	0.21942	0.21515	69.99	0
7	2	-0.42597	0.96052	1.14111	-0.16825	0.42099	-0.02973	0.08497	-0.20825	-0.55982	-0.0264	-0.37143	-0.23279	0.10591	0.25384	0.08108	3.67	0
8	4	1.22966	0.141	0.04537	1.20261	0.19188	0.27271	-0.21963	-0.16772	-0.27071	-0.1541	-0.78006	0.75014	-0.25724	0.03451	0.00517	4.99	0
9	7	-0.64427	1.41796	1.07438	-0.4922	0.94893	0.42812	-0.15674	1.94347	-1.01545	0.0575	-0.64971	-0.41527	-0.05163	-1.20692	-1.08534	40.8	0
10	7	-0.89429	0.28616	-0.11319	-0.27153	2.6696	3.72182	0.05274	-0.07343	-0.26809	-0.20423	1.01159	0.3732	-0.38416	0.01175	0.1424	93.2	0
11	9	-0.33826	1.11959	1.04437	-0.22219	0.49936	-0.24676	0.20371	-0.24691	-0.63375	-0.12079	-0.38505	-0.06973	0.0942	0.24622	0.08308	3.68	0
12	10	1.44904	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-0.38723	-0.0093	0.31389	0.02774	0.50051	0.25137	-0.12948	0.04285	0.01625	7.8	0
13	10	0.38498	0.61611	-0.8743	-0.09402	2.92458	3.31703	0.12599	0.04992	0.23842	0.00913	0.99671	-0.76731	-0.49221	0.04247	-0.05434	9.99	0
14	10	1.25	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.10276	-0.23181	-0.48329	0.08467	0.39283	0.16113	-0.35499	0.02642	0.04242	121.5	0
15	11	1.06937	0.28772	0.82861	2.71252	-0.1784	0.33754	-0.1532	-0.03688	0.07441	-0.07141	0.10474	0.54826	0.10409	0.02149	0.02129	27.5	0
16	12	-2.79185	-0.32777	1.64175	1.76747	-0.13659	0.8076	-1.58212	1.15166	0.22218	1.02059	0.02832	-0.23275	-0.23556	-0.16478	-0.03015	58.8	0
17	12	-0.75242	0.34549	2.05732	-1.46864	-1.15839	-0.07785	0.26345	0.49962	1.35365	-0.25657	-0.06508	-0.03912	-0.08709	-0.181	0.12939	15.99	0
18	12	1.10322	-0.0403	1.26733	1.28909	-0.736	0.28807	-0.11391	-0.02461	0.196	0.0138	0.10376	0.3643	-0.38226	0.09281	0.03705	12.99	0
19	13	-0.43691	0.91897	0.92459	-0.72722	0.91568	-0.12787	-0.04702	-0.1948	-0.67264	-0.15686	-0.88839	-0.34241	-0.04903	0.07969	0.13102	0.89	0
20	14	-5.40126	-5.45015	1.1863	1.73624	3.04911	-1.76341	-2.19685	-0.5036	0.98446	2.45859	0.04212	-0.48163	-0.62127	0.39205	0.94959	46.8	0
21	15	1.49294	-1.02935	0.45479	-1.43803	-1.55543	-0.72096	-0.38791	-0.17765	-0.17507	0.04	0.29581	0.33293	-0.22038	0.0223	0.0076	5	0
22	16	0.69488	-1.36182	1.02922	0.83416	-1.19121	1.30911	-0.13833	-0.29558	-0.57196	-0.05088	-0.30421	0.072	-0.42223	0.08655	0.0635	231.71	0
23	17	0.9625	0.32846	-0.17148	2.1092	1.12957	1.69604	-0.26932	0.144	0.40249	-0.04851	-1.37187	0.39081	0.19996	0.01637	-0.01461	34.09	0
24	18	1.16662	0.50212	-0.0673	2.26157	0.4288	0.08947	-0.30717	0.0187	-0.06197	-0.10385	-0.37042	0.6032	0.10856	-0.04052	-0.01142	2.28	0

creditcard DATA (диплом)

Рисунок 2.2 – Загальний вигляд даних в програмі Excel

Набір даних містить транзакції, здійснені кредитними картками у вересні 2013 року європейськими власниками карток. Цей набір даних представляє транзакції, що відбулися за два дні. Він містить лише числові вхідні змінні, які є результатом перетворення методом головних компонент (Principal component analysis). Єдиними функціями, які не були перетворені за допомогою PCA, є "Час" та "Сума". Функція "Час" містить секунди, що пройшли між кожною транзакцією та першою транзакцією набору даних. Функція "Клас" є змінною відповіді, і вона приймає значення 1 у випадку шахрайства та 0 в іншому випадку.

Кількість транзакцій становить 284 807. Кількість не шахрайських транзакцій становить 284 315, а число шахрайських платежів – 492, а їх співвідношення становить 577.87. Це дуже незбалансований набір даних. В такому випадку результат скоріше за все буде упередженим або «перекошеним» до класу більшості і може навіть іноді взагалі ігнорувати клас меншості при класифікації. В наборі даних немає відсутніх / нульових значень ні в стовпцях, ні в рядках. Код програми (див. ДОДАТОК Г) містить аналіз і підготовку вхідних даних, реалізацію методів undersampling та oversampling для збалансування незбалансованої вибірки, та методів XGBoost, AdaBoost, градієнтного бустингу та ансамблі відповідних моделей для побудови класифікатора банківських транзакцій. Розглянемо реалізацію та підсумки експериментів покроково:

1. Проведемо візуалізацію вхідних даних. Це допоможе нам зрозуміти як розподілена вибірка. Для більш зрозумілого зображення даних, вхідні ознаки було розбито на частини по 7 ознак на кожний графік. Покажемо розподіл на рис.2.3-2.6:

```
В [9]: data[['V1','V2','V3','V4','V5', 'V6','V7']].plot(figsize = (10,6))
Out[9]: <AxesSubplot:>
```

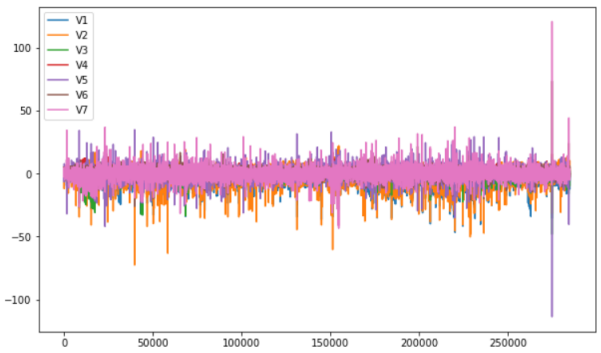

Рисунок 2.3 – Фрагмент коду, візуалізація даних, ввід і вивід № 9

```
В [10]: data[['V8','V9','V10','V11','V12', 'V13','V14']].plot(figsize = (10,6))
Out[10]: <AxesSubplot:>
```

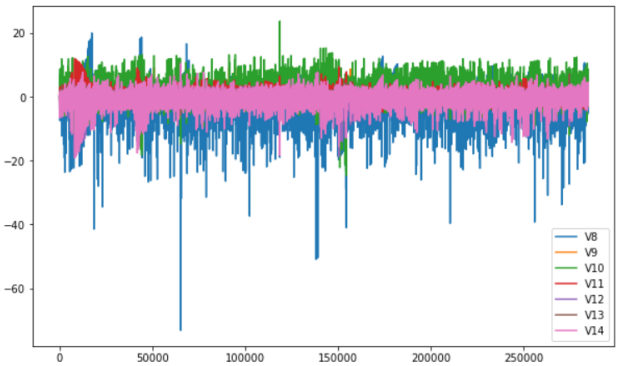

Рисунок 2.4 – Фрагмент коду, візуалізація даних, ввід і вивід № 10

```
В [11]: data[['V15','V16','V17','V18','V19', 'V20','V21']].plot(figsize = (10,6))
Out[11]: <AxesSubplot:>
```

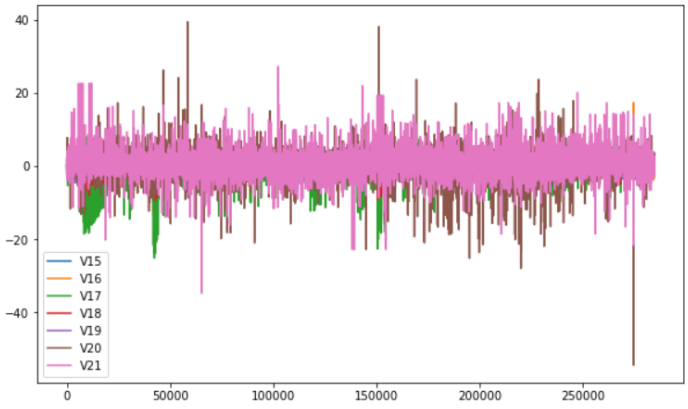

Рисунок 2.5 – Фрагмент коду, візуалізація даних, ввід і вивід № 11

```
В [12]: data[['V22','V23','V24','V25','V26', 'V27','V28']].plot(figsize = (10,6))
Out[12]: <AxesSubplot:>
```

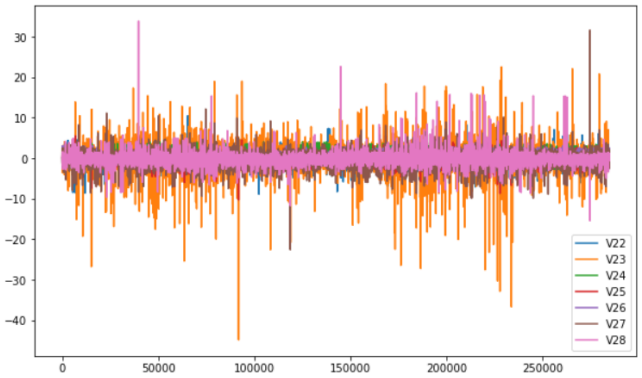

Рисунок 2.6 – Фрагмент коду, візуалізація даних, ввід і вивід № 12

Як можна побачити на графіку більшість даних розподілені нормально і знаходяться на головній діагоналі, проте існують і одиничні відхилення (викиди).

2. Проведено розбиття вхідних вибірок на навчальну і контрольну, як це показано на рис.2.7:

```
В [14]: #separating train and test data

        data_for_train = pd.DataFrame(data.iloc[: int(2*len(data)/3)])
        data_for_test = pd.DataFrame(data.iloc[int(2*len(data)/3):])

В [15]: data_for_train['Class'].value_counts()

Out[15]: 0    189501
         1       370
         Name: Class, dtype: int64

В [16]: data_for_test['Class'].value_counts()

Out[16]: 0    94814
         1      122
         Name: Class, dtype: int64
```

Рисунок 2.7 – Фрагмент коду, розбиття на навчальну і контрольну вибірки, ввід і вивід № 14-16

Як можна побачити в контрольній вибірці міститься 189 501 не шахрайська транзакція і 370 шахрайських. В тестовій вибірці міститься 94 814 не шахрайських транзакцій і 122 шахрайські.

Надалі проведемо експерименти по реалізації методів XGBoost, AdaBoost, градієнтного бустингу та ансамблів відповідних моделей при різних умовах:

Експеримент №1: Реалізація методів XGBoost, AdaBoost, градієнтного бустингу та ансамблів відповідних моделей *на незбалансованій вибірці*

Проведено реалізацію методів Adaboost, Xgboost та градієнтного бустингу на незбалансованій вибірці (див. ДОДАТОК Г, ввід і вивід № 17-20), результати роботи яких показано на рис. 2.8-2.9:

```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],pred_xgb).ravel()
print((tn, fp, fn, tp))
```

```
(94810, 4, 37, 85)
```

Рисунок 2.8 – Фрагмент коду, результати роботи моделі Xgboost, В № 17

```
В [18]: grad_model.fit(X_train, Y_train.values.ravel())
        pred_grad = grad_model.predict(X_test)

        tn, fp, fn, tp = confusion_matrix(Y_test['Class'],pred_grad).ravel()
        print((tn, fp, fn, tp))
```

```
(94778, 36, 45, 77)
```

```
В [19]: ada_model.fit(X_train, Y_train.values.ravel())
        pred_ada = ada_model.predict(X_test)

        tn, fp, fn, tp = confusion_matrix(Y_test['Class'],pred_ada).ravel()
        print((tn, fp, fn, tp))
```

```
(94802, 12, 38, 84)
```

```
В [20]: from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
        print(classification_report(Y_test['Class'],pred_xgb))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     94814
           1       0.96      0.70      0.81       122

    accuracy                           1.00     94936
   macro avg       0.98      0.85      0.90     94936
weighted avg       1.00      1.00      1.00     94936
```

Рисунок 2.9 – Фрагмент коду, результат роботи Adaboost та градієнтного бустингу В № 18-20

Як можна побачити, результати роботи цих методів відрізняються. Слід відзначити метод градієнтного бустингу, який показав незадовільні результати, адже число шахрайських транзакцій, визначених правильно, значно менший, за те, що ми мали на початку в навчальній вибірці (122 транзакції) і нижче результатів методів Adaboost та Xgboost, також значно перевищує кількість транзакцій, неправильно віднесених до «хороших», а от у методів Adaboost та Xgboost показники досить схожі між собою. Також слід відзначити, що з-поміж 3 методів, метод градієнтного бустнигу працював довше всього, процес навчання відбувався дуже довгий час, на відміну від методу Xgboost.

Виходячи з цього було прийнято рішення виключити метод градієнтного бустингу і продовжувати навчання за методами Adaboost та Xgboost.

Також на рис. 2.9 зображено оцінки класифікатору для моделі Xgboost. Дійсно можна побачити, що точність (Accuracy) такої моделі становить 1, проте як ми бачимо не всі шахрайські транзакції було виявлено, отже це підтверджує те, що показник accuracy – не є покажчиком високої точності моделі, адже має схильність до поганої роботи при високій апріорній ймовірності у одного з класів.

Натомість ми можемо побачити, що такі показники як повнота та точність не дорівнюють одиниці для класу шахрайських транзакцій, для моделі Xgboost повнота = 0.70, а точність (precision) = 0.96.

Експеримент №2: Реалізація методів XGBoost, AdaBoost, градієнтного бустингу та ансамблів відповідних моделей *на незбалансованій вибірці з подрібненням їх на 5 окремих наборів*.

З метою підвищення показників точності і повноти моделі, адже вхідна вибірка не є збалансованою, було розділено вибіку на два фрейми фродових/не фродових операцій (див. ДОДАТОК Г, В №21). Далі проведено розділення даних вибірок на 5 окремих наборів даних, кожен з яких містять усі шахрайські

транзакції та в 100 разів більше не шахрайських, щоб зменшити дисбаланс у вибірках (див. ДОДАТОК Г, В №22). Також, як було зазначено в постановці задачі, можна аналізувати модель не тільки по предікту класу (0 або 1), а і по ймовірності, що обраний елемент належить до того чи іншого класу. Це виконується завдяки методу predict_proba - це метод класифікатора, що виводить вірогідність знаходження екземпляра в кожному з класів.

На рис. 2.10, для методу Xgboost, та рис. 2.11, для методу Adaboost, виведено результати роботи скрипту на оновлених умовах експерименту.

```
B [25]: tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
        print((tn, fp, fn, tp))

        tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
        print((tn, fp, fn, tp))

        (94756, 58, 28, 94)
        (94718, 96, 25, 97)

B [26]: from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
        print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))

                      precision    recall  f1-score   support

                   0       1.00      1.00      1.00     94814
                   1       0.62      0.77      0.69       122

            accuracy                           1.00     94936
           macro avg       0.81      0.88      0.84     94936
        weighted avg       1.00      1.00      1.00     94936
```

Рисунок 2.10 – Фрагмент коду, ввід і вивід № 25 і 26

```
B [28]: tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
        print((tn, fp, fn, tp))

        tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
        print((tn, fp, fn, tp))

        (94380, 434, 27, 95)
        (87669, 7145, 23, 99)

B [29]: from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
        print(classification_report(Y_test['Class'],Y_test['mean_bin']))

                      precision    recall  f1-score   support

                   0       1.00      0.92      0.96     94814
                   1       0.01      0.81      0.03       122

            accuracy                           0.92     94936
           macro avg       0.51      0.87      0.49     94936
        weighted avg       1.00      0.92      0.96     94936
```

Рисунок 2.11 – Фрагмент коду, ввід і вивід № 28 і 29

Аналізуючи отримані результати варто відзначити, що після розбиття вибірки на 5 окремих рівних частин результати методів значно покращилися, адже як можна побачити число правильно виявлених шахрайських транзакцій зросло в обох методах, проте метод Adaboost виявив все ж таки більшу кількість шахрайських транзакцій. Проте результати методу Xgboost не кардинально відрізняються, також цей метод має більш високу точність, ніж у методу Adaboost.

