Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Факультет інформаційних технологій

(факультет)

Кафедра системного аналізу та управління

(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня бакалавра

Студента Бутенко Надії Віталіївни

академічної групи 124-19-2

спеціальності 124 Системний аналіз

на тему: «Аналіз методів класифікації незбалансованих даних на прикладі прогнозу дефолту бізнес займів»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**Інституційною**	
кваліфікаційної роботи	*д.ф-м.н., проф. Купенко О.П.*			
розділів:				
Інформаційно-аналітичний	*д.ф-м.н.,проф. Купенко О.П.*			
Спеціальний розділ	*д.ф-м.н.,проф. Купенко О.П.*			
Рецензент	*д.т.н.,проф. Гнатушенко В.В.*			
Нормоконтролер	*к.ф.-м.н.,доц. Хом'як Т.В.*			

Дніпро

2023

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

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

«____» _____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *бакалавра*

студенту *Бутенко Н.В.* **академічної групи** *124- 19-2*
спеціальності: *124 Системний аналіз*
на тему «*Аналіз методів класифікації незбалансованих даних на прикладі прогнозу дефолту бізнес займів»*
затверджену наказом ректора НТУ «Дніпровська політехніка»
від 16.05.2023 №350-с

Розділ	Зміст	Терміни виконання
1. Інформаційно-аналітичний розділ	*Проаналізувати структуру об'єкта дослідження. Визначити предметну область дослідження та проблему, що розв'язується. Обґрунтувати методи виконання поставлених завдань*	25.03.2023-02.04.2023
2.Спеціальний розділ	*Розв'язати поставлені задачі: проанілузвати методи балансування даних і користувацьких функцій втрат, використовуючи незбалансований датасет.*	05.04.2023-27.05.2023

Завдання видано _____ *проф. Купенко О.П.*
(підпис) (прізвище, ініціали)

Дата видачі: *20.03.2022 р.*
Дата подання до екзаменаційної комісії: _____
Прийнято до виконання _____ Бутенко Н.В.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, двох розділів, 12 підрозділів, загального висновку, списку використаних джерел, 9 додатків. У кваліфікаційній роботі розміщено 53 рисунка, 12 таблиць, використано 24 джерела. Кваліфікаційна робота має обсяг 159 сторінок.

Ключові слова: незбалансований датасет, модель, класифікація, способи балансування даних, балансування, користувацькі функції втрат.

Об'єктом дослідження в цій кваліфікаційній роботі є процес прогнозування дефолту бізнес-позик, з особливим акцентом на вплив незбалансованості класів у наборах даних на якість прогнозування.

Предметом дослідження є різні методи класифікації незбалансованих даних, включаючи їх теоретичні аспекти, практичне застосування на реальних даних та оцінювання їх ефективності.

Метою цієї кваліфікаційної роботи є дослідження та аналіз різних методів класифікації незбалансованих даних для покращення прогнозування дефолту бізнес позик, оцінювання ефективності цих методів на основі відповідних даних.

Результати кваліфікаційної роботи включають детальний аналіз та оцінку різних методів класифікації незбалансованих даних, включаючи їх ефективність в контексті прогнозування дефолту бізнес-позик. Інноваційність роботи полягає в підході до вибору та застосуванні методів, які найбільше підходять для обробки незбалансованих даних. метрик якості, що враховують особливості незбалансованості даних.

Використовується мова програмування Python із бібліотеками, такими як Pandas, NumPy, Matplotlib, Seaborn та Scikit-learn, для обробки даних, виконання аналізу та прогнозування.

Робота вписується в ширший контекст досліджень у галузі машинного навчання та аналізу даних, зокрема в дослідження, пов'язані з класифікацією незбалансованих даних.

Результати роботи рекомендується використовувати в банківській сфері для покращення процесу прогнозування дефолту бізнес-позик.

Ефективність роботи проявляється в зменшенні ризику кредитних втрат для банків та підвищенні їх прибутку.

Робота важлива, оскільки вона спрямована на розв'язання актуальної проблеми класифікації незбалансованих даних, що має велике значення в багатьох галузях, зокрема в банківській

ABSTRACT

The qualification work consists of an introduction, two chapters, 12 subsections, a general conclusion, a list of references, and 5 appendices. The essay includes 53 figures, 12 tables, and utilizes 24 sources. The qualification work has a volume of 159 pages.

Keywords: imbalanced dataset, model, classification, data balancing techniques, balancing, custom loss functions.

The object of research in this diploma work is the process of predicting default in business loans, with a particular focus on the impact of class imbalance in the datasets on the quality of prediction.

The subject of research includes various methods of classification for imbalanced data, including their theoretical aspects, practical application on real data, and evaluation of their effectiveness.

The aim of this diploma work is to investigate and analyze different methods of classifying imbalanced data to improve the prediction of default in business loans and evaluate the effectiveness of these methods based on relevant data.

The results of the diploma work include a detailed analysis and evaluation of different methods of classifying imbalanced data, including their effectiveness in the context of predicting default in business loans. The innovation of the work lies in the approach to selecting and applying methods that are most suitable for processing imbalanced data, as well as quality metrics that consider the characteristics of data imbalance.

The programming language Python is used with libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn for data processing, analysis, and prediction.

The work fits into the broader context of research in the field of machine learning and data analysis, particularly in research related to the classification of imbalanced data.

The results of the work are recommended for use in the banking sector to improve the process of predicting default in business loans.

The effectiveness of the work is manifested in reducing credit losses for banks and increasing their profits.

The work is important as it aims to address the current problem of classifying imbalanced data, which is of great significance in many fields, particularly in banking.

ЗМІСТ

ВСТУП

Банківська сфера присутня у житті кожної людини і відповідно кожна людина хоча б раз у житті користувалася послугами цієї сфери. Виходячи з цього висновку можна зрозуміти, що перед сучасними банками постають наступні питання: «Яким чином можна надати послуги якомога більшій кількості людей?», «Як з цієї великої кількості клієнтів віднайти тих, хто з великою вірогідністю не поверне борг?». Відповідь на це питання – автоматизація.

Автоматизація або прогнозування повернення чи не повернення боргу клієнтом є звичайною задачею бінарної класифікації. Однак, у реальних банківських даних часто виникає проблема незбалансованості класів, коли кількість невідшкодованих кредитів значно менша за кількість відшкодованих. Це може призвести до неправильного прогнозування дефолту, що у свою чергу може привести до збитків для банку або невиправданого відмовлення в кредитуванні з боку засновника. Тому вирішення проблеми незбалансованості класів в задачах класифікації є важливим напрямком дослідження.

Метою даної роботи є дослідження різних методів класифікації незбалансованих даних на прикладі прогнозування дефолту бізнес-позик. У роботі будуть проаналізовані теоретичні аспекти класифікації даних, незбалансовані дані та їх вплив на класифікацію. Також будуть розглянуті різні методи класифікації незбалансованих даних, їх переваги та недоліки. У дослідженні будуть використані реальні дані про бізнес позики банку, які будуть підготовлені та очищені перед дослідженням. Результати дослідження будуть проаналізовані та порівняні

Отже, дана робота має важливе значення для практичного застосування в банківській сфері та допоможе покращити якість прогнозування дефолту та уникнути можливих негативних наслідків.

1 ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

1.1 Дослідження бізнес позик та особливості набору даних

Дані про бізнес-позики є одними з найбільш цікавих і важливих наборів даних в галузі аналізу кредитного ризику. Вони містять інформацію про кредитні заявки, подані бізнесами, та результати рішень щодо цих заявок. Такий аналіз може бути корисним для оцінки кредитного ризику та прийняття рішень у банківському секторі. Однією з особливостей цього набору даних є те, що він містить велику кількість категоріальних ознак (наприклад, стан роботи, мета кредиту, статус подружжя тощо).

Це може ускладнити роботу з даними, оскільки більшість алгоритмів машинного навчання потребують числових ознак. Тому, щоб використовувати цей набір даних, можна застосовувати методи кодування категоріальних ознак, такі як Label Encoding або One-Hot Encoding.