Експеримент №3: Реалізація методів XGBoost, AdaBoost та ансамблів відповідних моделей на збалансованій вибірці *методами undersampling та oversampling.*

В попередніх експериментах застосовувалися методи побудови класифікаторів на незбалансованій вибірці, або ж ми намагалися відновити баланс шляхом ручного розбиття на подрібнені вибірки, аби зменшити дисбаланс між фродовими і нормальними данними. Проте існують спеціальні методи, які дозволяють збалансувати дані в вибірці - методи семплінгу: undersampling та oversampling. В програмному листі (див. ДОДАТОК Г, ввід № 30) виконано реалізацію методів XGBoost, AdaBoost та ансамблів відповідних моделей з використанням методу збалансування вибірки *undersampling*, алгоритм NearMiss Виведемо отримані результати навчання, а саме коефіцієнти матриці оцінок і матрицю помилок класифікатора, і порівняємо результати методів Xgboost (рис 2.12) та Adaboost (рис. 2.13):

```
B [32]: tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
        print((tn, fp, fn, tp))

        tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
        print((tn, fp, fn, tp))

        (93640, 1174, 24, 98)
        (93041, 1773, 21, 101)

B [33]: from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
        print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))

                      precision    recall  f1-score   support

                   0       1.00      0.99      0.99     94814
                   1       0.08      0.80      0.14       122

            accuracy                           0.99     94936
           macro avg       0.54      0.90      0.57     94936
        weighted avg       1.00      0.99      0.99     94936
```

Рисунок 2.12 – Фрагмент коду, ввід і вивід № 31-33

```
B [35]: tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
        print((tn, fp, fn, tp))

        tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
        print((tn, fp, fn, tp))

        (80423, 14391, 15, 107)
        (77342, 17472, 12, 110)

B [36]: from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
        print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))

                      precision    recall  f1-score   support

                   0       1.00      0.85      0.92     94814
                   1       0.01      0.88      0.01       122

            accuracy                           0.85     94936
           macro avg       0.50      0.86      0.47     94936
        weighted avg       1.00      0.85      0.92     94936
```

Рисунок 2.13 – Фрагмент коду, ввід і вивід № 34-36

Аналізуючи отримані результати слід зазначити, що число правильно визначених шахрайських транзакцій побільшало, а отже це свідчить про ефективність застосування методу undersampling для збалансування незбалансованої вибірки. Можна побачити, що за методом Adaboost було визначено більше шахрайських транзакцій, ніж за методом Xgboost, проте і до числа підозрілих транзакцій було зараховано набагато більшу кількість транзакцій, ніж за методом Xgboost. Виходячи з цього показник recall в методі

Adaboost є вищим, ніж в методі Xgboost, але в даному методі нижче оцінка точності моделі. Тобто можна зробити висновок, що використання методу undersampling покращило результати, адже була визначена більша кількість шахрайських транзакцій.

Тепер застосуємо метод *oversampling* для збалансування вхідних даних і визначення, який метод працює краще (див. ДОДАТОК Г. ввід №37 - Реалізація методів Xgboost та Adaboost з використанням методу oversampling) та виведемо отримані результати навчання, а саме коефіцієнти матриці оцінок і матрицю помилок класифікатора, і порівняємо результати методів Xgboost (рис 2.14) та Adaboost (рис. 2.15):

```
B [39]: tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
        print((tn, fp, fn, tp))

        tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
        print((tn, fp, fn, tp))

        (94656, 158, 23, 99)
        (94628, 186, 22, 100)

B [40]: from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
        print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))

                      precision    recall  f1-score   support

                   0       1.00      1.00      1.00     94814
                   1       0.39      0.81      0.52       122

            accuracy                           1.00     94936
           macro avg       0.69      0.90      0.76     94936
        weighted avg       1.00      1.00      1.00     94936
```

Рисунок 2.14 – Фрагмент коду, ввід і вивід № 38-40

```
B [42]: from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
        print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))

                      precision    recall  f1-score   support

                   0       1.00      0.98      0.99     94814
                   1       0.05      0.81      0.09       122

            accuracy                           0.98     94936
           macro avg       0.52      0.90      0.54     94936
        weighted avg       1.00      0.98      0.99     94936

B [43]: tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
        print((tn, fp, fn, tp))

        tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
        print((tn, fp, fn, tp))

        (92825, 1989, 23, 99)
        (81854, 12960, 21, 101)
```

Рисунок 2.15 – Фрагмент коду, ввід і вивід № 41-43

Аналізуючи отримані результати можемо зробити висновок, що застосування методу oversampling майже не вплинуло на результат роботи алгоритмів бустингу в порівнянні зі застосуванням методу undersampling, вихідні значення в методі Xgboost майже залишилися незмінними, а в методі Adaboost дещо погіршилися, так як зменшилася кількість правильно визначених шахрайських транзакцій а також зменшилося значення оцінки recall (0,81) в порівнянні з попереднім кроком (0,88), пороте збільшився показник точності (0,98) в порівнянні з попереднім експериментом (0,85).

2.3 Висновки по результатам розрахунків

Підсумовуючи результати програмної реалізації алгоритмів XGBoost, AdaBoost, градієнтного бустингу та ансамблі відповідних моделей для побудови класифікатора банківських транзакцій, детально розглянутих в спеціальному розділі , можна зробити наступні висновки:

1. Вхідні данні є достатньо сильно незбалансованими, адже кількість не шахрайських транзакцій складає 284 315, а кількість шахрайських транзакцій становить 492, що складає <1%.

2. Після застосування методів XGBoost, AdaBoost, градієнтного бустингу та ансамблів відповідних моделей на незбалансованій вибірці, метод градієнтного бустингу показав найгірші результати і час його роботи був дуже довгим, тому його було виключено з подальших розрахунків.

3. Наступним кроком було експериментально проведено спробу зменшити дисбаланс в вибірці, шляхом розділення її на 5 частин. Результати обох методів вже були кращими, а саме було виявлено більшу кількість шахрайських транзакцій.

4. На наступному кроці було застосовано методи XGBoost та AdaBoost та ансамблів відповідних моделей на збалансованій вибірці методом undersampling. Результати значно покращилися, в порівнянні з попередніми навчаннями на незбалансованих вибірках, що підкреслює важливість збалансування даних.

5. На наступному кроці було застосовано методи XGBoost та AdaBoost та ансамблів відповідних моделей на збалансованій вибірці методом oversampling. Результати були майже ідентичними до попередніх, проте метод undersampling KNN дав більш точніші результати, ніж рандомний oversampling.

Тобто, аналізуючи результати варто відзначити що обидва методи XGBoost та AdaBoost давали достатньо точні результати, проте є певні відмінності:

– алгоритм XGBoost є більш швидким (тобто коротший час навчання) і має достатньо високу точність, проте кількість правильно визначених шахрайських транзакцій була дещо нижчою, ніж у метода AdaBoost.

– алгоритм AdaBoost займав все ж більше часу для навчання, проте давав більш точну оцінку глибини матриці класифікації, проте показник точності був нижчим ніж у метода XGBoost. Це зумовлено тим, що алгоритм AdaBoost навчається на «поганих» результатах попереднього навчання. Тобто з кожним кроком все зменшується кількість поганих результатів навчання. Проте за рахунок цього достатньо велика кількість, можливо хороших транзакцій потрапляють до «підозрілих».

Тобто алгоритм XGBoost має більш вищий показник точності класифікатора, а у алгоритму AdaBoost кращий показник глибини класифікатора. В такому випадку можна відштовхуватися від побажань замовника подібних класифікаторів, або ж доречно використовувати ці два алгоритми разом. Вони, так би мовити, будуть збалансовувати результати навчання одне одного.

В нашому випадку замовником є державна правоохоронна установа, метою якої є виявити якомога більше підозрілих транзакцій, так як вони в подальшому можуть складати більшу загрозу. Тому можна вважати, що в даному випадку оцінка глибини є більш опорною, ніж точність моделі.

3 ПРОЕКТНИЙ РОЗДІЛ

Попередні розділи кваліфікаційної роботи містили саме збір наукової літератури та технічної інформації, що необхідні для розробки функціоналу класифікатору, з урахуванням усіх особливостей даних (незбалансованість вибірки), а також безпосередньо розробку самої Antifraud системи. В даному розділі буде розглянуто саме проектну частину реалізації та впровадження класифікатору банківських транзакцій. Задля того, щоб додаток Antifraud системи відповідав усім вимогам замовника (кіберполіції), містив увесь необхідний функціонал, був зручним та зрозумілим у використанні, доцільним буде використання методів бізнес-аналізу.

Бізнес-аналіз — діяльність, яка уможливлює проведення змін у організації, які приносять користь зацікавленим сторонам, шляхом виявлення потреб та обґрунтування рішень, що описують можливі шляхи реалізації змін. Основне завдання бізнес-аналізу — зробити можливим проведення змін в організації шляхом реалізації обраного рішення. Рішення розробляється з метою усунення виявлених у процесі бізнес-аналізу бізнес-проблем. Поняття рішення

включає широкий діапазон можливих шляхів усунення виявлених бізнес-проблем: розробка нових або зміна існуючих бізнес-процесів або бізнес-правил, оптимізація організаційної структури організації, розробка нових стратегічних планів організації і т.п.

3.1 Збір та обробка вимог до додатку протидії кіберзлочинам

Аналіз вимог полягає в визначенні потреб та умов, які висуваються щодо нового, чи зміненого продукту, враховуючи можливо конфліктні вимоги різних замовників, таких як користувачі чи бенефіціари.

Аналіз вимог є критичним для успішної розробки проекту. Вимоги мають бути задокументованими, вимірними, тестовними, пов'язаними з бізнес-потребами, і описаними з рівнем деталізації достатнім для конструювання системи. Вимоги можуть бути архітектурними, структурними, поведінковими, функціональними, та не функціональними [17].

Вимоги до програмного забезпечення — набір вимог щодо властивостей, якості та функцій програмного забезпечення, що буде розроблено, або знаходиться у розробці. Вимоги визначаються в процесі аналізу вимог та фіксуються в специфікації вимог, діаграмах прецедентів та інших артефактах процесу аналізу та розробки вимог [18].

Розробка вимог до програмної системи може бути розділена на декілька етапів:

- Знаходження вимог (збір, визначення потреб заінтересованих осіб та систем, проблем та ін).
- Аналіз вимог (перевірка цілісності та закінченості).
- Специфікація (документування вимог).
- Тестування вимог.

Зацікавлені сторони, зацікавлені особи, заінтересовані сторони, причетні сторони , або stakeholders — фізичні та юридичні особи, які мають легітимний інтерес у діяльності організації, тобто певною мірою залежать від неї або можуть впливати на її діяльність[19].

У проектах розробки ПЗ можуть застосовуватися різні методи виявлення вимог. Насправді навряд чи знайдеться проектна команда, в якій використовується лише один метод. Завжди є кілька типів інформації, яку треба виявляти, і різні зацікавлені особи надають перевагу різним підходам. Один користувач може чітко сформулювати, як він використовує систему, а за іншим доведеться спостерігати в роботі, щоб отримати таке розуміння використання сценарію. Методи виявлення вимог діляться на колективні, у яких беруть участь зацікавлені особи, та незалежні, коли ви працюєте самі над виявленням інформації. Колективні методи орієнтуються на виявлення користувальницьких та бізнес-вимог. Безпосередня робота з користувачами необхідна, тому що вимоги користувача пов'язані з завданнями, які користувачі повинні виконувати в системі.

Незалежні методи доповнюють вимоги, що надаються користувачами, та дозволяють виявити функціональність, про яку кінцеві користувачі можуть знати. У більшості проектів використовується поєднання колективних та незалежних методів виявлення вимог. Різні методи надають можливість по-різному досліджувати вимоги та навіть можуть виявляти різні вимоги. У наступних розділах описується кілька методів, які зазвичай застосовуються для виявлення вимог [18].

3.1.1 Методи дослідження стейкхолдерів

Інтерв'ю

Найочевидніший спосіб дізнатися, що потрібно користувачам системи, просто запитати у них. Інтерв'ю — традиційне джерело вимог як для серійних

продуктів, так і для інформаційних систем у будь-яких методиках розробки ПЗ, тобто це метод дослідження UX (анг. User Experience), під час якого дослідник задає одному користувачеві запитання про цікаву тему (наприклад, використання системи, поведінку та звички) з метою дізнатися усе про цю тему. Більшість бізнес-аналітиків організують інтерв’ю у якійсь формі: індивідуальні інтерв'ю чи інтерв'ю у невеликих групах, задля виявлення вимог у проектах. У проектах гнучкої розробки (agile) інтерв'ю активно використовуються як механізм безпосереднього залучення користувачів. Інтерв'ю простіше запланувати і провести чим великі групові заходи, такі як семінари з виявлення вимог. Інтерв'ю також зручні для збирання бізнес-вимог у топ-менеджерів, у яких зазвичай мало часу на зустрічі.

Далі наводяться кілька рекомендацій щодо проведення інтерв'ю:

– Встановіть контакт;
– Дотримуйтесь меж проекту;
– Заздалегідь підготуйте питання та попередні моделі;
– Пропонуйте ідеї;
– Слухайте активно[20].

Опитування

Опитувальні листи — один із способів обстеження великих груп користувачів з метою з'ясування їхніх потреб. Це недорого, добре підходить для збору інформації у великих спільнотах користувачів і їх легко використовувати навіть якщо користувачі далеко рознесені географічно.

Результати аналізу опитувальних листів можна виклоистати як підготовку до застосування інших методів виявлення вимог. Наприклад, опитувальні листи можна застосовувати для виявлення наболілих проблем користувачів у існуючій системі, а результати використовувати при обговоренні

пріоритетів з відповідальними за прийняття рішень на семінарі. Опитувальні листи також можна використовувати для отримання відгуків користувачів серійного продукту.

Найскладніше у опитувальних аркушах — поставити правильні запитання. Існує багато рекомендацій щодо написання опитувальних листів, нижче буде перераховано найважливіші:

– варіанти відповідей мають охоплювати весь діапазон можливих відповідей;

– варіанти відповідей повинні бути взаємовиключними (ніяких накладань у числових діапазонах) та вичерпними (наведіть усі можливі варіанти та/або додайте вільне місце, куди можна вписати варіант, про який ви не здогадалися);

– питання не повинно мати на увазі отримання «правильної» відповіді;

– шкали величин потрібно використовувати однаково у всьому опитувальному листі;

– використовуйте закриті питання з двома або більшими варіантами, якщо опитувальний лист призначений для статистичного аналізу. Відкриті питання дозволяють користувачам відповідати так, як їм хочеться, що ускладнює пошук загальних моментів у результатах;

– варто проконсультуватися зі спеціалістом зі складання матеріалів та проведення опитувань, щоб переконатися, що ви ставите правильні питання «правильним» людям;

– завжди тестуйте опитувальний лист до його поширення. Дуже прикро надто пізно виявити, що питання було поставлене неоднозначно або якесь питання взагалі забули додати;

– не потрібно ставити занадто багато запитань, тому що люди просто не відповідатимуть[20].

Спостереження

Якщо ви попросите користувачів описати, як вони виконують свою роботу, їм, напевно, буде важко бути точним — деталі можуть бути відсутніми або некоректними. Часто це відбувається через те, що завдання складні і кожну дрібницю не згадаєш. В інших випадках причина в тому, що користувачі довели виконання завдання даних до такого автоматизму, що не в змозі сформулювати, що вони точно роблять.

Тому іноді можна дізнатися багато, спостерігаючи за тим, як користувачі в реальності виконують свої завдання. Спостереження займають багато часу, тому вони не підходять для кожного користувача чи завдання. Щоб не порушувати щоденну роботу користувачів, необхідно обмежити спостереження двома годинами або менше. Для спостереження варто обрати важливі чи високо ризиковані завдання та множинні класи користувачів.

При спостереженні в проектах гнучкої розробки варто по спостерігати за користувачами та лише за завданнями, що стосуються майбутньої ітерації. Спостереження робочого процесу користувача в робочому середовищі дозволяє бізнес-аналітику перевіряти інформацію, отриману з інших джерел, визначати нові теми для інтерв'ю, виявляти проблеми з поточною системою та визначати можливості покращення, щоб нова система краще підтримувала робочий процес[20].

Участь клієнта дуже важлива для створення продукту відмінної якості. Необхідно забезпечити, щоб бізнес-аналітик та менеджер проекту із самого початку активно працювали з потрібними представниками клієнта. Якість зібраних вимог до ПЗ, а отже, і успіх ПЗ залежить від того, наскільки голос користувача буде почутий розробниками.

За респондентів було обрано саме прямих користувачів продуктом - працівників кіберполіції, тобто саме вони були визначені, як ключові особи, що надають усі необхідні вхідні дані для складання вимог до ПЗ.

Задля збору вимог було використано 2 методи, а саме інтерв'ю та опитування, також під час проходження виробничої та переддипломної практики було використано методику спостережень, що допомогло зрозуміти істині вимоги користувача, наявні проблеми, потреби, інтереси та інше .

Для цього було складено перелік запитанньв та бланків опитування, на які надавали відповіді 20 працівників головного управління протидії кіберзлочинам НПУ у Дніпропетровській області. Нижче буде подано фрагменти опитувань та інтерв'ю, а також відповіді та результати, які було отримано в ході процедури збору даних.

Частина питань, які були поставленні працівникам кіберполіції підчас опитування:

1. «*Чи вистачає вам у роботі інструментів та технологій, що допомагають ефективно виявляти та протидіяти кіберзлочинам у банківській сфері?*» - на що фактично 90% респондентів надали відповідь, що таких інструментів та технологій їм не вистачає в роботі, результати візуалізовано та показано на рис.3.1.

Рисунок 3.1 Фрагмент опитування, питання №1

2. *«Чи маєте ви розширені бази даних, що містять не тільки особисту інформацію про фізичну особу та її притягнення до кримінальної відповідальності, а і про усі відповідні банківські транзакції/переміщення/зв'язки та ін.?»* - на що 90% респондентів надали відповідь, що вони не мають таких розширених баз даних, а 2% дали відповідь, що бази даних є недостатньо інформативні, результати візуалізовано та показано на рис.3.2:

Рисунок 3.2 Фрагмент опитування, питання №2

3. *«Чи знайомі ви з функціоналом anti-fraud системи? Чи може він бути корисним у вашій роботі?»* - на що 40% кіберполіціантів надали відповідь, що знайомі з функціоналом,та вважають що він може бути корисним у роботі, 35% не знайомі з таким функціоналом, проте хочуть дізнатися більше про нього, 25% знайомі з функціоналом, проте не знають переваг і жоден з респондентів не вважають, що такий функціонал не буде корисним у роботі, наочно результати показано на рис. 3.3.

Рисунок 3.3 Фрагмент опитування, питання №3

4. *«Чи цікава вам зведена статистика вашої активності? (справи в роботі/виконані/необхідно виконати…)»* - на що 95% респондентів надали відповідь, що такий функціонал є цікавим для них, як це показано на рис. 3.4 , і міг би слугувати додтковим джерелом мотивації та самоконтролю і лише 5% (1 респондент) сказав, що такий функціонал не цікавить його.

Рисунок 3.4 Фрагмент опитування, питання №4

Також були задані і інші запитання, відповіді на які були занотовані для збору вимог до додатку.

Також було проведено інтерв'ю стейкхолдерів, в процесі якого задавалися наступні питання:

1. «З якими видами кіберзлочинів ви стикаєтеся найчастіше?»

Усі відповіді, які надали кіберполіціанти, було зібрано в наступний список:

- злочини в банківській сфері;
- комп'ютерні віруси;
- DdoS-атаки;
- спам;
- крадіжки особистих даних.

Причому в усіх випадках першочергово прозвучала відповідь саме про злочини в банківській сфері, а саме фрод, фішинг, скімінг, кеш трапінг, кардинг та інші види кіберзлочинів. Це говорить про те, що дійсно злочини в банківській сфері найбільш розповсюджені і інструменти боротьби з ними є актуальними та необхідними.

2. *«Які методи чи інструменти на вашу думку допомогли б допомогти вам в боротьбі з кіберзлочинністю?»*. На що отримала наступні варіанти відповіді, схожі було згруповано:

- Доступи до різноманітних даних, шляхом централізації та інтеграції комп’ютерних банків даних;
- Розбудова та широке використання ефективних та потужних комп’ютерних мереж та додатків;
- Застосування спеціалізованих засобів захисту інформації.

Причому велика кількість кіберполціантів підкреслювала той факт, що в своїй роботі, нажаль, вони мають дуже обмежений об’єм даних і не мають доступів до зовнішніх джерел даних, або ж не мають власного обширного банку даних про всіх фізичних та юридичних осіб. Пошук і збір усіх даних іноді займає дуже великий проміжок часу, а іноді і зовсім не має результатів і часто справа «заморожується».

3. «Які функції додатку були б корисними при боротьбі з кіберзлочинами в банківській сфері?». На що отримала наступні відповіді:

- Детальна та структурована інформація по підозрілим транзакціям;
- Повна інформація про фізичну або юридичну особу, що потенційно скоїла неправомірні дії ;
- Візуалізація залучених до шахрайських дій усіх сутностей;
- Можливість переглядати архівні дані задля перевірки усіх можливих зв'язків та закономірностей.

4. «Яка інформація по підозрілим транзакціям необхідна вам для збору доказів злочину?»

- Група обмежень транзакції (нетипова сума переводу, IP, стоп-лист та ін.);
- Метрики оцінки підозрілості транзакції;
- Виконавець та залучені до транзакції особи;
- Ресурс через який проведено транзакцію;

Як пояснили респонденти, саме ці параметри можуть допомогти кіберполіціантам у швидкому розкритті злочину, значно скоротять час збору даних, слугуватимуть надалі прямими доказами скоєного злочину.

5. «Яка інформація по залученим сутностям буде необхідною вам для ефективного розслідування?»

- Персональна інформація сутності;
- Банківська активність;
- Оцінка довіри;
- Використані онлайн платформи;

Причому під сутністю мається на увазі як персона, що виконала транзакцію, так і отримувач коштів, а під персональною інформацією мається на увазі як ПІБ, дата народження, ІНПП, місце народження, проживання, так і

історія притягання до відповідальності, створені ЄРДР, штрафи, стягання, правопорушення та інше. Під оцінкою довіри мається на увазі та оцінка, яку їй привласнить класифікатор, задля розуміння, чи становить потенційно ця людина загрозу повторних скоєнь злочинів. Під платформами розуміється джерело, через який могли контактували громадяни з потенційним шахраєм, для подальшого відстеження/блокування та ін.

3.1.2 Діагностика та ідентифікація проблем, дерево проблем

Після проведених інтерв'ю зі стейкхолдерами було виявлено, що найчастіше вимоги до розробки ПЗ виникають саме з наявності проблем із вже існуючим, тому на початковому етапі також важливо діагностувати і ідентифікувати проблеми. Це дозволить зрозуміти який функціонал вже наявний у клієнта і чого їм не вистачає для ефективного виконання поставлених задач.

Діагностика проблем - це аналіз основних причинно-наслідкових зв'язків конкретної ситуації. Існує два способи розгляду проблеми: по-перше, проблемою вважається ситуація, коли поставлені цілі не досягнуті; по-друге, проблемою вважають ситуацію потенційної можливості (щось мало статися, але не сталося). При цьому під ситуацією розуміється реальний стан справ (стан об'єкта управління) щодо поставленої мети.

Діагноз проблеми (ідентифікація) - складний процес, який виконується в кілька етапів:

1. Усвідомлення і встановлення симптомів ускладнень або можливостей.
2. Збір, аналіз зовнішньої (щодо організації) і внутрішньої інформації .
3. Виділення релевантної інформації.
4. Виявлення причин виникнення проблеми, аналіз основної причини.

5. Опис проблеми за допомогою відповідей на питання, що дозволяють менеджерам виявити основні причини подій, що відбулися.

6. Аналіз проблеми.

Дерево проблем

Термін "дерево" в даному контексті припускає використання ієрархічної структури, отриманої шляхом поділу загальної проблематики на основний тип проблематики (стовбур), інші присутні типи (гілки), підтипи (відгалуження) і власне проблеми (листи).

Метод дерева проблем орієнтований на отримання відносно стійкої структури проблематики. Для досягнення цього при побудові початкового варіанта структури враховувалися закономірності і використовувалися принципи формування ієрархічних структур [21].

Виходячи з отриманих відповідей на питання під час опитувань і інтерв'ю було побудовано дерево проблем, що зображено на рис.3.5, де безпосередньо виявлено головну проблему, а саме низький рівень/темпи розкриття кіберзлочинів в банківській сфері.

Як видно на діаграмі, крім головної проблеми, було визначено й інші проблеми, що по версії кіберполіціянтів потенційно впливали на головну, а саме: відсутність доступу до даних від банківських установ та розкриття злочинів «наосліп», тобто застарілі методики збору доказів: збір чеків та роздруківок з банків, що дійсно така транзакція була виконана, бюрократичні процеси задля отримання більш розширеної інформації від банківських установ (судові дозволи).

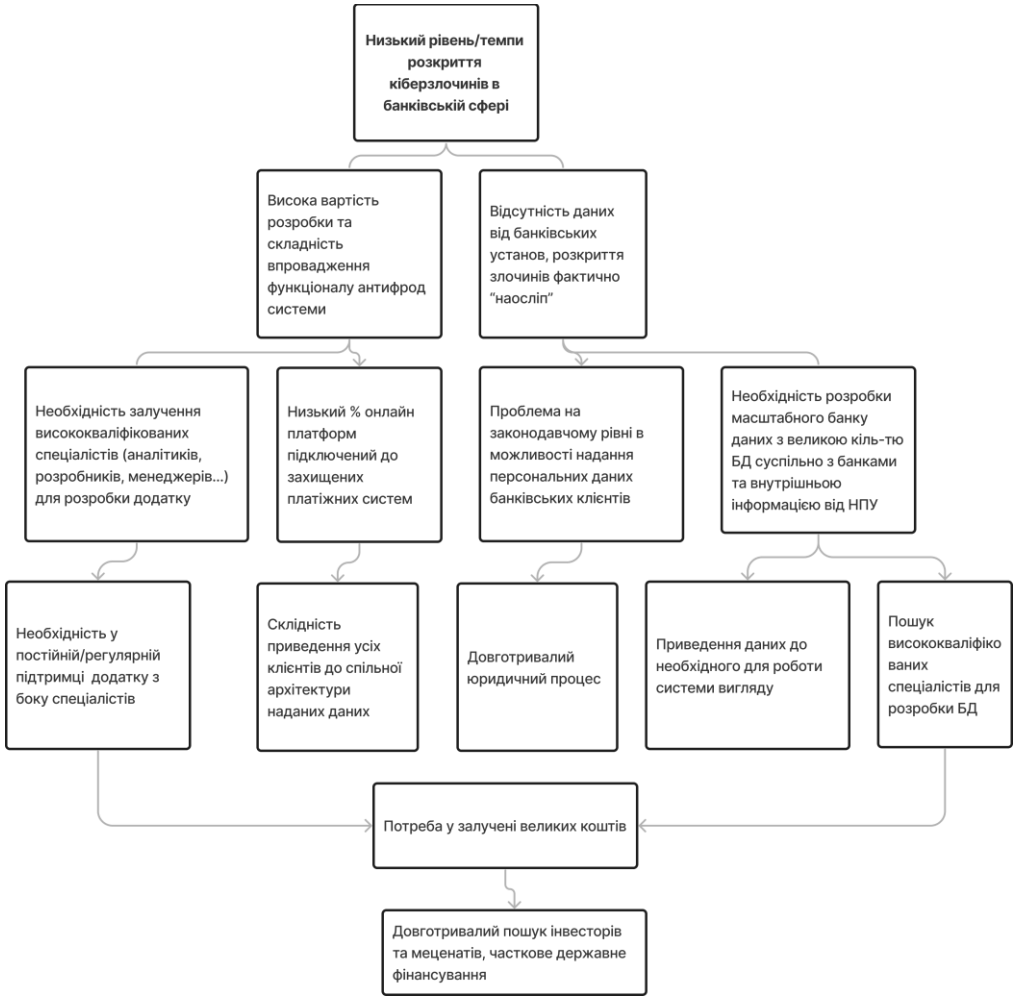

Рисунок 3.5 Дерево проблем

Проте для доступів до банківських даних необхідно усунути наступні проблеми, такі як: внести зміни на законодавчому рівні про банківську таємницю та надати дозвіл на перегляд інформації кіберполіціянтами, а також створення спільних масштабних БД з банківськими установами, щоб кіберполіцейські мали змогу в будь-який момент отримувати результати на запити до БД через зручний користувацький функціонал. Проте ці проблеми спіткають й інші під проблеми, такі як необхідність залучення висококваліфікованих спеціалістів, довготривалі судові процеси, залучення великих коштів на розробку та ін.

Проте після створення такого дерева проблем досить зручно буде розробити детальний план їх рішення, що в свою чергу і буде фактично містити головні етапи по розробці додатку протидії кіберзлочинам в банківській сфері.

3.1.3 Stakeholder Map

Stakeholder Map — це візуальний процес розміщення всіх зацікавлених сторін продукту, проекту чи ідеї на одній карті. Основна перевага карти зацікавлених сторін полягає в тому, щоб отримати візуальне представлення всіх людей, які можуть вплинути на ваш проект, і те, як вони пов’язані. Іноді люди плутають зацікавлених сторін з акціонерами. Хоча акціонери володіють часткою публічної компанії (через акції) і зацікавлені в ефективності компанії, це не означає, що вони повинні бути зацікавленими сторонами кожного проекту чи продукту, запущеного компанією. Зацікавлені сторони можуть працювати на більш детальному рівні, і вони також часто зацікавлені в ефективності проекту чи продукту, не лише тому, що це впливає на ефективність акцій компанії.

На яких етапах, Stakeholder Map є необхідною?

1. Побудова продукту. Створюючи новий продукт з нуля, вам потрібно буде знати зацікавлених сторін для різних груп. Кількість і ролі зацікавлених сторін можуть відрізнятися залежно від типу продукту, над яким ви працюєте.

Можливий список груп потенційних зацікавлених сторін: клієнти/користувачі, галузі / ринки, постачальники, інвестори.

2. Проникнення на ринок.

3. Початок нового проекту. Для початку нового проекту також знадобляться внутрішні зацікавлені сторони, що може складатися з наступних осіб: керівник проекту, розробник, дизайнер, генеральний директор/керівник рівня С і тд.

Чотири кроки до створення карти зацікавлених сторін:

1. Мозковий штурм: визначення всіх потенційних зацікавлених сторін.

2. Категоризація: групування визначених зацікавлених сторін.

3. Розстановка пріоритетів, ієрархія.

4. Комунікації із зацікавленими сторонами.

Нижче було розроблено наступне дерево зацікавлених осіб, що показане на рис. 3.6 у вигляді ієрархії прямих клієнтів, користувачів та зацікавлених осіб. Також було розділено користувача «кіберполіцію» на окремі особи, що будуть безпосередньо та регулярно приймати участь у забезпечені життєдіяльності та правильної роботи ПЗ та функціоналу класифікатору, а саме:

– старший оператор (відповідальна особа – кіберполіцейський, прямий користувач, що відслідковує процес роботи чергового над обробкою вхідних оповіщень про шахрайські дії, правильність роботи додатку, розподіляє навантаження між працівниками та ні.);

– черговий оператор (прямий користувач, кіберполіцейський, що приймає на вхід сповіщення про шахрайські дії, бере їх в роботу та безпосередню обробку, проводить розслідування);

– адміністратор БД (технічний працівник, забезпечує правильне функціонування, запис, зчитування даних та виправляє несправності);

– команда розробників проекту (технічні працівники, спеціалісти, що забезпечуватимуть безперебійну та коректну роботу як скрипту функціоналу класифікатора, так і роботи самого додатка)

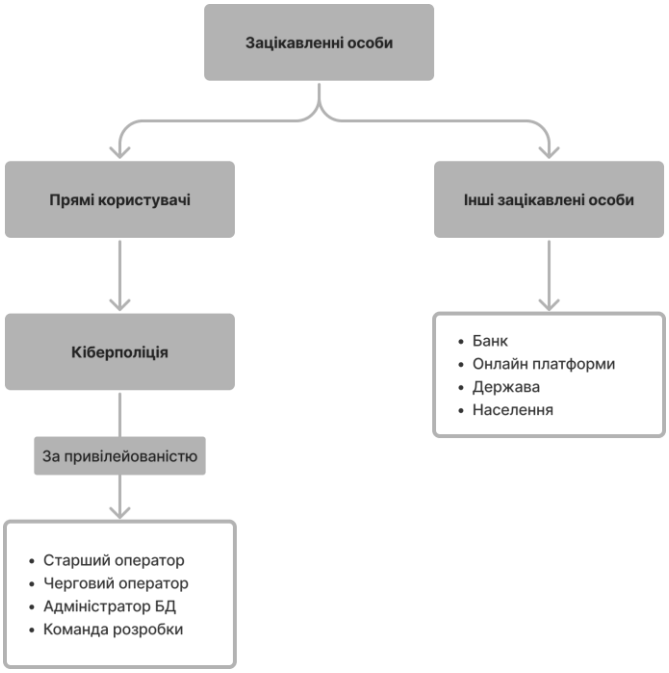

Рисунок 3.6 Ієрархія клієнтів, користувачів та зацікавлених осіб

Також на діаграмі було визначено і іншиих зацікавлених осіб, що мають відношення до додатку протидії кіберзлочинам в банківській сфері, а саме:

- банк,
- онлайн платформи (мається на увазі усі потенційно можливі платформи продажу товарів та послуг, соціальні мережі, сайти та інше),
- держава,
- населення.