Іншою особливістю цього набору даних є несбалансованість класів. Це означає, що кількість зразків одного класу (наприклад, відмови в кредиті) значно менша, ніж кількість зразків іншого класу (наприклад, успішні кредити). Це може призвести до проблем з точністю класифікації, оскільки алгоритми будуть навчатись на домінуючому класі, і результати для менш домінуючого класу можуть бути менш точними.

Для ефективного аналізу та розуміння набору даних для дослідження бізнес-кредитів можуть використовуватися різноманітні методи EDA (exploratory data analysis), такі як візуалізація даних та статистичний аналіз, а також методи машинного навчання для побудови моделей класифікації та прогнозування.

Для роботи з таким набором даних можна використовувати різні методи боротьби з несбалансованістю класів, такі як зменшення кількості зразків

більшого класу (undersampling), збільшення кількості зразків меншого класу (oversampling), або комбінації цих методів.

1.2 Етапи аналізу даних

У цьому розділі буде и розглянуто трійчатий підхід до аналізу даних: "Initial Data Analysis" (IDA), "Exploratory Data Analysis" (EDA) і "Confirmatory Data Analysis" (CDA). Ці етапи роботи з даними спрямовані на забезпечення якісного дослідження, починаючи від первинної обробки та перевірки даних, до їх детального аналізу з використанням статистичних методів, та наостанок, використання розроблених моделей для отримання відповідей на поставлені дослідницькі питання. Кожен етап відіграє важливу роль у створенні об'єктивної та точної картини, яка дозволяє прийняти обґрунтовані рішення.

1.2.1 IDA (Initial Data Analysis)

Initial Data Analysis (IDA), або Початковий аналіз даних, — це перший етап роботи з даними в проекті з аналізу даних або машинного навчання. IDA зазвичай включає в себе наступні кроки:

1. Завантаження даних – дані завантажуються з файлів, баз даних, API, зберігань у хмарі або інших джерел.
2. Огляд даних – здійснюється перша перевірка даних на об’є їхнього формату, об'єму, кількості змінних та типів даних.
3. Попередня обробка даних – це може включати видалення дублікатів, розбиття стовпців на окремі змінні, перетворення типів даних та інше.
4. Перевірка якості даних – на цьому етапі досліджуються пропуски в даних, аномальні значення та інші можливі проблеми.
5. Описова статистика – залежно від проекту, може бути корисним провести описову статистику, що включає розрахунок середніх значень, медіани,

стандартного відхилення, кількості унікальних значень для категоріальних змінних тощо.

Ці кроки допомагають досліднику зрозуміти структуру та якість даних, визначити, які додаткові кроки попередньої обробки або очищення даних можуть бути необхідними, а також формулювати гіпотези для подальшого аналізу.

1.2.2 EDA (Exploratory Data Analysis)

Науковці даних можуть використовувати дослідницький аналіз даних, щоб переконатися, що отримані результати є дійсними та застосовними до будь-яких бажаних бізнес-результатів і цілей. EDA також допомагає зацікавленим сторонам, підтверджуючи, що вони ставлять правильні запитання. EDA може допомогти відповісти на запитання про стандартні відхилення, категоричні змінні та довірчі інтервали. Після того, як EDA буде завершено та отримано розуміння, його функції можна буде використовувати для більш складного аналізу даних або моделювання, включаючи машинне навчання.

Етапи EDA (Exploratory Data Analysis) включають ряд послідовних кроків, які допомагають зрозуміти, аналізувати та візуалізувати дані перед застосуванням будь-яких методів моделювання чи передбачення. Ось кілька ключових етапів EDA:

1. Збір даних – цей етап включає збір даних з різних джерел, таких як бази даних, файли, API або веб-сторінки. Метою цього кроку є забезпечення якісних і достовірних даних для аналізу.
2. Очищення та підготовка даних – на цьому етапі аналітики перевіряють дані на наявність пропусків, аномалій, дублікатів та неточностей. Це включає виправлення помилок, видалення або заміну пропущених значень та перетворення даних у відповідний формат.
3. Дослідження структури даних – на цьому етапі аналітики вивчають розподіл змінних, кількість унікальних значень та залежності між змінними.

Це включає використання статистичних методів, таких як кореляція, коваріація та аналіз розподілу.

4. Візуалізація даних – застосування графічних методів для відображення даних, що допомагає краще зрозуміти тенденції, шаблони та взаємозв'язки між змінними. Це може включати створення графіків, таких як гістограми, діаграми розсіювання, ящики з вусами (box plots) та теплові карти.

Оцінка та вибір функцій: На цьому етапі відбувається відбір найважливіших змінних або функцій для побудови моделі.

Також слід пам’ятати, що збір даних може відбуватися на етапах як IDA, так і EDA, але це залежить від контексту проекту.

У ситуації, коли дослідник вперше зустрічається з даними, IDA є початковим етапом аналізу, де відбувається збір даних. На цьому етапі можуть проводитися перші спостереження і початкові обчислення, але без глибокого розуміння структури даних або можливих взаємозв'язків між ними.

Після IDA, коли дослідник уже має деяке уявлення про дані, починається етап EDA. На цьому етапі, якщо необхідно, можуть збиратися додаткові дані для подальшого дослідження, або може відбуватися більш глибокий аналіз вже наявних даних.

Таким чином, збір даних може відбуватися як на етапі IDA, так і EDA, але залежить від потреб дослідження.

1.2.3 CDA (Confirmatory Data Analysis)

Confirmatory Data Analysis (CDA) є наступним етапом після проведення Exploratory Data Analysis (EDA). Під час CDA, ви здійснюєте більш глибокий аналіз з метою підтвердження ваших гіпотез, які були сформовані під час EDA.

Отже, основні етапи CDA можуть включати наступне:

1. Формування гіпотези – на основі результатів EDA, ви формуєте гіпотези, які ви хочете перевірити.
2. Вибір відповідних статистичних тестів – вибір тесту залежить від типу даних і природи гіпотези. Наприклад, ви можливо використовуватимете t-тест для порівняння середніх двох груп, або хі-квадрат тест для аналізу асоціативних відносин між категоріальними змінними.
3. Проведення тесту і інтерпретація результатів – застосовуючи обраний статистичний тест, ви отримуєте результати, які потім інтерпретуєте. Чи підтверджується ваша гіпотеза? Чи є результати статистично значущими?
4. Відображення результатів – результати CDA часто представляються за допомогою візуалізацій, таких як графіки або діаграми, для легшого розуміння. Для цього в Python можна використовувати бібліотеки matplotlib або seaborn.
5. Написання висновків – після аналізу і інтерпретації результатів, ви формуєте висновки. Чи підтверджується ваша гіпотеза? Чи є результати важливими для вашого дослідження або бізнесу?

Важливо пам'ятати, що CDA є частиною циклу аналізу даних і часто йде в поєднанні з EDA і IDA для комплексного аналізу даних.

1.3 Методи підвищення якості моделей класифікації

Підвищення продуктивності моделі машинного навчання іноді може бути складним завданням, так як багато аспектів можуть значною мірою впливати на подальше прогнозування.

Розглянемо нижче деякі методи для підвищення ефективності моделей класифікації.

1. Очищення і підготовка даних:

   Видалення пропусків або їх заповнення за принципом категоріальні заповнюються модою, а чисельні – середнім значенням.

Видалення або заміна вибросів у даних. виявлення та видалення вибросів можна здійснити за допомогою статистичних методів (наприклад, IQR) або моделей машинного навчання (наприклад, Isolation Forest).

Кодування категоріальних даних – категоріальні дані кодуються у числову форму за допомогою вже відомих методів one-hot encoding, label encoding або інших;

Масштабування або нормалізація числових змінних – допомагає забезпечити, що різні числові змінні мають порівнянні значення. Можна застосувати MinMaxScaler, StandardScaler або інші методи.

2. Відбір ознак

   Використання статистичних методів (наприклад, кореляція або хі-квадрат).

   Використання методів, заснованих на моделях (наприклад, Lasso, Ridge або важливість ознак в деревах рішень).

   Рекурсивне видалення ознак або використання методів вбудованого відбору ознак (наприклад, використання RandomForest), видалення ознак за VIF (Variance Inflation Factor). VIF. використовується для виявлення мультиколінеарності між незалежними змінними в лінійних регресійних моделях. VIF розраховується для кожної незалежної змінної, і якщо значення VIF вище за певний поріг (наприклад, 5 або 10), це може свідчити про наявність мультиколінеарності. У такому випадку, можна видалити одну з мультиколінеарних змінних або застосувати методи регуляризації, щоб зменшити вплив мультиколінеарності на модель класифікації.

3. Cross-validation (перехресна перевірка). допомагає оцінити ефективність моделі на різних наборах даних. Це дозволяє уникнути перенавчання та вибрати оптимальні параметри моделі. Зазвичай використовують k-кратну крос-валідацію, де набір даних розбивається на k частин, а модель навчається та перевіряється кілька разів, кожен раз на різних наборах даних.

4. Оптимізація гіперпараметрів моделі

Grid Search (Пошук по сітці) – систематично перебирає всі можливі комбінації значень гіперпараметрів, щоб знайти ті, які дають кращі результати. Може бути повільним, але гарантує знаходження оптимального набору значень.

Random Search (Випадковий пошук) – випадковим чином перебирає комбінації значень гіперпараметрів, що дозволяє знайти непогані значення за менший час порівняно з пошуком по сітці.

Байєсівська оптимізація – використовує байєсівські методи для оцінки ймовірності оптимальних значень гіперпараметрів, швидше збігається до оптимального набору значень, ніж пошук по сітці та випадковий пошук.

5. Ансамблі моделей

Bagging (Беггінг) – створює кілька моделей з випадковими підмножинами навчальних даних, потім об'єднує їхні прогнози для досягнення кращого результату. Випадковий ліс (Random Forest). один з алгоритмів беггінга.

Boosting (Бустінг) – поєднує кілька слабких моделей, навчаючи кожну з них на остатках попередньої моделі, щоб створити сильну модель. Градієнтний бустінг (Gradient Boosting) та XGBoost. приклади алгоритмів бустінга.

Stacking (Стекінг) – навчає кілька різних моделей на навчальному наборі даних, а потім використовує ще одну модель (наприклад, лінійну регресію), яка навчається на прогнозах цих моделей, щоб зробити кінцевий прогноз. Це дозволяє комбінувати сильні сторони різних алгоритмів для досягнення кращої точності.

6. Балансування даних – даний метод буде розглянуто детальніше нижче.

1.4 Поняття незбалансованості даних та проблема класифікації

Незбалансовані дані є частим явищем в задачах класифікації, коли кількість зразків у кожному класі суттєво відрізняється. Наприклад, якщо у нас є задача розпізнавання шкідливого вмісту на сайті, то кількість позитивних зразків (тобто зразків зі шкідливим вмістом) може бути набагато меншою, ніж кількість негативних зразків (тобто зразків з нешкідливим вмістом).

Нерівномірний розподіл класів може призвести до низької точності класифікації, особливо якщо міноритарний клас (рідкісні випадки) є об’єктом інтересу. У такому випадку, модель може бути схильна до прогнозування більшої кількості екземплярів до домінуючого класу, тим самим зменшуючи точність класифікації міноритарного класу.

Існує декілька підходів для роботи з незбалансованими даними. Один з них полягає в зборі додаткових даних для міноритарного класу. Це може бути складним завданням в деяких сценаріях, оскільки деякі класи можуть бути дуже рідкісними або взагалі недоступними для збору.

Інші підходи зазвичай зосереджені на зміні алгоритмів машинного навчання, щоб вони були більш ефективними для незбалансованих даних. Наприклад, можна використовувати ваги класів, які дають більшу вагу міноритарному класу при навчанні моделі, або використовувати метрики, які враховують незбалансованість класів, такі як precision-recall або ROC AUC.

Також можна використовувати техніки підвищення класів (class boosting), такі як SMOTE [3] (Synthetic Minority Over-sampling Technique), які створюють штучні екземпляри міноритарного класу, щоб збалансувати розподіл класів. Ці техніки не завжди працюють добре в практичних застосуваннях, але вони можуть бути корисними в деяких сценаріях.

Або використання користувацьких функцій витрат для прогнозування, такі як: focal loss, weighted loss та інші.

Важливо враховувати, що вибір методу залежить від конкретних властивостей набору даних і потребує додаткового аналізу.

1.5 Аналіз методів балансування даних

Основні методи балансування даних включають в себе oversampling та undersampling. Oversampling полягає в дублюванні менших класів, тоді як undersampling випадково видаляє зайві зразки більших класів. Нижче можна подивитися на графічну інтерпретацію роботи цих двох методів (Рис. 1.1).

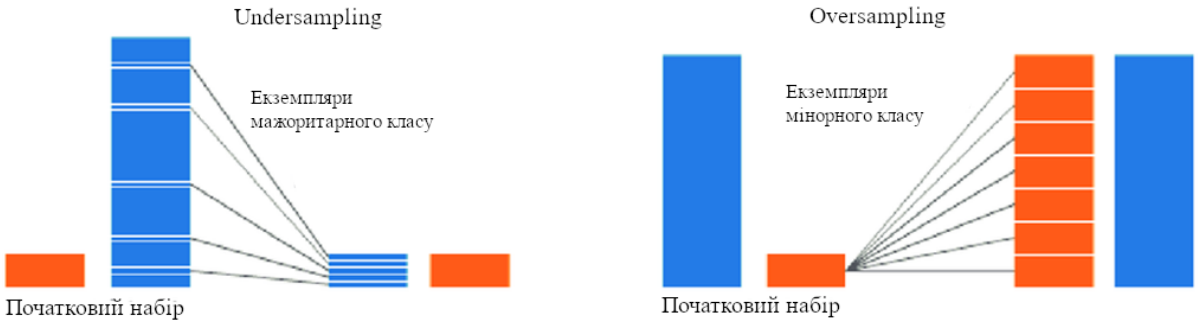


Рис. 1.1. Графічна інтерпретація роботи Oversampling і Undersampling

Однак, ці методи можуть привести до перенавчання моделі або недостатнього навчання моделі на мало представлених класах. Тому можуть використовуватись інші методи, такі як Synthetic Minority Over-sampling Technique (SMOTE) – цей алгоритм описали Nitesh Chawla та ін. у своїй статті 2002 року [17].

SMOTE працює, обираючи приклади, які є близькими у просторі ознак, проводить лінію між прикладами у просторі ознак і створює новий зразок в точці на цій лінії.

Конкретно, спочатку вибирається випадковий приклад з меншої категорії. Потім знаходяться k найближчих сусідів для цього прикладу (зазвичай $k = 5$). Вибирається випадково обраний сусід, і синтетичний приклад створюється в випадково вибраній точці між двома прикладами у просторі ознак. Знаходиться різниця між векторами ознак сусідніх елементів a і b цього класу, що знайдено методом k-найближчих сусідів.

Ця процедура може використовуватися для створення стільки синтетичних прикладів для меншої категорії, скільки потрібно. Як описано в статті, спочатку пропонується використовувати випадкове недооцінювання для зменшення кількості прикладів у більшій категорії, а потім застосовувати SMOTE для збільшення меншої категорії з метою збалансування розподілу класів.

Підхід ефективний, оскільки створюються нові синтетичні приклади з меншої категорії, які є правдоподібними, тобто знаходяться відносно близько у просторі ознак до існуючих прикладів з меншої категорії.

Детальніше алгоритм описаний на рисунку нижче (Рис. 1.2) [17].