Також було використано методологію виділення головних стейкхолдерів, з визначенням їх вигоди, очікувань, інтересів та обмежень, а також занотовано основні моменти у вигляді таблиці 3.1:

Таблиця 3.1

Stakeholders	**Вигода**	**Очікування**	**Інтереси**	**Обмеження**

Кіберполіція	- Підвищення довіри населення НПУ.	- Швидкий розгляд та розкриття кіберзлочинів та шахрайських схем	- Розширення можливостей розкриття кіберзлочинів.	- Необхідність залучення профільних спеціалістів (аналітики, розробники, оператори)
Банки	- Зниження навантаження на працівників відділу моніторингу. - Підтримка НПУ в питаннях кіберзлочинів в банківській сфері.	- Колективна співпраця з НПУ з питань фродових операцій	- Скорочення витрат на впровадження Antifraud систем.	- Передача певних повноважень та даних клієнтів НПУ
Онлайн платформи	-Підключення до захищених платіжних систем. - Можливість безоплатного використання Antifraud системи.	- Підвищення довіри користувачів	- Скорочення витрат на впровадження Antifraud систем. - Підвищення рейтингу на ринку надання послуг.	- Невідповідність вимогам Antifraud системи. - Унеможливлення передачі необхідних даних НПУ.
Держава	- Зниження розмірів щорічних збитків від кіберзлочинів. -Залучення інвесторів	- Підвищення довіри населення до НПУ	- Цифрова трансформація банківської сфери та кіберполіції.	- Недостатня кількість ресурсів на розробку та впровадження системи. - Необхідність внесення змін до чинного законодавства.
Населення	- Зниження ризиків втратити власні кошти	- Зниження ймовірності натрапити на шахраїв	- Скорочення кількості етапів розгляду звернення з приводу кіберзлочинів.	- Додаткові персональні верифікації на онлайн платформах.

З цієї таблиці бачимо, що опитування та інтерв’ю користувачів та зацікавлених осіб допомогли зібрати усю необхідну вхідну інформацію для

побудови ієрархії клієнтів, користувачів та зацікавлених осіб, а також визначити їх вигоди, очікування інтереси та обмеження, що допоможе в розробці усіх необхідних функцій додатку.

3.1.4 User Stories

User Story (історія користувача) – це неформальне загальне пояснення функцій програмного забезпечення, написане з точки зору кінцевого користувача. Її мета полягає в тому, щоб сформулювати, яку цінність для замовника несе функціонал програмного забезпечення. Вона формулює не тільки бізнес-цінність, а й вимоги для розробки й тестування. User Story називається так, бо вона створюється через розповідь, як історія.

Ключовим компонентом гнучкої розробки програмного забезпечення є ставлення людей на перше місце, а історія користувача ставить кінцевих користувачів у центр всього процесу. Ці історії не використовують технічну мову, щоб створити контекст для команди розробки продукту. Прочитавши історію користувача, команда розуміє, що вони роблять, чому вони це роблять і яку цінність має продукт, який вони створюють.

Сам формат можна викласти в короткому реченні на кшталт: «Як персона, я хочу, так що...» (As a [person], I [want to], [so that]). Так користувач/замовник пояснює, що саме він хоче і навіщо.

Далі розглянемо цей шаблон більш детально, щоб стало зрозуміліше, як саме повинна звучати User Story:

«Персона» – для кого ми це створюємо?

«Хочу» – тут описуються наміри людей, а не функції, які вони використовують.

«Так що» – яка мета, чи яку проблему потрібно вирішити? Якої загальної вигоди потрібно досягти? Яку проблему необхідно вирішити?

Текст самої User Story повинен пояснювати роль/дії користувача в системі. Попри те, що історія користувача відіграє роль, яка раніше належала специфікаціям вимог, сценаріям використання тощо, вона все ж відчутно відрізняється рядом тонких, але критичних нюансів:

– User Story не є детальним описом вимог;

– вона коротка та легко читається, зрозуміла розробникам, стейкхолдерам і користувачам;

– являє собою невеликі інкременти цінної функціональності, які можуть бути реалізовані в рамках декількох днів або тижнів;

– зусилля, що необхідні для реалізації, можуть бути швидко визначені;

– історія користувача не міститься у величезних, громіздких документах, а організована у списки, які легше впорядкувати й перевпорядкувати з надходженням нової інформації;

– вона не деталізована на початку проєкту;

– вона вимагає мінімум або зовсім не вимагає супроводу і може бути безпечно скасована після імплементації [22].

Отже, по підсумкам проведених інтерв'ю та опитувань, а також за допомогою інших методик збору інформації було складено наступний User Story:

1. Як користувач, я хочу відстежувати статистику роботи визначення підозрілих транзакцій, щоб приймати зважені та ефективні рішення в майбутньому.

2. Як користувач, я хочу отримувати інформацію про залучених до транзакції сутностей, щоб зібрати необхідні докази кіберзлочину та розкрити шахрайські схеми.

3. Як користувач, я хочу мати доступ до мережі пов’язаних сутностей, щоб поглибитись у проблему та визначити потенційні схеми злочину.

4. Як користувач, я хочу мати доступ до архіву даних, щоб звертатися до історії попередніх транзакцій.

5. Як користувач, я хочу знати за якими параметрами система оцінює транзакцію або сутність, щоб розрахувати ступінь ймовірності шахрайства.

6. Як користувач, я хочу бачити як система оцінила транзакцію та/або сутність, щоб протидіяти конкретному типу шахрайству.

7. Як користувач, я хочу бачити як система оцінила транзакцію та/або сутність, щоб визначати паттерни поведінки шахраїв.

Отже, підводячи підсумки, можна сказати, що User Story який ми отримали - короткий і максимально зрозумілий опис функціоналу продукту або його особливостей чи користі, яку хоче отримати користувач. Тобто в цих реченнях ми можемо побачити який саме функціонал необхідний користувачу, а також для чого він, які потреби він має задовольняти: а саме впровадження аналітичних статистик, доступів до архівів даних, до розширених баз даних, а також безпосередньо бачити роботу класифікатору. Це допоможе розробнику краще зрозуміти продукт, область й аудиторію замовника. Затребуваність такого формату оцінюється виявленням і опрацюванням призначених для користувача потреб, продукт на виході повинен їх задовольняти.

3.2 Розробка схеми бази даних додатку

Модель клієнт-серверної архітектури.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними.

Функції, які реалізуються на сервері:

– зберігання, доступ, захист і резервне копіювання даних;
– обробка клієнтського запиту;
– відправлення результату (відповіді) клієнту.

Функції, які реалізуються на стороні клієнта:

– надання користувальницького інтерфейсу;
– формулювання запиту до сервера і його відправка;
– отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних).

Архітектура клієнт-сервер визначає принципи спілкування між комп'ютерами, а правила і взаємодії визначені в протоколі [23].

Нижче, на рисунку 3.7 зображено модель клієнт-серверної архітектури, проте банк віднесено не до прямого користувача (клієнта), а до зовнішнього, так як запити будуть відправлятися саме з комп’ютера кіберполіціянта, проте він в свою чергу може здійснювати обмін даними роботи класифікатору транзакцій з зовнішнім клієнтом, у нашому випадку узагальнено названим «Банк», проте який повинен надавати базі даних банківські дані, а поліція - внутрішні поліцейські дані.

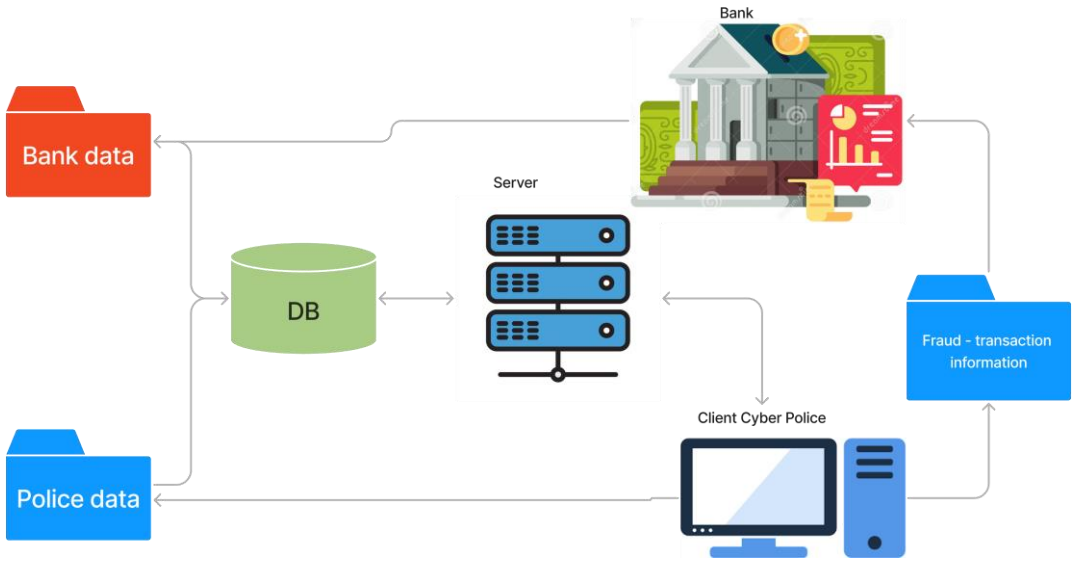

Рисунок 3.7 Модель клієнт-серверної архітектури

Причому вище зображена саме дворівнева архітектура, що складається з двох вузлів:

– сервер, який відповідає за отримання запитів і відправку відповідей клієнту, використовуючи при цьому лише власні ресурси;

– клієнт, який представляє користувацький інтерфейс.

Принцип роботи полягає в тому, що сервер отримує запит, обробляє його і відповідає безпосередньо, без використання сторонніх ресурсів. За потреби дану модель можна буде змінити на трирівневу або багаторівневу, де запит клієнта буде обробляти одночасно декілька серверів. Це буде необхідно якщо буде виникати навантаження на сервер.

Логічна схема БД

Якість розробленої БД цілком залежить від якості виконання окремих етапів її проектування. Величезне значення має якісна розробка логічної моделі даних, так як вона, з одного боку, забезпечує адекватність бази даних предметної області, а з іншого боку, визначає структуру фізичної БД і, отже, її експлуатаційні характеристики. Одні і ті ж дані можуть групуватися в таблиці-відносини різними способами, тобто можлива організація різних наборів відносин взаємопов'язаних інформаційних об'єктів предметної області. Угруповання атрибутів у відносинах повинна бути раціональною, що гранично скорочує дублювання даних і спрощує процедури їх обробки та оновлення. Певний набір відносин володіє кращими властивостями при включенні, модифікації і видалення даних, якщо він відповідає конкретним вимогам нормалізації відносин.

Нормалізація відносин - формальний апарат обмежень на їх формування, який дозволяє усунути дублювання даних, забезпечити їх несуперечливість і зменшити витрати на підтримку БД. На практиці найбільш часто використовуються поняття першої, другої і третьої нормальних форм [24].

Нижче, на рис. 3.8 зображена ER-діаграма - модель, яка найчастіше використовуються для розробки або налагодження реляційних баз даних в областях, бізнес-інформаційних системах і дослідженнях, де реляційна база даних — це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

Важливо відзначити, що для задоволення усіх потреб клієнта було визначено, що база даних додатку повинна містити 8 таблиць: *користувач* (інформація про поліціанта), *кейс* (інформація про фродову транзакцію), *сущність* (інформація про особу що виконала/приймала операцію), *індивід* (персональна інформація про фіз.особу), *сполучені сущності* (друга сторона приймач/відправник операції), *банк* (інформація про будь-які зв’язки з банками сутності), *транзакція* (детальна інформація про кожну транзакцію), *фічі* (фічі транзакції, попередньо опрацьовані системою антифроду, що містять певну оцінку по кожній фічі транзакції).

Важлива відзначити, що в структурі БД враховано побажання замовника, та впроваджено залучення збору даних від банків та запис і під’єднання даних, що містить внутрішня поліцейська база по кожній особі.

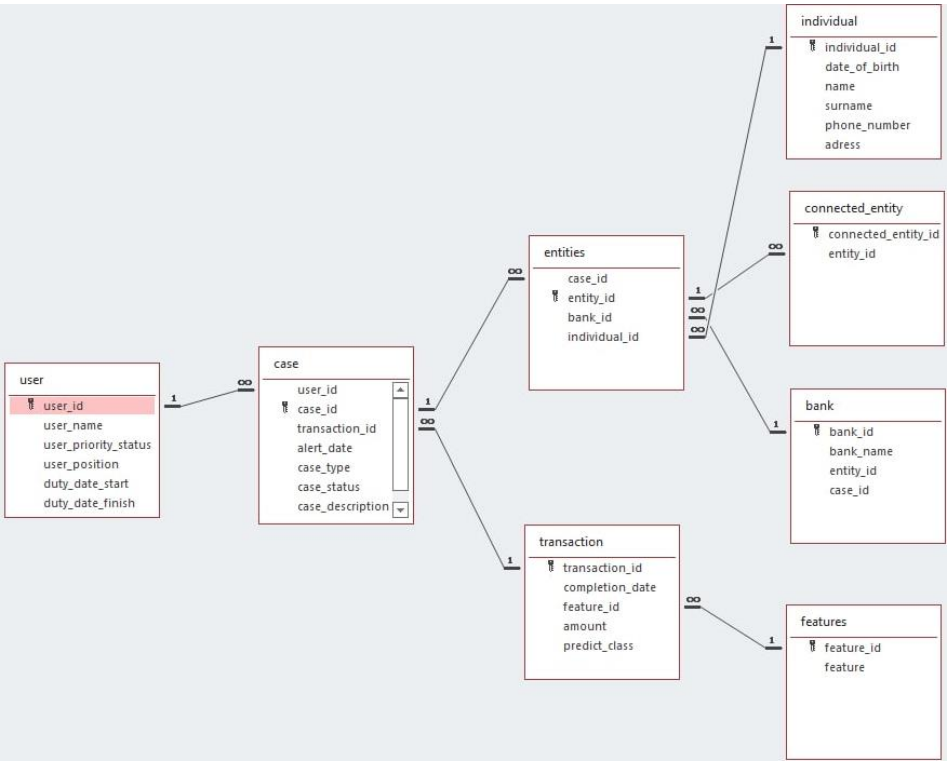

Рисунок 3.8 Розробка логічної схеми бази даних

3.3 Моделювання елементів алгоритму роботи додатку

Для швидкого й ефективного розроблення ПЗ потрібно залучати високопрофесійну команду розробників, вибрати правильні методи та інструменти роботи. Необхідно, щоб процес розроблення був ретельно продуманий і адаптований до мінливих потреб користувачів, технології розроблення та життєвого циклу (ЖЦ) системи.

Основою створення якісного ПЗ є моделювання, що дозволяє:

– наочно продемонструвати бажану структуру і поведінку системи;

– здійснювати візуалізації та керування її архітектурою;

– забезпечити краще розуміння створюваної системи, що найчастіше призводить до її спрощення й забезпечує можливість повторного використання;

Моделювання програмного забезпечення є усталеною і загальноприйнятою інженерною методикою. Розробляють архітектурні моделі

будинків, аби допомогти їхнім майбутнім мешканцям у всіх деталях уявити готовий об’єкт.

Також застосовують і математичне моделювання об’єктів, що полягає в урахуванні впливу різних фізичних навантажень (сильного вітру, вібрацій автостради, землетрусу тощо). Моделювання застосовується не тільки в будівництві, а й в інших галузях

Модель - спрощений вигляд реальності, креслення системи, куди може входити як детальний план, так і більш абстрактний вигляд, так би мовити, «з висоти пташиного польоту». Якісна модель завжди включає елементи, які суттєво впливають на результат і не включає ті, які малозначні на даному рівні абстракції.

Кожна система може бути описана з різних точок зору, для чого використовуються різні моделі, кожна з яких є семантично замкненою абстракцією системи. Модель може бути структурною, що підкреслює організацію системи, або моделлю поведінки, що відображає її динаміку.

Модель ПЗ будується для того, щоб краще розуміти розроблювану систему. Моделювання дозволяє вирішити чотири різні завдання:

1. Візуалізувати систему в її поточному або бажаному для замовника стані.

2. Описати структуру або поведінку системи.

3. Отримати шаблон, що дозволяє сконструювати систему.

4. Документувати прийняті рішення, використовуючи отримані моделі.

Чим більша і складніша система, тим більшого значення набуває моделювання при її розробленні. Без моделі складну систему неможливо сприйняти як одне ціле (навіть програмний еквівалент собачої будки суттєво виграє від застосування моделювання).

Людське сприйняття складних сутностей є обмеженим. Моделювання системи чи об’єкта дозволяє звузити проблему, зосередивши увагу в кожен

момент тільки на певних її аспектах (принципу «розділяй і володарюй»). Складне завдання завжди легше вирішити, якщо розділити його на менші.

Моделювання підсилює можливості людського інтелекту. Правильно обрана модель дозволяє створювати проекти на вищих рівнях абстракції [25].

3.3.1 Інформаційна архітектура додатку

Інформаційна архітектура — це перш за все ефективне управління вмістом, чітко визначені процедури та інформаційна політика сайту, додатку, тощо. Інформаційна архітектура є складним процесом, який включає в себе багато інших галузей знання.

Існує багато визначень інформаційній архітектури, наприклад:

1) Поєднання схем організації, предметизації і навігації, реалізованих в інформаційній системі.

2) Структурне проектування інформаційного простору, що сприяє виконанню завдань і інтуїтивному доступу до вмісту.

Проведення кордонів інформаційної архітектури виявляється ще більш нетривіальним завданням.

Наприклад, деякі речі визначені але не відносяться до інформаційної архітектури:

– Графічне оформлення НЕ є інформаційною архітектурою.

– Розробка програмного забезпечення НЕ є інформаційної архітектурою.

– Проектування юзабіліті НЕ є інформаційною архітектурою.

Тобто інформаційна архітектура – це потужний інструмент для знайомства зі складною багатовимірної природою інформаційних просторів. Як і будівлі, вебсайти/додатки володіють архітектурою, що визначає нашу роботу з ними. Деяким додаткам властива логічна структура, що дозволяє нам знайти

відповідь і виконати завдання, в інших же відсутня яка-небудь логічна організація, і спроби навігації по ним виявляються марними.

Можливість знайти інформацію та необхідний функціонал — вирішальний фактор для зручності додатку в цілому.

Нижче, на рис.3.9 зображено загальний вигляд компонентів (сторінок) додатку:

Рисунок 3.9. Загальне представлення рівнів додатку

Як бачимо, загальна структура верхніх рівнів додатку виглядає наступним чином:

1) Найвищий рівень – Antifraud – початкова сторінка, заставка, наявність функції авторизації.

2) Home – в цьому блоці користувач – кіберполіянт потрапляє на особисту домашню сторінку з загальною статистикою та інформацією.

3) Alert - у цьому блоці описано таблицю з активними (готовими до розгляду оператором) транзакціями

4) Search – блок, що надає можливість користувачу виконати пошук за певним критерієм.

Наступний рівень складається з 2 блоків:

1) Allert – блок, що істить більш детальну інформацію по транзакціям.

2) Individual’s info – блок, що містить інформацію про індивіда, що зустрічається на шляху того, чи іншого етапу проведення транзакції.

Розглянемо більш детальну інформацію про архітектуру додатку по кожному з блоків:

Блок Home

Фактично, це головна сторінка, на яку потрапляє користувач, після процедури логіну в додатку. Вона містить наступні блоки більш низького рівня, які зображено на рис. 3.10:

1) Alert Summary – загальна інформація по сповіщеню про фродову операцію, яка в свою чергу містить інформацію про:

- тип кейсу (перевищеня лімітів, нетипова локація, перевищеня суми і тд.)
- кіль-ть таких операцій по певному типу кейсу
- час розляду операції

2) Personal Metrics – сторінка з особистими показниками поліціанта, включає в себе:

- період (день/тиждень/місяь.. за який надається статистика)
- дія (вжиті за вибраний період дії)
- середній час роботи (мається на увазі час від моменту взяття в роботу до моменту закриття дії)

3) нещодавно переглянуті (будь що: особи/справи/транзакції, буде розділено по різним підтемам для зручності, і виводитися в кожній по 5 останніх).

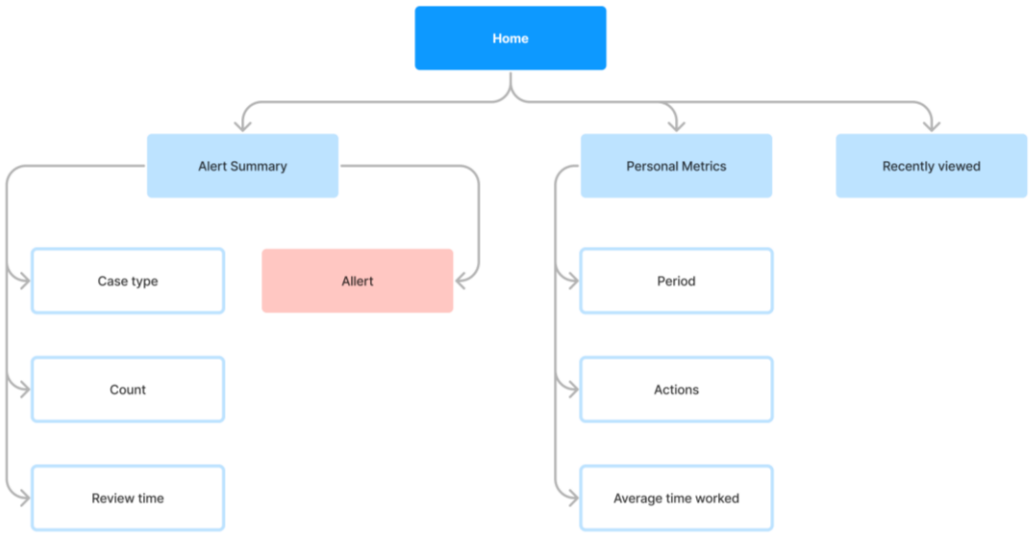

Рисунок 3.10 Архітектура блоку Home

Блок Alerts

У цьому блоці описано таблицю з активними (готовими до розгляду оператором) транзакціями. Вона містить наступні блоки більш низького рівня, які зображено на рис. 3.11:

1) Alerts Table – глобальна таблиця із сповіщеннями про підозрілі транзакції, яка в свою чергу містить інформацію про:

– оцінку транзакції від класифікатору;

– дату;

– ID пригоди;

– Тип пригоди;

– Статус пригоди;

– Опис пригоди (можливо якісь повідомлення від банку);

– Хост (ім’я виконавця транзакції, проте через це поле можна перейти до більш детальної інформації про виконавця)

– Alert – перехід до іншого блоку додатку з більш детальною інформацією про сповіщення про фродову операцію.

2) Alert individuals (Preview) – попередня інформація про інидвіда, доступна лише якщо обрано якийсь кейс, включає в себе:

– оцінка довіри (оцінка довіри класифікатору/банку до індивіду)

– ім’я

– тип індивіду (відправник/отримувач)

– локація (звідки була виконана операція)

– банківський аккаунт/рахунок індивіда

3) Мережа (попередній перегляд) попередня інформація про індивіда та потенційні зв’язки з іншими особами.

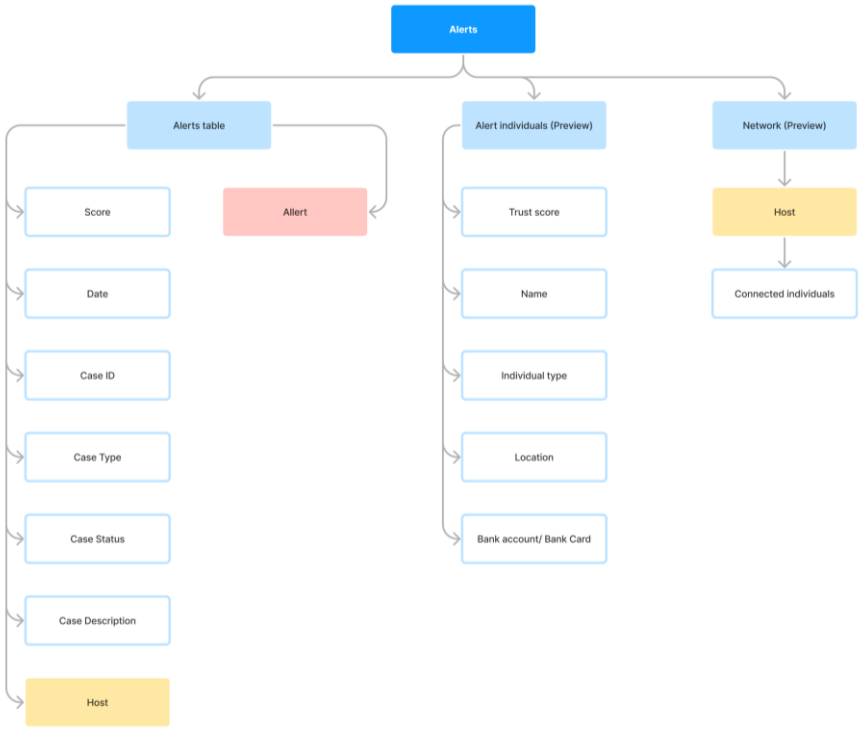

Рисунок 3.11 Архітектура блоку Alerts

Блок Search

У цьому блоці користувачу надається можливість пошуку необхідної інформації. Вона містить наступні блоки більш низького рівня, які зображено на рис. 3.12:

1) Пошук – пошукова стрічка.

2) Випадок– пошук інформації про унікальні випадки, включає в себе:

– Дата
– ID випадку;
– Тип фроду;
– Статус випадку;
– Опис випадку;
– Оператор, що розглядав пригоду (поліцейський);
– Хост (інформація про індивіда, можна перейти по цьому полю до більш розгорненої інформації).

Рисунок 3.12 Архітектура блоку Search

Блок Alert

У цьому блоці надається детальний опис конкретної транзакції.

На неї можна потрапити шляхом переходу з блоку Alerts Table або з Alert Summary, та містить наступні блоки більш низького рівня, які зображено на рис. 3.13:

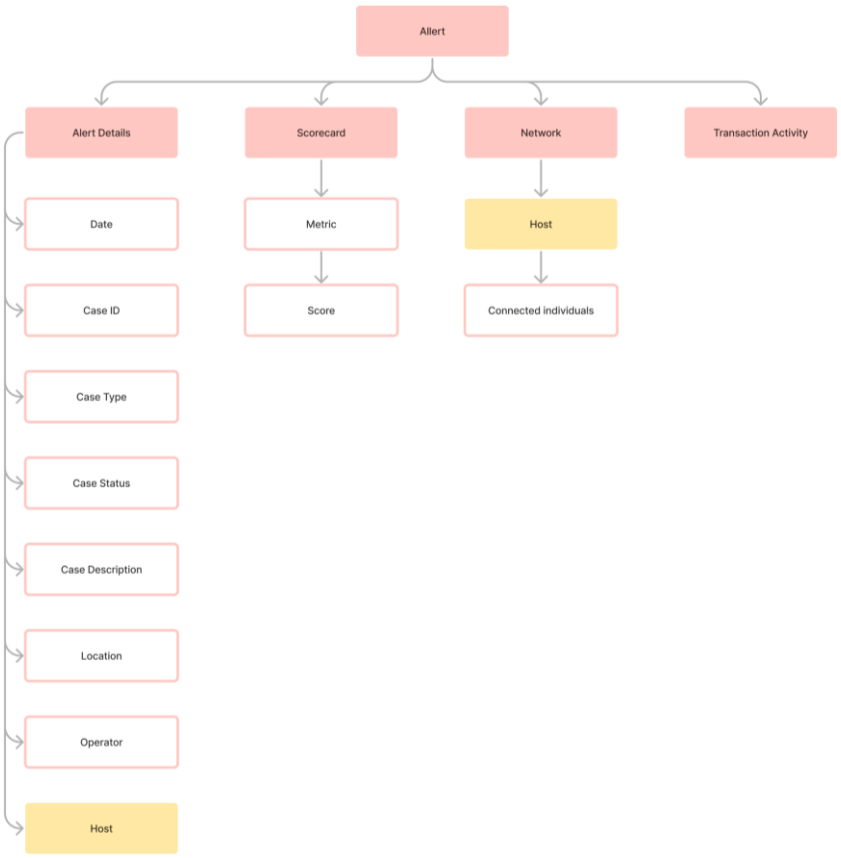

Рисунок 3.13 Архітектура блоку Alert

1) Alert Details – у вигляді таблиці надається уся детальна інформація про підозрілі транзакції, яка в свою чергу містить інформацію про:

- дата проведення;
- ID пригоди;
- Тип пригоди;
- Статус пригоди;
- Опис пригоди (можливо якісь повідомлення від банку);
- Локація (звідки відбулася транзакція)
- Оператор (хто працює над випадком)
- Хост (ім’я виконавця транзакції, через це поле можна перейти до більш детальної інформації про виконавця)

4) Scorecard – метрики, які алгоритм антифроду використовує для визначення підозрілої транзакції, включає в себе:

- метрики
- оцінки метрики

5) Мережа: інформація про індивіда та потенційні зв’язки з іншими.

6) Активність по транзакції: остання активність по транзакції.

Блок Individual’s Info

У цьому блоці надається повна інформація про індивіда. Важливо, що інформація збирається не тільки з банківських джерел, а й з закритої поліцейської бази даних. На неї можна потрапити шляхом переходу з блоку з будь якого попереднього блоку додатку, та містить наступні блоки більш низького рівня, які зображено на рис. 3.14:

1) Персональна інформація – у вигляді таблиці надається уся детальна персональна інформація про індивіда, яка в свою чергу містить інформацію про:

- Унікальний ID в базі;
- ПІБ;
- Стать
- IBAN;
- Локація (звідки відбулася транзакція)
- Оператор (хто працює над випадком)
- Хост (ім’я виконавця транзакції, через це поле можна перейти до більш детальної інформації про виконавця)

2) Адреса – детальна інформація про локацію індивіда, включає в себе:

- Країна;
- Місто;

– Вулиця;
– ІНП.

3) Активність: історія активностей індивіда (притягнення до кримінальної відповідальності і тд.), в свою чергу містить інформацію про:

– Дату притягнення;
– Який був випадок (статті кримінального провадження і тд);
– Тип випадку;
– Статус (закрита справа/вирок суду і тд)
– Опис випадку (обставини скоєння і тд..)

4) Оцінка довіри: оцінка довіри від класифікатора до особи.

5) Мережі: потенційні зв'язки з іншими сутностями. У вигляді графічного зображення можуть бути як інші сутності, так і соц.мережі, торгові платформи, тощо.

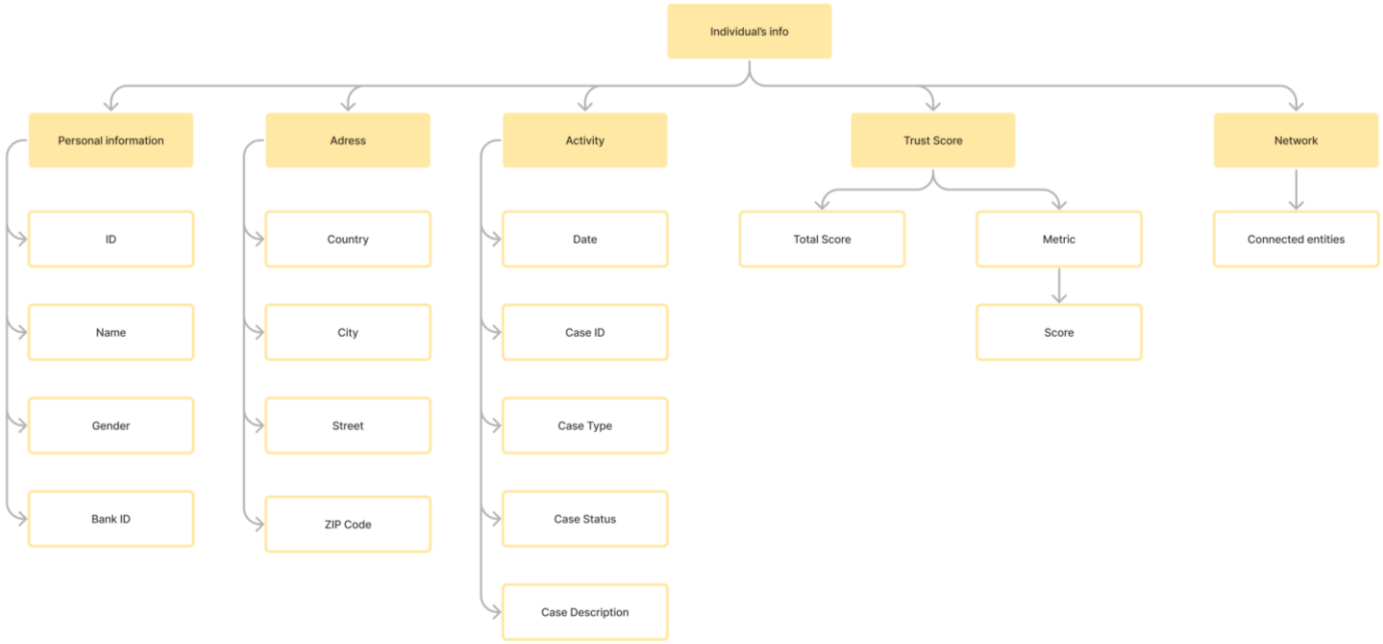

Рисунок 3.14 Архітектура блоку Individual`s Info

В результаті ми отримуємо повноцінну архітектуру додатку, що повинна забезпечувати наявність повного функціоналу, що необхідний для роботи

кіберполіціанту, а також надаватиме змогу коректно в виконати розробку додатку.

3.3.2 User flow

User Flow — це шлях, який проходить користувач через рішення, він демонструє варіанти того, як юзери переміщаються всередині продукту для вирішення своїх.