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE N%; Number of nearest neighbors k
Output: $(N/100) * T$ synthetic minority class samples
1. (∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
8. k = Number of nearest neighbors
9. *numattrs* = Number of attributes
10. *Sample*[ ][ ]: array for original minority class samples
11. *newindex*: keeps a count of number of synthetic samples generated, initialized to 0
12. *Synthetic*[ ][ ]: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
13. **for** $i \leftarrow 1$ **to** T
14. Compute k nearest neighbors for i, and save the indices in the *nnarray*
15. Populate(N, i, *nnarray*)
16. **endfor**

Populate(N, i, *nnarray*) (∗ *Function to generate the synthetic samples.* ∗)
17. **while** $N \neq 0$
18. Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
19. **for** $attr \leftarrow 1$ **to** *numattrs*
20. Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
21. Compute: gap = random number between 0 and 1
22. $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
23. **endfor**
24. $newindex{+}{+}$
25. $N = N - 1$
26. **endwhile**
27. **return** (∗ *End of Populate.* ∗)
End of Pseudo-Code.

Рис. 1.2. Псевдокод алгоритму SMOTE

На рисунку (Рис. 1.3) нижче продемонстрована графічна інтерпретація алгоритму SMOTE.

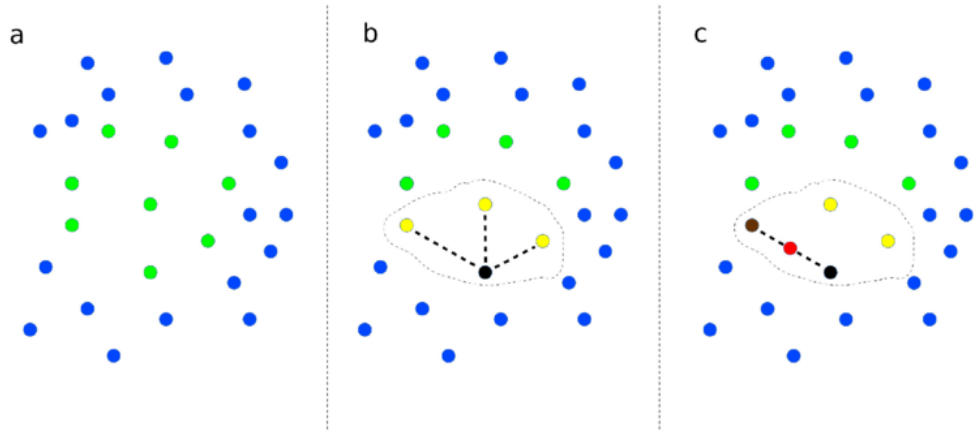


Рис. 1.3. Графічна інтерпретація роботи SMOTE

Також слід не забувати про недоліки даного методу балансування даних, а саме, що цей алгоритм може створювати синтетичні приклади, які є шумовими або відображають області перекриття між класами. А також він є достатньо часозатратним для великих наборів даних через генерацію синтетичних зразків.

Метод SMOTE також є підґрунтям для інших методів балансування даних таких як: SMOTETOMEK, SMOTEEN, ADASYN .

SMOTETOMEK. це комбінований підхід до балансування наборів даних, який використовує два методи: SMOTE (Synthetic Minority Over-sampling Technique) та Tomek Links. Метод SMOTE було розглянуто вище, а метод посилань Томека (Tomek Links). це метод видалення прикладів, який зосереджується на видаленні екземплярів, що створюють шум у наборі даних.

Тобто цей метод можна описати наступним чином: задано два приклади E_i та E_j, які належать до різних класів, і $d(E_i, E_j)$. це відстань між E_i та E_j Пара (E_i, E_j) називається зв'язком Томека, якщо не існує такого прикладу E_l, для якого $d(E_i, E_l) < d(E_i, E_j)$ або $d(E_j, E_l) < d(E_i, E_j)$. Якщо два приклади утворюють зв'язок Томека, то або один з цих прикладів є шумом, або обидва приклади є прикладами на межі. Зв'язки Томека можуть використовуватися як метод недооцінювання або як метод очищення даних. Як метод недооцінювання,

видаляються лише приклади, які належать до більшості класів, а як метод очищення даних. приклади обох класів видаляються [15].

Хоча збільшення кількості прикладів меншої категорії може збалансувати розподіл класів, деякі інші проблеми, які зазвичай присутні в наборах даних із спотвореним розподілом класів, не вирішуються. Часто кластери класів не є чітко визначеними, оскільки деякі приклади класу більшості можуть вторгатися в простір меншої категорії. Протилежне також може бути правдою, оскільки інтерполяція прикладів меншої категорії може розширити кластери меншої категорії, вводячи штучні приклади меншої категорії занадто глибоко в простір класу більшості. Індукція класифікатора в такій ситуації може призвести до перенавчання.

ADASYN. це метод збільшення меншості, розроблений спеціально для вирішення проблеми незбалансованості даних у задачах класифікації. Він створює синтетичні приклади меншості на основі адаптивного збільшення, що враховує щільність околиці прикладів меншості. Алгоритм ADASYN працює за наступними кроками, що наведено на рисунку (Рис. 1.4) зі статті [16]:

Procedure

(1) Calculate the degree of class imbalance:

$$d = m_s/m_l \tag{1}$$

where $d \in (0,\ 1]$.

(2) If $d < d_{th}$ then (d_{th} is a preset threshold for the maximum tolerated degree of class imbalance ratio):

(a) Calculate the number of synthetic data examples that need to be generated for the minority class:

$$G = (m_l - m_s) \times \beta \tag{2}$$

Where $\beta \in [0,\ 1]$ is a parameter used to specify the desired balance level after generation of the synthetic data. $\beta = 1$ means a fully balanced data set is created after the generalization process.

(b) For each example $\boldsymbol{x_i} \in minorityclass$, find K nearest neighbors based on the Euclidean distance in n dimensional space, and calculate the ratio r_i defined as:

$$r_i = \Delta_i/K, \quad i = 1, ..., m_s \tag{3}$$

where Δ_i is the number of examples in the K nearest neighbors of $\boldsymbol{x}_i$ that belong to the majority class, therefore $r_i \in [0,\ 1]$;

(c) Normalize r_i according to $\hat{r}_i = r_i / \sum_{i=1}^{m_s} r_i$, so that $\hat{r}_i$ is a density distribution ($\sum_i \hat{r}_i = 1$)

(d) Calculate the number of synthetic data examples that need to be generated for each minority example $\boldsymbol{x}_i$:

$$g_i - \hat{r}_i \times G \tag{4}$$

where G is the total number of synthetic data examples that need to be generated for the minority class as defined in Equation (2).

(e) For each minority class data example $\boldsymbol{x}_i$, generate g_i synthetic data examples according to the following steps:

Do the **Loop** from 1 to g_i:

(i) Randomly choose one minority data example, $\boldsymbol{x}_{zi}$, from the K nearest neighbors for data $\boldsymbol{x}_i$.

(ii) Generate the synthetic data example:

$$\boldsymbol{s_i} = \boldsymbol{x_i} + (\boldsymbol{x_{zi}} - \boldsymbol{x_i}) \times \lambda \tag{5}$$

where $(\boldsymbol{x_{zi}} - \boldsymbol{x_i})$ is the difference vector in n dimensional spaces, and λ is a random number: $\lambda \in [0,\ 1]$.

End **Loop**

Рис. 1.4. Псевдокод ADASYN

Основна ідея алгоритму ADASYN полягає в використанні розподілу щільності $\widehat{\tau_l}$ (Рис. 1.4, формула 3) як критерію для автоматичного визначення кількості синтетичних зразків, які необхідно створити для кожного прикладу даних меншої категорії. Фізично, $\widehat{\tau_l}$ – це вимірювання розподілу ваг для різних прикладів меншої категорії залежно від їх рівня складності навчання. Результуючий набір даних після ADASYN не тільки забезпечить збалансоване представлення розподілу даних (відповідно до бажаного рівня балансу, заданого коефіцієнтом β), але й змусить алгоритм навчання зосередитись на тих важких

для навчання прикладах. Це головна відмінність порівняно з алгоритмом SMOTE [17], в якому однакова кількість синтетичних зразків генерується для кожного прикладу даних меншої категорії [16], але це може призвести до збільшення шуму в даних, оскільки створює більше синтетичних прикладів у складних регіонах, де менший клас перекривається з більшим класом. Створення синтетичних даних може бути часозатратним, особливо для великих наборів даних

SMOTEEN. цей метод поєднує здатність SMOTE генерувати синтетичні приклади для класу меншості та здатність ENN видаляти деякі спостереження з обох класів, які ідентифіковані як такі, що мають різні класи між класом спостереження та його K-найближчим сусіднім класом більшості.