Компоненти User Flow:

– 

подія, яка визначає початок і кінець шляху користувача (заходить на головну сторінку, підтверджує реєстрацію тощо);

– 

вказує на кроки, які здійснює користувач (шукає інформацію, додає товар в кошик тощо);

– 

вказує на місце, де користувач має зробити вибір чи ухвалити рішення, яке вплине на його подальший шлях;

–  вказує на напрямок шляху користувача.

Для прикладу було розроблено 2 сценарії реагування кіберполіціантом на сповідення про шахрайство в банківській сфері та проілюстровано їх на рис. 3.15: варіант 1 - User Flow за умов до впровадження додатку, та варіант 2 - User Flow за умов використання функціоналу додатку. Як бачимо, в новому User flow вже відсутні фактично 80% бюрократичних кроків по отриманню необхідної для розслідування інформації та доказів злочину. Шлях користувача став на багато коротшим, чітко сформульованим, займає в багато разів менше часу.

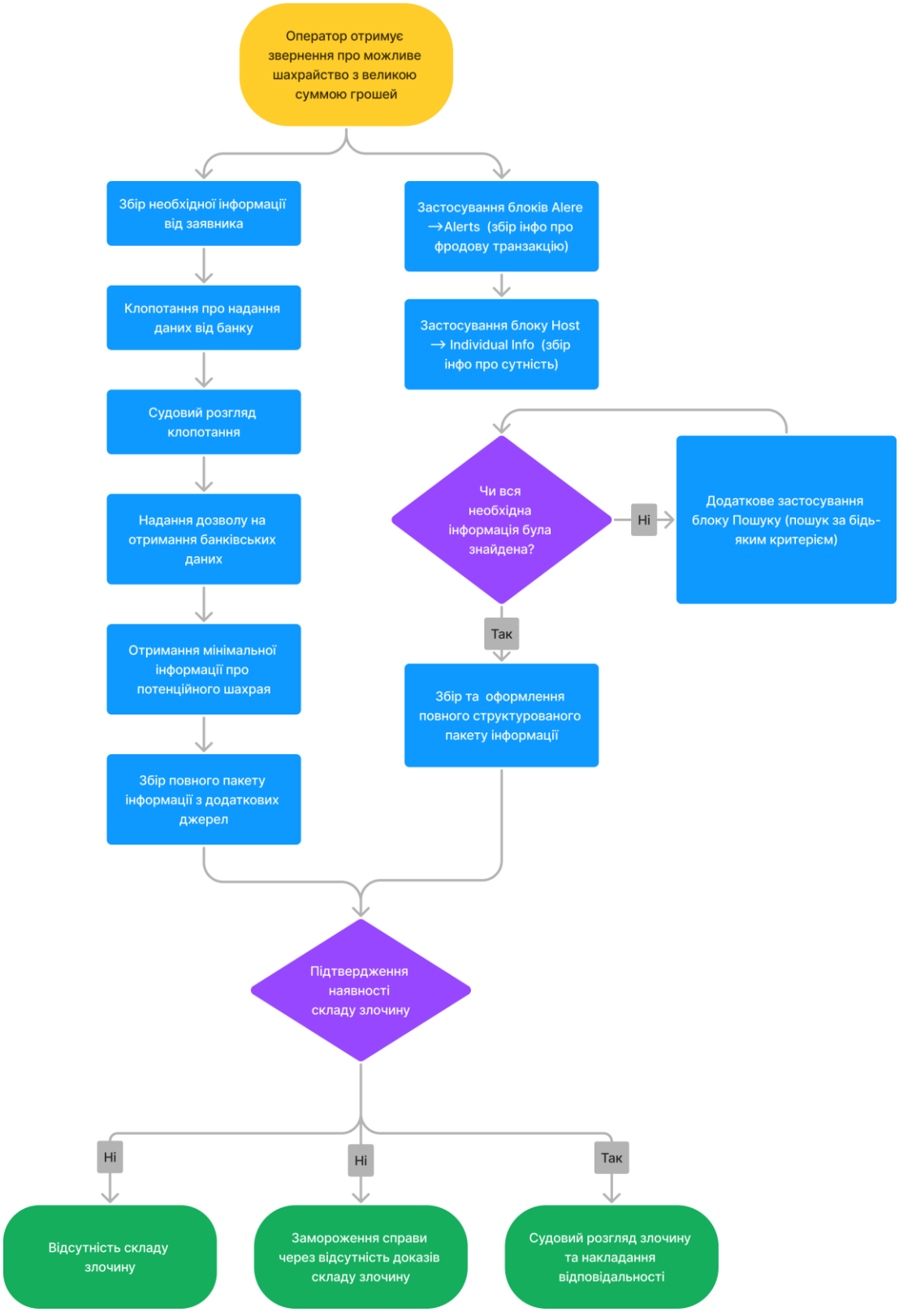

Рисунок 3.15 User flow реагування на оповіщення про шахрайство

3.4 Результати впровадження та оцінка ефективності додатка

Перед створенням каркасів додатку, UI проектуванням, та іншими етапами розробки додатку було проведено тестування задля перевірки, чи добре спланована навігація в додатку або її потрібно скорегувати. Задля цього було створено Tree Test на сайті optimworkshop.com. Дерево відтворювало ту ж саму інформаційну архітектуру, що подавалася в попередньому розділі. Тестування з прямими користувачами полягало в тому, що вони мали певний перелік запитань по яким вони повинні були відшукати певний розділ інформації в карті додатку.

На рис. 3.16 показано як виглядало загальне тестування навігації прямими користувачами, а саме привітання, уточнення, чи є опитувана особа – кіберполіціянтом, і коротка інструкція як правильно надавати відповідь.

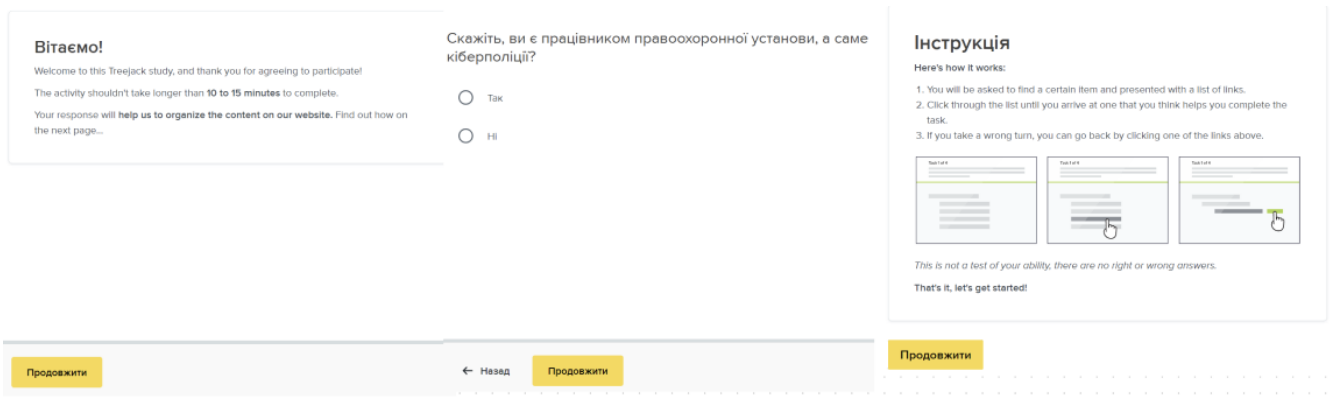

Рисунок 3.16 Тестування навігації в додатку користувачами

Нижче на рис. 3.17 показано результати тестування навігації по додатку. В них прийняло участь 6 учасників – оперуповноважені особи з відділку кіберполіції у Дніпропетровській області ДКП НПУ.

Participant	Status	Time taken	Question responses	Tasks completed	Tasks skipped	Tasks successful	
Participant 1	Completed	3:16	1	100%	0	100%	Actions
Participant 2	Completed	3:29	1	100%	0	100%	Actions
Participant 3	Completed	2:31	1	100%	0	85%	Actions
Participant 5	Completed	4:08	1	100%	0	100%	Actions
Participant 6	Completed	2:51	1	100%	0	100%	Actions
Participant 7	Completed	7:32	1	100%	0	100%	Actions

Рисунок 3.17 Статистика прямих користувачів по навігації додатком

Якщо розглядати більш детальну статистику, то як показано на рис. 3.18 6 6 з 6 опитуваних впоралися з поставленими задачами, середній час пошуку необхідного розділу з інформацією складає 3 хвилини 22 секунди, а 98% мали правильну відповідь.

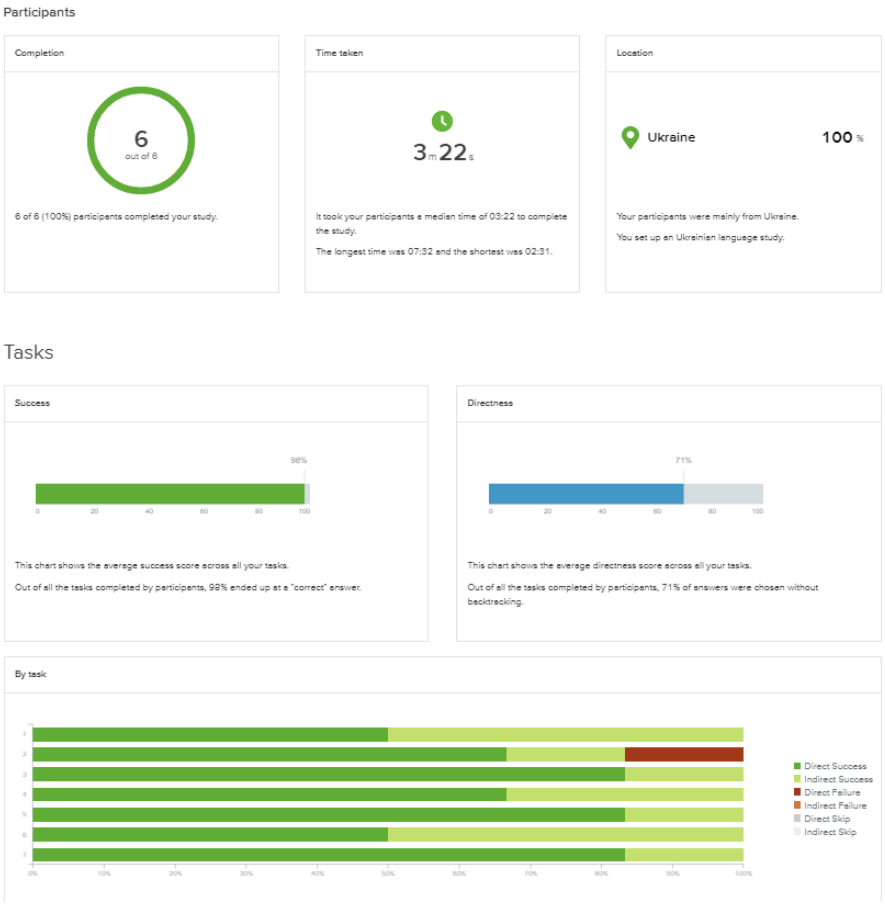

Рисунок 3.18 Результати тестування навігації по додатку

Це тестування дало змогу зрозуміти, що розроблена інформаційна структура додатку є зручною та зрозумілою у використанні, а отже потенційно допоможе кіберполіціянтам швидко відшукувати необхідну інформацію.

Оцінка ефективності додатку

З офіційного звіту НПУ бачимо, що дійсно з кожним роком зростає кількість виявлених кіберзлочинів, як це показано на рис.3.19.

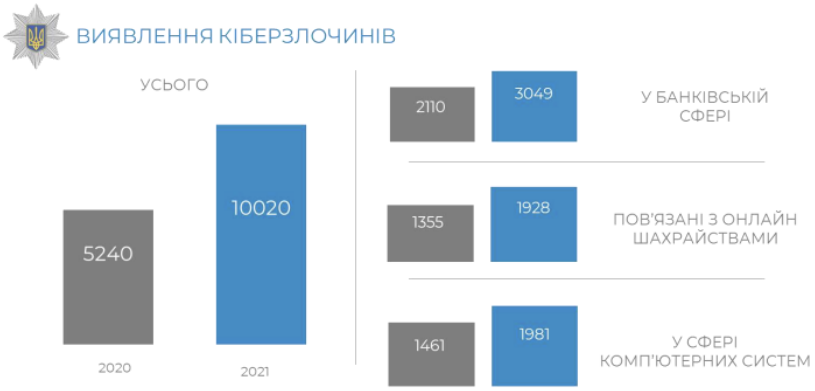

Рисунок 3.19 Кількість зареєстрованих випадків кіберзлочинів

Зокрема, у майже півтора рази зросла динаміка реєстрації злочинів у банківській сфері та на третину - у сфері комп’ютерних систем. При цьому кількість розкритих кіберзлочинів збільшилася вдвічі. Проте не варто стверджувати, що такі темпи задовільнять високий рівень захисту населення від злочинів. У 2021-2022 році українці частіше обирали безготівкові платежі та покупки онлайн, що зумовлювалося як продовженням карантинних обмежень, так і змінами у звичках користувачів платіжних карток. Це стимулювало шахраїв, які не полишали спроб ошукати громадян, зокрема і у віртуальному просторі.

Проте вже за 3 місяці тестування функціоналу вдалося побачити реальний ефект. Як показано на рис. 3.20 бачимо, що за період з вересня по

листопад 2022 року відносно аналогічного періоду в 2021 році спостерігається ріст тотально зареєстрованих кримінальних проваджень на +15%, проте не варто вважати, що це є негативний ефект, адже система класифікатору дозволяє виявляти все більше фродових операцій і автоматично бути доказом ззлочину, тоді як до впровадження системи моніторингу не завжди людина, ошукана злочинцем, зверталася до поліції. Також з важливих позитивних аспектів бачимо, що покращується динаміка по кількості розглянутих справ: +113% в порівнянні з аналогічним періодом в минулому році, та зниження кількості справ що ще очікують розгляду -18%.

Рисунок 3.20 Статистика по кримінальним впровадженням

Динаміка зміни суми збитків, шо показана на рис 3.21 також показує позитивні зміни: з вересня по листопад місцяь 2022 року вдалося скоротити розмір збитків на 30% відносно аналогічного періоду в 2021 році, в числовому еквіваленті це становить 14.5 млн. грн.

Рисунок 3.21 Статистика по збиткам

Найбільшим змінам підлягають зменшення збитків по шахрайським аферам в мережі Інтернет збитки зменшилися майже на 30%, на торгівельних платформах (ОЛХ, Розетка, Пром та ін.) зменшилися більше ніж на 40%, та по іншим видам неправомірних злочинів майже на 50%. До речі, саме шахрайські дії в мережі Інтернет становлять майже 70% від усіх звернень, які надходять до кіберполіції. Найрозповсюдженішими у 2021 році шахрайськими схемами були: продаж неіснуючих товарів, псевдовиграші, телефонні шахрайства, фішингові ресурси для привласнення грошей або збору персональних даних, заволодіння грошима під приводом надприбутків та прохання знайомих про допомогу в соціальних мережах.

Рисунок 3.22 Динаміка змін суми збитків

Завдяки впровадженню Antifraud-системи вже за 3 місяці в 2022 році це показало ефект, який можна побачити на рис.3.23. Фактично по певним групам

злочинів вдалося досягти показників 2020 року, хоча тенденція останніх років навпаки лише демонструвала ріст. Також додатково було розраховано прогнозовані значення на 2023 рік, з урахуванням того, що система в подальшому буде постійно самонавчатися та вдосконалюватися, бази даних будуть доповнюватися та розширюватися, кіберполіціянти будуть більше ознайомлені з функціоналом та будуть за потреби доповнювати його, проте не включає можливості виникнення все нових видів кіберзлочинів, що матимуть можливість протистояти системі розпізнавання.

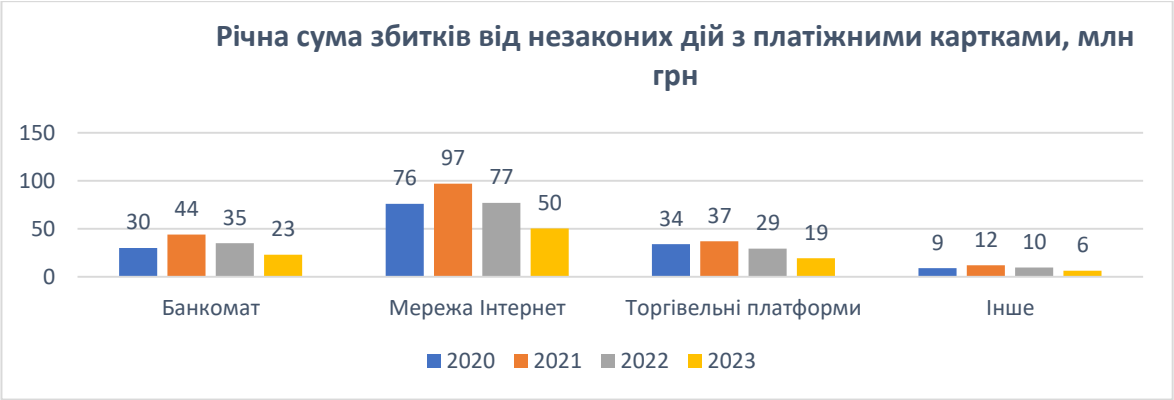

Рисунок 3.23 Динаміка змін річної суми збитків в розрізі групи злочинів

Загальна динаміка виглядає наступним чином: впровадження на 3 місяці готового функціоналу допомогло нам заощадити майже 40 млн.грн у 2022 році і становить 151 млн. грн, а прогнозоване значення на 2023 рік становить 99 млн грн, що на 35% менше ніж у 2022 році та на майже 50% менше ніж у 2020 році.