ENN (Edited Nearest Neighbors). це метод очищення даних, який видаляє приклади, які можуть бути шумовими або помилково класифікованими. ENN перевіряє клас кожного прикладу за допомогою його найближчих сусідів. Якщо більшість сусідів належить до іншого класу, приклад видаляється. Це допомагає видалити області перекриття між класами та забезпечити чіткіше розділення. SMOTEENN спочатку застосовує SMOTE для створення синтетичних прикладів меншості, а потім використовує ENN для очищення як меншого, так і більшого класів. Цей комбінований підхід має на меті покращити якість класифікації незбалансованих наборів даних, забезпечуючи краще розподілення класів та чіткіше розділення між ними, але слід не забувати, що цей алгоритм може бути ще більш часозатратним, ніж SMOTE та ADASYN, оскільки включає як генерацію синтетичних прикладів, так і процес очищення ENN. Може видаляти занадто багато прикладів під час процесу очищення, що призводить до втрати корисної інформації.

Також можна використовувати методи ваг класів та ваг у зразках даних. Метод ваг класів полягає в наданні більшої ваги меншому класу шляхом зміни ваг кожного класу в функції втрат або ваг у зразках даних. Метод ваг у зразках полягає в додаванні ваг до кожного зразка, щоб збалансувати дані.

Одним з недавно розроблених методів балансування даних є focal loss. це кастомна функція втрат, яка дозволяє зосередитися на надзвичайно складних екземплярах, тобто на тих, які мають велику помилку навіть після навчання на багатьох даних. Вона використовує вагу для кожного класу, залежно від його важливості, що дозволяє зменшити вплив легко класифікованих зразків на процес навчання та покращити якість моделі (1.3). Тобто формально додає фактор $-(1-p_t)^{\gamma}$ до стандартного cross entropy критерію. Встановлюючи $\gamma > 0$ і зменшуючи відносні втрати для добре класифікованих прикладів $p_t > 0.5$, $\gamma \geq 0$ параметр, який можна змінювати.

$$FL(p_t) = -(1-p_t)^{\gamma}\log(p_t) \quad (1.3)$$

Іншим методом балансування даних є використання функцій втрат з вагами для кожного класу. Weighted loss functions. це кастомні функції втрат, які дозволяють надавати більшу вагу помилкам на меншому класі. Вони використовуються для того, щоб збільшити значення помилки на меншому класі, що допомагає збалансувати дані [2].

Weighted binary cross-entropy (WBCE). це один з прикладів weighted loss functions. Ця функція втрат надає вагу помилкам на меншому класі, забезпечуючи більшу точність прогнозування на меншому класі [2]. Зважена перехресна ентропія (1.5) застосовує параметр масштабування альфа до бінарної перехресної ентропії (1.4), що дозволяє нам суворіше карати помилкові позитивні або помилкові негативні результати. Якщо необхідно, щоб FP (false positives) каралися більше, ніж FN (false negatives), альфа має бути більшим за 1. В іншому випадку він має бути меншим за 1.

$$L_{BCE} = -ylog\big(\hat{y}(p)\big) - (1-y)\log(1-\hat{y}(p)) \quad (1.4)$$

$$L_{WBCE} = -\alpha y log(\hat{y}(p) - (1-y)\log(1-\hat{y}(p)) \quad (1.5)$$

Далі необхідно знайти відповідно градієнт і гесіан.

Градієнт – перша похідна функції втрат (1.6), показує напрямок найбільшого зростання функції втрат. В контексті оптимізації моделі машинного навчання, градієнт використовується для оновлення ваг моделі таким чином, щоби мінімізувати функцію втрат.

$$G_{WBCE}(p) = \frac{\partial L_{WBCE}}{\partial p} = -wy\left(\frac{1}{p}\right) + (1-y)(\frac{1}{1-p}) \quad (1.6)$$

Гесіан – друга похідна функції втрат (1.7), показує кривизну функції втрат. В контексті оптимізації моделі машинного навчання, гесіан використовується алгоритмами оптимізації, які враховують другий порядок інформації, такі як метод Ньютона та квазиньютонівські методи (наприклад, BFGS, L-BFGS). Гессіан дозволяє ефективніше визначити напрямок оптимізації та швидкість оновлення ваг моделі, тому що він враховує кривизну функції втрат.

$$H_{WBCE}(p) = \frac{\partial^2 L_{WBCE}}{\partial^2 p} = wy\left(\frac{1}{p^2}\right) + (1-y)(\frac{1}{(1-p)^2}) \quad (1.7)$$

У кожній задачі машинного навчання необхідно добре збалансувати дані, щоб отримати найкращі результати. Вибір методу залежить від типу даних та конкретного завдання машинного навчання. Для досягнення найкращих результатів може бути ефективним поєднання декількох методів балансування даних, таких як oversampling, undersampling, SMOTE, SMOTETOMEK, SMOTEEN, ADASYN, ваги класів та ваги у зразках даних.

1.6 Методи класифікації незбалансованих даних

Random Forest – розроблений Лео Брейманом [6], є групою необрізаних класифікаційних або регресійних дерев, зроблених з випадковий вибір зразків навчальних даних. У процесі індукції вибираються випадкові ознаки. Прогноз робиться шляхом агрегування (більшість голосів за класифікація або усереднення для регресії) передбачення ансамблю. Кожне дерево вирощується, як описано в [7]:

1. Шляхом вибірки N випадковим чином, якщо кількість випадків в навчальному наборі N але з заміною, від вихідні дані. Цей зразок буде використовуватися як навчальний набір для вирощування дерева.
2. Для M кількості вхідних змінних, змінна m вибрано так, що $m \ll M$ вказано на кожному вузол, m змінних вибираються випадковим чином із M і найкращий розподіл на цих m використовується для розщеплення вузла. Під час вирощування лісу в значення m залишається постійним.
3. Кожне дерево вирощується в максимально можливій мірі. Обрізка не використовується.

Нижче на рисунку (Рис. 1.5) представлений псевдокод, що було взято зі статті [14].

Algorithm 1 Random Forest

Precondition: A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, features F, and number of trees in forest B.

```
function RandomForest(S, F)
    H ← ∅
    for i ∈ 1, ..., B do
        S^(i) ← A bootstrap sample from S
        h_i ← RandomizedTreeLearn(S^(i), F)
        H ← H ∪ {h_i}
    end for
    return H
end function
function RandomizedTreeLearn(S, F)
    At each node:
        f ← very small subset of F
        Split on best feature in f
    return The learned tree
end function
```

Рис. 1.5. Псевдокод Random Forest

Випадковий ліс загалом показує значний підвищення продуктивності порівняно з одним деревом класифікатор, наприклад C4.5. Показник загальної помилки, який він дає, сприятливо порівнюється з AdaBoost, проте він більш стійкий до шуму.

AdaBoost – це алгоритм ансамблю, який адаптивно налаштовується на помилки слабких гіпотез. У процесі навчання будує композицію з базових алгоритмів навчання поліпшення їх ефективності. AdaBoost є алгоритмом адаптивного бустингу тому, що кожен наступний класифікатор будується по об'єктах, які погано класифікуються попередніми класифікаторами.

AdaBoost викликає слабкий класифікатор у циклі. Після кожного виклику оновлюється розподіл ваг, які відповідають важливості кожного з об'єктів навчальної множини для класифікації. На кожній ітерації ваги кожного неправильно класифікованого об'єкта зростають, у такий спосіб новий класифікатор «фокусує свою увагу» на цих об'єктах [8]. Друкер, Шапіре та Сімар [9] у своїх експериментах, які вони проводили на реальній нейронній мережі, помітили, що підсумування виходів нейронних мереж, а потім з неї обирати

найкращий прогноз працює краще, ніж обирати кращий прогноз з кожної нейронної мережі, а потім комбінувати їх, використовуючи мажоритарне правило.

Нижче представлений псевдокод алгоритму AdaBoost (Рис. 1.6), який було запозичено зі статті [12].

Algorithm 1. *AdaBoost [8]*

1. *Initialize the observation weights* $w_i = 1/n$, $i = 1, 2, \ldots, n$.
2. *For* $m = 1$ *to* M:
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*

$$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$

   (c) *Compute*

$$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}.$$

   (d) *Set*

$$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),$$

   for $i = 1, 2, \ldots, n$.
   (e) *Re-normalize* w_i.
3. *Output*

$$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

Рис. 1.6. Псевдокод алгоритму AdaBoost

XGBoost [11] – це ансамбль дерева рішень, заснований на градієнтному посиленні. Подібно до посилення градієнта, XGBoost створює доповнення розширення цільової функції шляхом мінімізації функції втрат. Враховуючи що XGBoost зосереджений лише на деревах рішень як базових класифікаторах, варіації функції втрат використовується для контролю складності дерев.

Крім того, XGBoost впроваджує кілька методів для збільшення швидкості навчання дерев рішень, які не пов'язані безпосередньо з точністю ансамблю. Зокрема, XGBoost зосереджується на зменшенні обчислювальної складності при пошуку найкращого розбиття, що є найбільш часозатратною частиною алгоритмів побудови дерев рішень. Алгоритми пошуку розбиття зазвичай перебирають всі

можливі кандидатські розбиття і вибирають те, яке має найбільший приріст. Це вимагає проведення лінійного сканування кожного відсортованого атрибуту, щоб знайти найкраще розбиття для кожного вузла. Щоб уникнути повторного сортування даних у кожному вузлі, XGBoost використовує специфічну стислу структуру на основі стовпців, в якій дані зберігаються відсортованими [10]. Нижче представлений псевдокод XGBoost (Рис. 1.7), який було взято зі статті [13].

Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$, a differentiable loss function $L(y, F(x))$, a number of weak learners M and a learning rate α.

Algorithm:

1. Initialize model with a constant value:

$$\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta).$$

2. For m = 1 to M:

   1. Compute the 'gradients' and 'hessians':

$$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$$

$$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$$

   2. Fit a base learner (or weak learner, e.g. tree) using the training set $\left\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right\}_{i=1}^{N}$ by solving the optimization problem below:

$$\hat{\phi}_m = \arg\min_{\phi \in \Phi} \sum_{i-1}^{N} \frac{1}{2} \hat{h}_m(x_i) \left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i)\right]^2.$$

$$\hat{f}_m(x) = \alpha \hat{\phi}_m(x).$$

   3. Update the model:

$$\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x).$$

3. Output $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x).$

Рис. 1.7. Псевдокод XGBoost

1.7 Метрики якості моделей класифікації

Метрики якості моделей класифікації допомагають оцінити, наскільки добре модель працює на задачі класифікації. Деякі з основних метрик якості класифікації, які будуть використані при оцінці якості класифікації буде наведено нижче.

Матриця помилок (Confusion Matrix). це таблиця, яка відображає кількість правильних та неправильних класифікацій, зроблених моделлю класифікації.

Матриця помилок має наступну структуру (Таблиця 1.1) для бінарної класифікації:

Таблиця 1.1

Структура матриці помилок

	$y = 1$	$y = 0$
$\hat{y} = 1$	TP	FP
$\hat{y} = 0$	FN	TN

де $\hat{y}$. відповідь алгоритму на об'єкті; y – справжня мітка класу на цьому об'єкті; TP (True Positive). кількість дійсно позитивних прикладів, які модель правильно класифікувала; FP (False Positive). кількість дійсно негативних прикладів, які модель помилково класифікувала як позитивні; FN (False Negative). кількість дійсно позитивних прикладів, які модель помилково класифікувала як негативні; TN (True Negative). кількість дійсно негативних прикладів, які модель правильно класифікувала.

Для багатокласової класифікації матриця помилок має розмірність KxK, де K. кількість класів. Елементи на діагоналі відображають правильні класифікації, а елементи поза діагоналлю. неправильні класифікації між класами.

Матриця помилок корисна для аналізу моделі, оскільки вона дозволяє виявити які класи модель плутає, а також які типи помилок (FP, FN) модель здійснює найчастіше.

Точність (Accuracy). це метрика якості моделі класифікації, яка відображає відсоток правильно класифікованих прикладів відносно загальної кількості

прикладів (1.7) [4]. Точність є однією з найпростіших метрик для оцінки роботи моделі класифікації.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1.7)$$

Але ця метрика недостатньо добре описує якість моделі, яка була побудована на наборі даних з незбалансованими класами. Нехай, перед нами поставлена задача оцінити якість класифікації щодо виконаних транзакцій (шахрайська, не шахрайська). Наша ціль – визначати шахрайні транзакції. Є набір даних де 3000 не шахрайських транзакцій, 2900 з яких класифікатор визначив правильно (True Negative = 2900, False Positive = 100), а також 100 шахрайських транзакцій, 50 з яких класифікатор також визначив правильно (True Positive = 50, False Negative = 50). Тоді ми отримаємо наступну точність:

$$Accurace = \frac{50 + 2900}{50 + 2900 + 100 + 50} = 0{,}95$$

Але, якщо ми просто будемо помічати усі транзакції як не шахрайськи, то отримаємо більш високу оцінку точності:

$$Accurace = \frac{0 + 3000}{0 + 3000 + 0 + 100} = 0{,}97$$

При цьому, наша модель зовсім неспроможна роботи прогноз відповідно до вимог задачі.

Precision Score – цю метрику можна інтерпретувати як долю об’єктів, що названо класифікатором позитивним і при цьому дійсно належать до позитивного класу. Формула (1.8) для обчислення цієї метрики наведено нижче. Ця метрика використовується у парі з Recall Score для оцінки роботи моделі прогнозування машинного навчання.

$$Precision = \frac{TP}{TP + FP} \quad (1.8)$$

Recall Score – ця метрика показує, яку долю об’єктів позитивного класу з усіх об’єктів позитивного класу класифікував використаний алгоритм. Для обчислення цієї метрики використовується формула (1.9) [4].

$$Recall = \frac{TP}{TP + FN} \quad (1.9)$$

Візуалізація інтерпретації метрики представлена на рисунку нижче (Рис. 1.5).

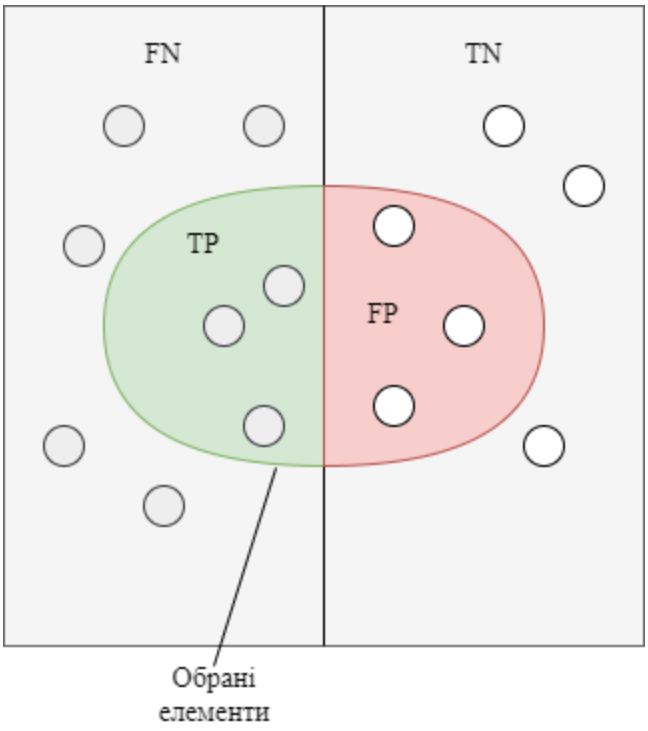


Рис. 1.8. Візуалізація метрик Precision і Recall

Precision і Recall score – дві важливі метрики для оцінки якості моделей класифікації, особливо в задачах з незбалансованими наборами даних. Як вже було розглянуто вище, то ці дві метрики вимірюють різні аспекти класифікації.