Рисунок 3.24 Динаміка змін річної суми збитків загальна

Також розробка та впровадження додатку з необхідним функціоналом по сприяли скороченню часу на розгляд справи: раніше в середньому на збір необхідної інформації по справі йшло 1 тиждень і 3 дні, наразі це займає 4 дні.

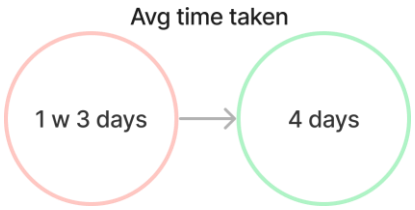

Рисунок 3.25 Середній час розгляду справи

3.5 Висновки до розділу

В даному розділі було проведено бізнес-аналіз додатку протидії кіберзлочинам в банківській сфері. В ході дослідження було виявлено, що бізнес-аналіз — це діяльність, яка уможливлює проведення змін у організації, які приносять користь зацікавленим сторонам, шляхом виявлення потреб та обґрунтування рішень, що описують можливі шляхи реалізації змін.

Завдяки методам бізнес-аналізу було зібрано необхідні вимоги до додатку шляхом проведення інтерв'ю користувачів та опитувань. Також в ході дослідження було з'ясовано, що вагому роль відіграє правильне визначення потенційних стейкхолдерів (зацікавлених осіб) продукту, адже це дозволило побудувати всебічну модель додатку, яка враховує усі вимоги, очікування та інтереси та протистояти обмеженням, які були визначені у Stakeholder map.

Також на етапі розробки додатку було побудовано інформаційну архітектури додатку, а також відтворення user flow, що допомогло зрозуміти чи дійсно відбулися покращення у шляху юзера по досягненні однієї і тієї ж цілі в порівнянні з шляхом до впровадження додатку.

Під час тестування функціоналу та було виявлено, що 6 з 6 опитуваних впоралися з поставленими задачами, середній час пошуку необхідного розділу з інформацією складає 3 хвилини 22 секунди, а 98% мали правильну відповідь,

що говорить про те, що навігація по додатку є зручною та практично, весь функціонал, що був необхідний клієнту було реалізовано.

Впровадження на 3 місяці готового функціоналу допомогло нам заощадити майже 40 млн.грн у 2022 році і становить 151 млн. грн, а прогнозоване значення на 2023 рік становить 99 млн грн, що на 35% менше ніж у 2022 році та на майже 50% менше ніж у 2020 році. Також впровадження додатку по сприяли скороченню часу на розгляд справи: раніше в середньому на збір необхідної інформації по справі йшло 1 тиждень і 3 дні, наразі це займає 4 дні.

ВИСНОВКИ

Кіберпростір створює неймовірні можливості, розширює свободу, стимулює розвиток інновацій та збагачує суспільство. Однак, паралельно з позитивними тенденціями, набуває розвитку і кіберзлочинність, що, зрозуміло, завдає значної шкоди інтересам наших громадян та держави в цілому. З метою протидії такій кримінальній протиправності в Національній поліції функціонує підрозділ кіберполіції.

Кваліфікаційна робота магістра присвячена дослідженню та оптимізації діяльності управління протидії кіберзлочинам в Дніпропетровській області, яка в свою чергу входить до складу апарату центрального органу управління поліції, а також впровадженню новітніх технологій для щоденного використання в розслідуванні злочинів та запобіганні шахрайських дій в банківському секторі.

Метою роботи установи є захист населення від кіберзлочинів, запобігання неправомірних дій та розслідування і викриття вже скоєних злочинів. Основні задачі даного відділку досить широкий: поліціанти

займаються боротьбою з вірусами, DdoS-атаками, спамом, шахрайством з банківськими системами і крадіжкою особистих даних.

Об'єктом дослідження є діяльність управління протидії кіберзлочинам в Дніпропетровській області ДКП НПУ, а саме виявлення неправомірних дій в банківській сфері.

Предметом дослідження є використання методів бізнес-аналізу для розробки додатка ідентифікації кіберзлочинів у банківській сфері управлінням протидії кіберзлочинам в Дніпропетровській області ДКП НПУ.

Метою роботи є підвищення ефективності роботи кіберполіціантів при розпізнаванні шахрайських дій та подальшого пошуку злочинців, можливого прогнозування неправомірних дій в банківських системах за допомогою розробленої Antifraud detection-системи.

Для досягнення поставленої мети в роботі визначені та вирішені наступні *задачі дослідження*:

– дослідити методи, за допомогою яких можна побудувати класифікатор банківських транзакцій;

– розробка та побудова класифікатора на незбалансованих даних;

– аналіз отриманих результатів та вибір оптимального методу;

– проведення бізнес-аналізу додатка для ідентифікації кіберзлочинів у банківській сфері, а саме збір та обробка вимог до додатка, розробка логічної схеми БД, моделювання елементів алгоритму роботи, а також результати застосування та оцінка ефективності додатка;

Для розв’язання поставлених задач в роботі запропоновано використовувати наступні *методи дослідження*:

– методи Undersampling та Oversampling для збалансування незбалансованої вибірки;

– методи XGBoost, AdaBoost та градієнтного бустингу та ансамблі відповідних моделей для побудови класифікатора банківських транзакцій з метою виявлення шахрайської діяльності

– різноманітні техніки виконання бізнес-аналізу додатка, тощо.

В спеціальному розділі представлена програмна реалізація машинного навчання, а саме покрокова реалізація алгоритмів XGBoost, AdaBoost, градієнтного бустингу та ансамблів відповідних. В якості вхідних даних було взято вибірку з відкритого джерела, так як доступ до персональних даних, скачування і подальше використання і вивчення для осіб, що не мають спеціального дозволу, заборонено чинним законодавством. Набір даних містить транзакції, здійснені кредитними картками у вересні 2013 року українськими власниками карток, що відбулися за два дні. В ході аналізу вхідних даних було виявлено, що між даними існує великий дисбаланс, адже кількість шахрайських транзакцій сягала менше 1% від усієї вибірки.

Було проведено 4 експериментальних навчань за різних умов, задля виявлення найоптимальніших значень. В результаті було отримано наступні результати:

Експеримент №1. Побудова класифікатора на незбалансованій вибірці :

– Метод градієнтного бустингу: TN = 94 778, FP = 36, FN = 45, TP = 77;

– Метод XGBoost: TN = 94 810, FP = 4, FN = 37, TP = 85;

– Метод AdaBoost: TN = 94 802, FP = 12, FN = 38, TP = 84.

Експеримент №2. Побудова класифікатора на вибірці, поділеній на 5 рівних частин, з метою зменшення дисбалансу в вибірці :

– Метод XGBoost: TN = 94 756, FP = 58, FN = 28, TP = 94, а оцінки точності моделі складають : recall = 0.77, precision = 0.62, accuracy = 1.00;

– Метод AdaBoost: TN = 94 380, FP = 434, FN = 27, TP = 95 а оцінки точності моделі складають : recall = 0.81, precision = 0.01, accuracy = 0.92.

Експеримент №3. Побудова класифікатора на збалансованій вибірці методом Undersampling (NNK):

– Метод XGBoost: TN = 93 640, FP = 1174, FN = 24, TP = 98, а оцінки точності моделі складають : recall = 0.80, precision = 0.08, accuracy = 0.99;

– Метод AdaBoost: TN = 80 423, FP = 14 391, FN = 15, TP = 107 а оцінки точності моделі складають : recall = 0.88, precision = 0.01, accuracy = 0.85.

Експеримент №4. Побудова класифікатора на збалансованій вибірці методом Oversampling (Random):

– Метод XGBoost: TN = 94 656, FP = 158, FN = 23, TP = 99, а оцінки точності моделі складають : recall = 0.81, precision = 0.39, accuracy = 1.00;

– Метод AdaBoost: TN = 92 825, FP = 1 989, FN = 23, TP = 99 а оцінки точності моделі складають : recall = 0.81, precision = 0.05, accuracy = 0.98.

Аналізуючи результати машинного навчання варто відзначити, що метод градієнтного бустингу виявися не оптимальним, адже давав низькі за точністю результати і час виконання навчання був дуже довгим в порівнянні з іншими двома методами, тому його було виключено з подальших розрахунків. Також в ході реалізації двох методів було застосовано методи семплінгу, що посприяли покращенню результатів машинного навчання. Спираючись на результати експериментів можна вважати експеримент № 3 і №4 результативними, проте вибір конкретного методу бустингу і семплінгу може варіюватися.

– Алгоритм XGBoost має більш вищий показник точності класифікатора (більш точно відокремлює «хороші» транзакції), а також має більш швидкий час виконання алгоритму навчання, а у алгоритму AdaBoost кращий показник глибини класифікатора (більш точно відокремлює «погані» транзакції), зумовлено тим, що алгоритм AdaBoost навчається на «поганих»

результатах попереднього навчання. Тобто з кожним кроком все зменшується кількість поганих результатів навчання. Проте за рахунок цього достатньо велика кількість, можливо хороших транзакцій потрапляють до «підозрілих».

В такому випадку можна відштовхуватися від побажань замовника подібних класифікаторів або ж доречно використовувати ці два алгоритми разом. Вони, так би мовити, будуть збалансовувати результати навчання одне одного.

В нашому випадку замовником є державна правоохоронна установа, метою якої є виявити якомога більше підозрілих транзакцій, так як вони в подальшому можуть складати більшу загрозу. Тому можна вважати, що в даному випадку оцінка глибини є більш опорною, ніж точність моделі.

В практичному розділі було проведено бізнес-аналіз додатку протидії кіберзлочинам в банківській сфері. Тобто це можна вважати вже цілісним проектом по впровадженню Antifraud detection-системи, адже розобки самого функціоналу недостатньо. Для ефективного його використання необхідно розробити зручний функціонал, який буде відповідати усім вимогам користувача, та буде містити увесь необхідний функціонал.

Особливості побудованої Antifraud detection-системи:

– Обробка подій у реальному часі.

– Система відповідає найжорсткішим вимогам до швидкості – тисячі подій обробляються в режимі real-time.

– Максимум виявленого шахрайства при мінімумі хибних спрацьовувань.

– Одночасне застосування експертних правил та самонавчальних моделей.

– Відкрита технологія керування для коригування політик та моделей.

– Рішення побудоване на відкритій технології управління.

– Зручний інтерфейс для швидкого прийняття рішень.

– Можливість створення та наповнення даними різних бізнес-об'єктів.

Усе це стало можливим завдяки методам бізнес-аналізу, що дозволили виконати збір необхідних вимог до додатку шляхом проведення інтерв’ю користувачів та опитувань. Також вагому роль відіграло визначення потенційних стейкхолдерів (зацікавлених осіб) продукту, адже це дозволило побудувати всебічну модель додатку, яка враховує усі вимоги, очікування та інтереси та протистояти обмеженням, які були визначені у Stakeholder map.

Також важливим кроком на етапі розробки додатку було побудова інформаційної архітектури додатку, а також відтворення user flow, що допомогло зрозуміти чи відбулися покращення у шляху юзера по досягненні однієї і тієї ж цілі в порівнянні з шляхом до впровадження додатку.

Практична цінність отриманих результатів полягає у отриманні готового додатку класифікатора банківських транзакцій, який дозволятиме визначати до якого класу відноситься та чи інша транзакція, тобто чи є вона "хороша", або "погана", блокувати їх, а також з використанням внутрішніх та зовнішніх банків даних та зручного функціоналу відшукувати злочинців, швидко отримувати необхідні дані про них та попереджати подальші неправомірні злочини. Тестування додатку вже за 3 місяці показало, що раніше в середньому на збір необхідної інформації по справі йшло 1 тиждень і 3 дні, наразі це займає 4 дні, це з урахуванням того факту, що зазвичай шахраї не обходяться 1 злочином, а мають велику кількість жертв, і тому саме збір доказів по всім випадкам подовжує час обробки та збору інформації по справі.

Економічний ефект від отриманих результатів очікується позитивним завдяки тому, що зменшується щорічний збиток від фінансового шахрайства в сфері онлайн платежів: впровадження на 3 місяці готового функціоналу допомогло нам заощадити майще 40 млн.грн у 2022 році і становить 151 млн.

грн, а прогнозоване значення на 2023 рік становить 99 млн грн, що на 35% менше ніж у 2022 році та на майже 50% менше ніж у 2020 році.

Соціальний ефект від результатів роботи очікується позитивним через скорочення кількості злочинів в фінансовій сфері, захист населення від неправомірних транзакцій, тобто призводить до підвищення довіри населення до органів Національної поліції України. Складність впровадження такої системи полягає у тому, що необхідно залучати команду спеціалістів, що мають реалізувати впровадження і забезпечити подальше супроводження продукту, проводити аналізування результатів класифікатора, задля виявлення поламок функціоналу і швидкого їх усунення, адже така система потребує знань та навичок в багатох сферах інформаційних технологій, якими не може володіти кіберполіціянт. З іншого ж підтримка з боку держави, яке є одним зі стейкхолдерів та залучення інших меценатів мають допомогти в зборі необхідних коштів та персоналу, адже це значно покращуватиме положення захисту населення в банківській сфері і підвищуватиме довіру до НПУ та влади.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке антифрод і де це використовується. [Електронний ресурс] - Режим доступу: https://www.bigdataschool.ru/blog/antifraud-cases.html
2. Побудова класифікаторів на незбалансованих вибірках на прикладі кредитного скорингу. [Електронний ресурс] - Режим доступу:https://u.to/ow1bGw
3. Алгоритми семплінга. [Електронний ресурс] - Режим доступу: https://u.to/tA1bGw
4. Ансамблеві методи: беггінг, бустінг і стекінг. [Електронний ресурс] - Режим доступу: https://u.to/vA1bGw
5. Бустинг. [Електронний ресурс] - Режим доступу: https://u.to/1Q1bGw
6. Дерево рішень. [Електронний ресурс] - Режим доступу: https://wiki.loginom.ru/articles/decision-trees.html
7. Дерева рішень: загальні принципи. [Електронний ресурс] - Режим доступу: https://loginom.ru/blog/decision-tree-p1
8. AdaBoost. [Електронний ресурс] - Режим доступу:https://u.to/4g1bGw
9. Градієнтний бустинг. [Електронний ресурс] - Режим доступу: https://en.wikipedia.org/wiki/Gradient_boosting
10. Бустинг за допомогою AdaBoost і Gradient Boosting. [Електронний ресурс] - Режим доступу: https://lambda-it.ru/post/busting-s-pomoshchiu-adaboost-i-gradient-boosting
11. XGBoost. [Електронний ресурс] - Режим доступу: https://neerc.ifmo.ru/wiki/index.php?title=XGBoost

12. XGBoost. [Електронний ресурс] - Режим доступу: https://u.to/6g1bGw

13. Credit Card Fraud Detection. [Електронний ресурс] - Режим доступу: https://www.kaggle.com/mlg-ulb/creditcardfraud

14. Оцінки класифікаторів. [Електронний ресурс] - Режим доступу: https://u.to/gdRbGw

15. Матриця помилок. [Електронний ресурс] - Режим доступу: https://u.to/fElcGw

16. Топ 10 бібліотек Python для машинного навчання. [Електронний ресурс] - Режим доступу: https://www.kverner.ru/top-10-bibliotek-python-dlya-mashinnogo-obucheniya/

17. Аналіз вимог. [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D1%96%D0%B7_%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3

18. Вимоги до програмного забезпечення. [Електронний ресурс] – Режим доступу: https://dut.edu.ua/ua/news-1-1009-1501-vimogi-do-programnogo-zabezpechennya_kafedra-inzhenerii-programnogo-zabezpechennya

19. Зацікавлені сторони. [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%86%D1%96%D0%BA%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D1%96_%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%B8

20. Карл Вігерс, Джой Бітті. Розробка вимог до програмного забезпечення: книга/, 2014. — 736 стр.