Високе значення метрики Precision свідчить про те, що модель не дуже часто помиляється, визначаючи позитивні приклади. Однак, модель з високою точністю може мати низьку повноту, якщо вона пропускає позитивні приклади.

А високе значення Recall свідчить про те, що модель відносно нечасто пропускає позитивні приклади. Однак, модель з високою повнотою може мати низьку точність, якщо вона помиляється, визначаючи негативні приклади.

Precision і Recall не залежить, на відміну Accuracy, від співвідношення класів і тому їх можна використовувати в умовах незбалансованих наборів даних.

Часто у реальній практиці стоїть завдання знайти оптимальний (для замовника) баланс між цими двома метриками. Класичним прикладом є завдання визначення відтоку клієнтів.

Очевидно, що ми не можемо знаходити усіх клієнтів, що йдуть у відтік, і тільки їх. Але, визначивши стратегію та ресурс для утримання клієнтів, ми можемо підібрати потрібні пороги по Precision та Recall клієнтів або тих, хто піде з більшою ймовірністю, оскільки ми обмежені ресурсами колл-центру.

F Score – поєднання Precision та Recall в єдине число за допомогою гармонійного середнього. F Score допомагає оцінити загальну ефективність моделі класифікації, коли обидві метрики важливі для задачі. Особливо корисною F Score є при роботі з незбалансованими наборами даних, коли необхідно за умовами задачі враховувати як точність, так і повноту. Ця метрика знаходиться за формулою (1.10) [4].

$$F_\beta\ Score = (1 + \beta^2)\ \frac{Precision * Recall}{(\beta^2 * Precision) + Recall} \quad (1.10)$$

де β. це параметр, який визначає відносну важливість Precision і Recall. Значення $\beta > 1$ надає більшу вагу повноті, а значення $\beta < 1$ надає більшу вагу точності.

F Score дозволяє налаштовувати відносну важливість точності та повноти для конкретної задачі, що допомагає знайти оптимальний баланс між ними та оцінити якість моделі класифікації.

При конвертації дійсної відповіді алгоритму класифікації у бінарну мітку, ми повинні обрати якийсь допустимий поріг, при якому 0 буде становитися 1. Звісно, що близьким порогом здається число 0.5, але він не завжди є оптимальним. Наприклад, при незбалансованості класів у наборі даних.

Одним з способів оцінити модель в цілому, не прив’язуючись до конкретного порогу є AUC-ROC (Area Under Curve) – площина під кривою

похибок. Дана крива представляє собою лінію від (0;0) до (1;1) в координатах True Positive Rate (TPR) (1.11) і False Positive Rate (FPR) (1.12) [4]:

$$TPR = \frac{TP}{TP + FN} \quad (1.11)$$

$$FPR = \frac{FP}{FP + TN} \quad (1.12)$$

TPR – вже відома метрика, це Recall (1.9), а FPR показує, яку долю з об'єктів Negative класу алгоритм класифікував невірно. В ідеальному випадку, коли класифікатор не робить помилок, тобто FPR = 0, TPR = 1, то ми отримаємо площу під кривою, яка рівна 1; у іншому випадку, коли класифікатор буде класифікувати випадково вірогідності класів, то площа кривої буде наближатися до 0,5, так як класифікатор буде повертати однакову кількість TP і FP.

Кожна точка графіки відповідає вибору деякого порога. Площа під кривою у разі показує якість алгоритму (більше — краще), крім цього, важливою є крутість самої кривої — ми хочемо максимізувати TPR, мінімізуючи FPR, отже, наша крива в ідеалі має наближатися до точки (0;1).

Середнє геометричне (G-Mean) – це показник, який вимірює баланс між класифікацією виступи як для більшості, так і для меншості (1.13). Низький G-Mean є показником поганого стану ефективність у класифікації позитивних випадків, навіть якщо негативні випадки класифіковані правильно як такий. Цей захід важливий для уникнення перепідгонки негативного класу та недостатньої підгонки позитивний клас [4].

$$G - Mean = \sqrt{Recall * (\frac{TN}{TN + FP})} \quad (1.13)$$

Висновки

В ході вивчення теоретичного матеріалу кваліфікаційної роботи було розглянуто наступні основні аспекти:

Дослідження бізнес-кредитів та особливості набору даних, зокрема, виявлення ключових характеристик та властивостей даних, що впливають на якість класифікації.

Методи підвищення якості моделей класифікації, включаючи різні техніки, що можуть покращити якість моделей класифікації, такі як відбір ознак, оптимізація гіперпараметрів та використання користувацьких функцій витрат.

Проблема незбалансованості даних у задачах класифікації та її вплив на моделі класифікації. Було розглянуто причини та наслідки незбалансованості даних.

Аналіз методів балансування даних, таких як випадкове збільшення меньшої класи (oversampling), випадкове зменшення більшої класи (undersampling) та поєднання цих методів.

Методи класифікації незбалансованих даних, зокрема алгоритми, стійкі до незбалансованих даних, такі як Random Forest, Gradient Boosting, AdaBoost та XGBoost.

Метрики якості моделей класифікації, такі як точність (Accuracy), площа під кривою (AUC-ROC), повнота (Recall), точність (Precision) та F-міра, що допомагають оцінити ефективність моделей класифікації на незбалансованих даних.

На підставі вивчення теоретичного матеріалу було отримано розуміння проблеми незбалансованих даних у задачах класифікації та особливостей бізнес-позик. Викладений теоретичний базис дає можливість виробити ефективний підхід до вирішення задачі прогнозування дефолту бізнес позик на основі незбалансованих даних.

У подальшому розділі і подальших підрозділах кваліфікаційної роботи буде розглянуто практичне застосування цих методів і технік для побудови

класифікаційної моделі, яка зможе ефективно прогнозувати дефолти бізнес позик, враховуючи незбалансованість даних.

2 СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Постановка задачі

2.1.1 Призначення дослідження

Дослідження, що було проведено у даній кваліфікаційній роботі може слугувати підґрунтям для подальшої роботи з подібними наборами даних для класифікації працівниками банку дефолту клієнтів у бізнес сфері.

Призначенням дослідження є впровадження та імплементація розглянутих методів, які найбільше підходять до вирішення задачі з класифікації, а також підходять для розглянутого нижче набору даних.

2.1.2 Цілі та задачі дослідження

З урахуванням вступу та актуальності дослідження, метою даної кваліфікаційної роботи є дослідження різних методів класифікації незбалансованих даних на прикладі прогнозування дефолту бізнес позик. Для досягнення цієї мети, потрібно вирішити наступні задачі:

1. Вивчити теоретичні аспекти класифікації даних, проблему незбалансованості класів та їх вплив на якість класифікації.
2. Огляд та аналіз існуючих методів класифікації незбалансованих даних, включаючи їх переваги та недоліки.
3. Вибір та адаптація набору даних з реальними даними про бізнес позики банку для дослідження. Проведення попередньої обробки даних, включаючи очищення та підготовку (IDA, EDA, CDA).

4. Застосування різних методів класифікації на обраному наборі даних, зокрема алгоритмів, що враховують незбалансованість класів.
5. Оцінка якості різних методів класифікації на основі відповідних метрик, що показують ефективність моделей на незбалансованих даних.
6. Порівняння результатів та вибір найкращих методів класифікації для прогнозування дефолту бізнес позик.
7. Висновки щодо результатів дослідження в банківській сфері.

2.2 Вибір технологій для дослідження.