21. Діагностика та ідентифікація проблем. [Електронний ресурс] – Режим доступу: https://stud.com.ua/31874/menedzhment/diagnostika_identifikatsiya_problem

22. Що таке user story і як її писати. [Електронний ресурс] – Режим доступу: https://training.qatestlab.com/blog/technical-articles/user-story/

23. Клієнт-серверна архітектура. [Електронний ресурс] – Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/

24. Логічна модель даних. [Електронний ресурс] – Режим доступу: https://stud.com.ua/97320/informatika/logichna_model_danih_ponyattya_normalizatsiyi_vidnosin

25. Моделювання програмного забезпечення. [Електронний ресурс] – Режим доступу: http://elartu.tntu.edu.ua/bitstream/123456789/17796/1/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F.pdf

ДОДАТКИ

Додаток А

Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Назва	Кількість	Примітки
1				
2		Документація		
3				
4	САУ.КР.22.02.ПЗ	Пояснювальна записка	113	Формат А4
5				
6	САіУ.РД.21.01.ДМ	Демонстраційні матеріали		Презентація на CD-R
7				
8	САіУ.РД.21.01.КР	Копія роботи	1	Диск CD-R
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

					САУ.КР.22.02.ДА.ПЗ.
Змін.	Аркуш	№ докум.	Підпис	Дата	

Розроб.	Соловей			Матеріали дипломної роботи	Літ.	Аркуш	Аркушів
Керівн.	Желдак						
Керівн. Сп. Р.	Желдак				НТУ «ДП», 12; 124м-21		
Н. контр.	Хом’як						
Зав. Каф.	Желдак						

Додаток Б

ВІДГУК

на кваліфікаційну роботу ступеня «магістр»

за спеціальністю 124 - Системний аналіз

студентки групи 124м-21-1

Соловей Юлії Сергіївни

Кваліфікаційну роботу магістра присвячено темі ….

Керівник кваліфікаційної роботи магістра
Д. ф.-м. н., професор,
професор кафедри системного аналізу та управління _____/ Желдак Т.А.

РЕЦЕНЗІЯ

на кваліфікаційну роботу магістра за спеціальністю 124 – Системний аналіз
студентки групи 124м-21-1
Соловей Юлії Сергіївни

Тема кваліфікаційної роботи: «»

Рецензент

“ ” грудня 2022 р. _____

Додаток В

Програмна реалізація поставлених задача мовою програмування Python

```
B [1]:  import pandas as pd
        import numpy as np
B [2]:  data = pd.read_csv('creditcard.csv', header = 0, sep = ',')
B [3]:  len(data)
B [4]:  data.columns#columns names
B [5]:  data['Class'].value_counts() #number of non-fraudulant and fraudulant payments
B [6]: #imbalance proportion
        data['Class'].value_counts()[0]/data['Class'].value_counts()[1]
B [7]:  #are there any missing values?
          data.isna().sum()
B [8]:  data.tail()
B [9]:  data[['V1','V2','V3','V4','V5', 'V6','V7']].plot(figsize = (10,6))
B [10]: data[['V8','V9','V10','V11','V12', 'V13','V14']].plot(figsize = (10,6))
B [11]: data[['V15','V16','V17','V18','V19', 'V20','V21']].plot(figsize = (10,6))
B [12]: data[['V22','V23','V24','V25','V26', 'V27','V28']].plot(figsize = (10,6))
B [13]:  #time columns look strange
          data = data.drop(['Time'], axis = 1)
B [14]:  #separating train and test data
          data_for_train = pd.DataFrame(data.iloc[: int(2*len(data)/3)])
          data_for_test = pd.DataFrame(data.iloc[int(2*len(data)/3):])
B [15]: data_for_train['Class'].value_counts()
B [16]: data_for_test['Class'].value_counts()
B [17]:  #first model
          from sklearn.model_selection import train_test_split
          import xgboost as xgb
          from sklearn.metrics import confusion_matrix
          from imblearn.under_sampling import NearMiss
          from sklearn.model_selection import cross_val_score
          from sklearn.model_selection import RepeatedStratifiedKFold
          from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
          from imblearn.over_sampling import RandomOverSampler
          from imblearn.under_sampling import RandomUnderSampler
          from imblearn.under_sampling import OneSidedSelection, CondensedNearestNeighbour
          from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier
          feats1= ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
```

```
        'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
        'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28',]
    X_train = data_for_train[feats1]
    Y_train = pd.DataFrame(data_for_train['Class'])
    X_test = data_for_test[feats1]
    Y_test = pd.DataFrame(data_for_test['Class'])
    xgb_model = xgb.XGBClassifier(objective="binary:logistic", eval_metric = 'auc',
                          eta = 0.05, gamma = 0.1, reg_lambda = 1.5,
                            min_child_weight = 0.85, max_depth = 10,
                             tree_method = 'approx', n_estimators = 500,colsample_bytree = 0.5,
                            reg_alpha = 1.5,  random_state=11, n_jobs= 1, use_label_encoder=False, )
    grad_model = GradientBoostingClassifier(loss='deviance', learning_rate=0.1, n_estimators=600,
                                  subsample=1.0, criterion='friedman_mse', min_samples_split=2,
                                  min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=7,
                                  min_impurity_decrease=0.0, min_impurity_split=None, init=None,
                                  random_state=None, max_features=None, verbose=0,
                                  max_leaf_nodes=None, warm_start=False, validation_fraction=0.1,
                                  n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0)
    ada_model = AdaBoostClassifier(n_estimators=500, learning_rate = 0.7, random_state=0)
    xgb_model.fit(X_train, Y_train.values.ravel())
    pred_xgb = xgb_model.predict(X_test)
    from sklearn.metrics import confusion_matrix
    tn, fp, fn, tp = confusion_matrix(Y_test['Class'],pred_xgb).ravel()
    print((tn, fp, fn, tp))
B [18]: grad_model.fit(X_train, Y_train.values.ravel())
    pred_grad = grad_model.predict(X_test)
    tn, fp, fn, tp = confusion_matrix(Y_test['Class'],pred_grad).ravel()
    print((tn, fp, fn, tp))
B [19]: ada_model.fit(X_train, Y_train.values.ravel())
    pred_ada = ada_model.predict(X_test)

    tn, fp, fn, tp = confusion_matrix(Y_test['Class'],pred_ada).ravel()
    print((tn, fp, fn, tp))
B [20]: from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
    print(classification_report(Y_test['Class'],pred_xgb))
B [21]: #separating fraudulant and non-fraudulant data in the train data
    data_g = pd.DataFrame((data_for_train.loc[(data_for_train['Class'] == 0)])).reset_index(drop = True)
    data_s = pd.DataFrame(data_for_train.loc[(data_for_train['Class'] == 1)]).reset_index(drop = True)
```

B [22]:
```
#separating train data into 5 datasets containg all fraudulent data and 100 times more of non-fraudulent
#in order to reduce proportion
data_list = []
k = 0
for i in range(5):
    data_e = pd.concat([data_s, data_g[k:k+100*len(data_s)]],sort=True).reset_index(drop = True)
    k = k+100*len(data_s)
    data_list.append(data_e)
    print(len(data_list[i]))
```

B [23]:
```
model_list_xgb = []
model_list_ada = []
for i in range (5):
    data_s = pd.DataFrame(data_list[i].reset_index(drop = True))
    data1 = pd.DataFrame(data_s[feats1])
    data1['Class'] = data_s['Class']
    X_train = data1[feats1]
    Y_train = pd.DataFrame(data1['Class'])
     xgb_model = xgb.XGBClassifier(objective="binary:logistic", eval_metric = 'auc',
                                   eta = 0.05, gamma = 0.1, reg_lambda = 1.5,
                                     min_child_weight = 0.85, max_depth = 10,
                                      tree_method = 'approx', n_estimators = 500,colsample_bytree = 0.5,
                                     reg_alpha = 1.5,  random_state=11, n_jobs= 1, use_label_encoder=False, )
    ada_model = AdaBoostClassifier(n_estimators=500, learning_rate = 0.7, random_state=0)
    xgb_model.fit(X_train, Y_train.values.ravel())
    model_list_xgb.append(xgb_model)
    ada_model.fit(X_train, Y_train.values.ravel())
    model_list_ada.append(ada_model)
```

B [24]:
```
preds_x = []
prob_preds_x = []
for i in range(len(model_list_xgb)):
    pred = model_list_xgb[i].predict(X_test)
    prob_pred = model_list_xgb[i].predict_proba(X_test)
    preds_x.append(pred)
    prob_preds_x.append(prob_pred)
mean_prob = np.zeros(len(X_test))
for arr in prob_preds_x:
mean_prob = mean_prob+ arr[:,1]
mean_prob_x = mean_prob/len(prob_preds_x)
```

```
Y_test['prob_mean'] = mean_prob_x
Y_test['prob_mean_bin'] =  np.where(Y_test['prob_mean'] < 0.5, 0, 1)
preds_ax = np.array(preds_x)
Y_test['mean_pred' ] = preds_ax.mean(axis = 0)
Y_test['mean_bin'] =  np.where(Y_test['mean_pred'] < 0.5, 0, 1)
```

B [25]:
```
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
print((tn, fp, fn, tp))
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
print((tn, fp, fn, tp))
```

B [26]:
```
from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))
```

B [27]:
```
preds_a = []
prob_preds_a = []
for i in range(len(model_list_ada)):
    pred = model_list_ada[i].predict(X_test)
    prob_pred = model_list_ada[i].predict_proba(X_test)
    preds_a.append(pred)
    prob_preds_a.append(prob_pred)
mean_prob = np.zeros(len(X_test))
for arr in prob_preds_a:
   mean_prob = mean_prob+ arr[:,1]
mean_prob_a = mean_prob/len(prob_preds_x)
Y_test['prob_mean'] = mean_prob_a
Y_test['prob_mean_bin'] =  np.where(Y_test['prob_mean'] < 0.5, 0, 1)
preds_aa = np.array(preds_a)
Y_test['mean_pred' ] = preds_aa.mean(axis = 0)
Y_test['mean_bin'] =  np.where(Y_test['mean_pred'] < 0.5, 0, 1)
```

B [28]:
```
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
print((tn, fp, fn, tp))
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
print((tn, fp, fn, tp))
```

B [29]:
```
from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
print(classification_report(Y_test['Class'],Y_test['mean_bin']))
```

B [30]:
```
model_list_xgb = []
model_list_ada = []
for i in range (5):
    data_s = pd.DataFrame(data_list[i].reset_index(drop = True))
    data1 = pd.DataFrame(data_s[feats1])
```

```
        data1['Class'] = data_s['Class']
        X_train = data1[feats1]
        Y_train = pd.DataFrame(data1['Class'])
            xgb_model = xgb.XGBClassifier(objective="binary:logistic", eval_metric = 'auc',
                                  eta = 0.05, gamma = 0.1, reg_lambda = 1.5,
                                    min_child_weight = 0.85, max_depth = 10,
                                     tree_method = 'approx', n_estimators = 500,colsample_bytree = 0.5,
                                    reg_alpha = 1.5,  random_state=11, n_jobs= 1, use_label_encoder=False, )
          ada_model = AdaBoostClassifier(n_estimators=500, learning_rate = 0.7, random_state=0)
        undersample = NearMiss(version=3, n_neighbors=100)
        X_train, Y_train = undersample.fit_resample(X_train, Y_train)
        xgb_model.fit(X_train, Y_train.values.ravel())
        model_list_xgb.append(xgb_model)
        ada_model.fit(X_train, Y_train.values.ravel())
        model_list_ada.append(ada_model)
```

B [31]:
```
preds_x = []
prob_preds_x = []
for i in range(len(model_list_xgb)):
    pred = model_list_xgb[i].predict(X_test)
    prob_pred = model_list_xgb[i].predict_proba(X_test)
    preds_x.append(pred)
    prob_preds_x.append(prob_pred)

mean_prob = np.zeros(len(X_test))
for arr in prob_preds_x:
    mean_prob = mean_prob+ arr[:,1]
mean_prob_x = mean_prob/len(prob_preds_x)
Y_test['prob_mean'] = mean_prob_x
Y_test['prob_mean_bin'] =  np.where(Y_test['prob_mean'] < 0.5, 0, 1)
preds_ax = np.array(preds_x)
Y_test['mean_pred' ] = preds_ax.mean(axis = 0)
Y_test['mean_bin'] =  np.where(Y_test['mean_pred'] < 0.5, 0, 1)
```

B [32]:
```
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
print((tn, fp, fn, tp))
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
print((tn, fp, fn, tp))
```

B [33]:
```
from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
```

```
print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))
```

B [34]:
```
preds_a = []
prob_preds_a = []
for i in range(len(model_list_ada)):
    pred = model_list_ada[i].predict(X_test)
    prob_pred = model_list_ada[i].predict_proba(X_test)
    preds_a.append(pred)
    prob_preds_a.append(prob_pred)
mean_prob = np.zeros(len(X_test))
for arr in prob_preds_a:
    mean_prob = mean_prob+ arr[:,1]
mean_prob_a = mean_prob/len(prob_preds_x)
Y_test['prob_mean'] = mean_prob_a
Y_test['prob_mean_bin'] =  np.where(Y_test['prob_mean'] < 0.5, 0, 1)
preds_aa = np.array(preds_a)
Y_test['mean_pred' ] = preds_aa.mean(axis = 0)
Y_test['mean_bin'] =  np.where(Y_test['mean_pred'] < 0.5, 0, 1)
```

B [35]:
```
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
print((tn, fp, fn, tp))
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
print((tn, fp, fn, tp))
```

B [36]:
```
from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))
```

B [37]:
```
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import confusion_matrix
from imblearn.under_sampling import NearMiss
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.under_sampling import OneSidedSelection, CondensedNearestNeighbour
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier
model_list_xgb = []
model_list_ada = []
for i in range (5):
    data_s = pd.DataFrame(data_list[i].reset_index(drop = True))
```

```
    data1 =pd.DataFrame(data_s[feats1])
    data1['Class'] = data_s['Class']
    X_train = data1[feats1]
    Y_train = pd.DataFrame(data1['Class'])
    print(len(X_train),len(Y_train))
    xgb_model = xgb.XGBClassifier(objective="binary:logistic", eval_metric = 'auc',
                                  eta = 0.05, gamma = 0.1, reg_lambda = 1.5,
                                   min_child_weight = 0.85, max_depth = 6,
                                   tree_method = 'approx', n_estimators = 500,colsample_bytree = 0.5,
                                   reg_alpha = 1.5,  random_state=11, n_jobs= 1, use_label_encoder=False, )
    ada_model = AdaBoostClassifier(n_estimators=500, learning_rate = 0.7, random_state=0)
    over = RandomOverSampler(sampling_strategy=1)  # fit and apply the transform
    X_train, Y_train = over.fit_resample(X_train, Y_train)
    xgb_model.fit(X_train, Y_train.values.ravel())
    pred_train = xgb_model.predict(X_train)
    model_list_xgb.append(xgb_model)
    ada_model.fit(X_train, Y_train.values.ravel())



    model_list_ada.append(ada_model)
```

B [38]:

```
preds_x = []
prob_preds_x = []
for i in range(len(model_list_xgb)):
    pred = model_list_xgb[i].predict(X_test)
    prob_pred = model_list_xgb[i].predict_proba(X_test)
    preds_x.append(pred)
    prob_preds_x.append(prob_pred)
mean_prob = np.zeros(len(X_test))
for arr in prob_preds_x:
    mean_prob = mean_prob+ arr[:,1]
mean_prob_x = mean_prob/len(prob_preds_x)
Y_test['prob_mean'] = mean_prob_x
Y_test['prob_mean_bin'] =  np.where(Y_test['prob_mean'] < 0.5, 0, 1)
preds_ax = np.array(preds_x)
Y_test['mean_pred' ] = preds_ax.mean(axis = 0)
Y_test['mean_bin'] =  np.where(Y_test['mean_pred'] < 0.5, 0, 1)
```

B [39]:

```
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
print((tn, fp, fn, tp))
```

```
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
print((tn, fp, fn, tp))
```

B [40]:
```
rom sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))
```

B [41]:
```
preds_a = []
prob_preds_a = []
for i in range(len(model_list_ada)):
    pred = model_list_ada[i].predict(X_test)
    prob_pred = model_list_ada[i].predict_proba(X_test)
    preds_a.append(pred)
    prob_preds_a.append(prob_pred)
mean_prob = np.zeros(len(X_test))
for arr in prob_preds_a:
    mean_prob = mean_prob+ arr[:,1]
mean_prob_a = mean_prob/len(prob_preds_x)
Y_test['prob_mean'] = mean_prob_a
Y_test['prob_mean_bin'] =  np.where(Y_test['prob_mean'] < 0.5, 0, 1)
preds_aa = np.array(preds_a)
Y_test['mean_pred' ] = preds_aa.mean(axis = 0)
Y_test['mean_bin'] =  np.where(Y_test['mean_pred'] < 0.5, 0, 1)
```

B [42]:
```
from sklearn.metrics import accuracy_score, f1_score, classification_report, plot_roc_curve
print(classification_report(Y_test['Class'],Y_test['prob_mean_bin']))
```

B [43]:
```
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['prob_mean_bin']).ravel()
print((tn, fp, fn, tp))
tn, fp, fn, tp = confusion_matrix(Y_test['Class'],Y_test['mean_bin']).ravel()
print((tn, fp, fn, tp))
```