На сьогоднішній день існує безліч інструментів для проведення аналізу даних, а ще безліч для створення класифікаційних моделей, тому щоб ефективно проводити аналіз даних та вирішувати задачі прогнозування або класифікації, необхідно обрати відповідну мову програмування та інструменти. Вибір залежить від конкретних вимог до проекту, а також від переваг та недоліків кожної технології.

Python – це високорівнева мова програмування, яка популярна в області науки про дані через свою читабельність, гнучкість та різноманітність інструментів. Python має багато бібліотек для аналізу даних, включаючи:

- Pandas: дозволяє працювати з даними у вигляді таблиць, проводити їх очищення, трансформацію, агрегацію та аналіз.
- NumPy: використовується для обробки великих масивів даних.
- Matplotlib та Seaborn: ці бібліотеки використовуються для візуалізації даних.
- Scikit-learn: включає багато алгоритмів для машинного навчання, включаючи класифікацію, регресію, кластеризацію і так далі.
- TensorFlow та PyTorch: ці бібліотеки використовуються для глибокого навчання.

R – це мова програмування, спеціально розроблена для статистичного аналізу даних та візуалізації. Вона має багато пакетів для різних задач аналізу даних:

- dplyr: дозволяє проводити різноманітні операції з даними, такі як фільтрація, сортування, агрегація, і так далі.
- ggplot2: одна з найбільш потужних бібліотек для візуалізації даних.
- caret: центральний пакет для машинного навчання в R, який надає інтерфейс для сотень алгоритмів машинного навчання.

Julia – це високопродуктивна мова програмування, розроблена спеціально для наукових обчислень. Вона поєднує швидкість C зі зручністю використання Python. Ось деякі пакети Julia для аналізу даних:

- DataFrames: цей пакет надає структури даних для роботи з таблицями даних.
- Plots: це гнучка бібліотека візуалізації, яка підтримує багато різних бекендів.
- MLJ: це потужна бібліотека для машинного навчання, яка поєднує багато різних алгоритмів в єдиному інтерфейсі.

Звичайно, кожна мова програмування та її пакети мають свої особливості та можливі недоліки.

Пакети, такі як Pandas, можуть бути важкими для новачків і вимагати певного часу для навчання. Python може бути повільним у порівнянні з мовами, такими як C++ або Java.

Незважаючи на те, що Python є загальноприйнятим мовою для data science, він не є оптимальним для розробки програмного забезпечення великого масштабу або високопродуктивних систем.

R може бути складним для освоєння, особливо для людей без статистичного фону, а також не є ефективним для великих наборів даних через свої обмеження з пам'яттю.

R не є оптимальним для загальної розробки програмного забезпечення.

Хоча Julia є швидкою та гнучкою мовою, її спільнота та екосистема пакетів є значно меншими, ніж у Python або R.

Julia може бути менш підходящою для задач обробки даних, які не вимагають високої продуктивності.

Оскільки Julia є новою мовою, її синтаксис та стандарти можуть змінюватися, що може створювати труднощі для користувачів.

Для подальших досліджень з аналізу даних і вирішення задач прогнозування або класифікації буде використовуватися мова програмування Python. Причина вибору Python полягає в тому, що вона є загальноприйнятою мовою для аналізу даних, має широкий вибір бібліотек для цієї галузі, таких як Pandas, NumPy, Matplotlib, Seaborn та Scikit-learn, а також має багато інших корисних бібліотек для вирішення різноманітних задач. Python також має високу читабельність, гнучкість і легко зрозуміти синтаксис, що робить її популярною серед аналітиків даних та машинного навчання. Окрім того, Python має велику спільноту користувачів та розвинену інфраструктуру, що дозволяє легко знайти допомогу і матеріали для вивчення.

2.3 Початковий аналіз даних (IDA)

У ході дослідження буде використовуватися набір даних «loan_applic.csv», який є синтетичним. У ньому міститься інформація про стан позики у деяких клієнтів банку «N».

Змінні, які присутні у цьому наборі даних представлені нижче у таблиці Таблиця 2.1:

Таблиця 2.1

Змінні набору даних

Ім’я змінної	Значення
-1-	**-2-**

Application Number	Унікальний код заявки
Application: Buy Rate	Відсоткова ставка
Application: Funded Amount	Сума фінансування заявки
Application: BPA Broker Negotiation	

Продовження таблиці 2.1

-1-	**-2-**
Primary Contact Gender	Стать
Application: Close Date	Дата закриття заявки
customer Age	Вік клієнта
Amount	Сума позики
crime_record	Відмітка про злочин/и клієнта
Applications received All Time	Отримані заявки за весь час
Applications received by last 1 Month	Отримані заявки за 1 місяць
Applications received by last 3 Months	Отримані заявки за 3 місяці
Applications received by last 6 Months	Отримані заявки за 6 місяців
Average Daily Negatives	
Average Monthly Sales	Середні продажі за місяць
Avg Daily Bank Balance	Середній баланс банківського рахунку за місяць
Avg Number of Monthly Deposits	Усереднений за місяць залишок на депозитному рахунку
Brokers submitted All Time	Кількість заявок, що подано брокерами за весь час
Brokers submitted last 1 month	Кількість заявок, що подано брокерами за місяць

Brokers submitted last 3 months	Кількість заявок за 3 місяці
Brokers submitted last 6 months	Кількість заявок 6 місяців
Credit Score	Кредитоспроможність споживача
Daily Bank Balance v/s Daily Payment	Ця величина може впливати на здатність позичальника повертати кредитні кошти
Days	Кількість днів

Продовження таблиці 2.1

-1-	**-2-**
Has Website	Чи має клієнт веб-сторінку
Industry	Індустрія в якій працює клієнт
Inquiry Count	Кількість прохань на перегляд кредитного досье
Months	Кількість місяців
Number of Trade Lines	Діяльності для будь-якого типу кредиту, наданого позичальнику та наданого агентству кредитної звітності
Office Space	Чи наявний офіс
Open Bankruptcy	Чи об’явлене банкрутство
Position	Заставне забезпечення на власність
Public Records	Публічні записи (банкрутства / вид позики / податкові заборгованості / судові рішення)
Sales to Payment	Продажі до сплати
Satisfactory	Угода між позичальником і власником рахунку простроченої позики з детальним описом умов погашення.
Shipping State	Код штату США

Sum of Monthly Personal Debt	Сума місячного персонального боргу
Time In Business Actual	Кількість часу в бізнесі
Type	Тип заявки
Volume. 4 Months Ago	Загальний обсяг позики для кожної категорії позики за 4 місяці
Volume. 6 Months Ago	Загальний обсяг позики за 6 місяців

Продовження таблиці 2.1

-1-	**-2-**
Yearly Total Sales	Річна кількість продажів
Outcome	Чи закритий був кредит чи ні

Після ознайомлення з вмістом файлу «loan_applic.csv» наступним кроком на даному етапі є завантаження набору даних. Основним інструментом для завантаження і відображення набору даних у табличному вигляді є пакет «Pandas» мови програмування Python.

Спочатку необхідно також зазначити, що набір буде поділений на тестувальну та тренувальну вибірки шляхом діленням набору даних по даті. Тобто тренувальна вибірка буде складатися з елементів 2017 року, а тестувальна відповідно з 2018 року.

Оглянути дані можна на рис. 2.1.

Application: Buy Rate	Application: Funded Amount	Application: Origination Fee	Application: Remittance Frequency	Primary Contact Gender	Application: Close Date	customer Age	Amount	crime_record	Applications received All Time	...
1.25	9000.0	2.0	Daily	NaN	1/3/2017	35.0	9000.0	NaN	1.0	...
1.25	9500.0	2.0	Daily	Male	1/3/2017	47.0	9500.0	NaN	2.0	...
1.25	6000.0	2.0	Daily	Male	1/3/2017	58.0	6000.0	NaN	4.0	...
1.25	15000.0	2.0	Daily	Male	1/3/2017	62.0	15000.0	NaN	1.0	...
1.25	14000.0	2.0	Daily	Female	1/3/2017	64.0	14000.0	traffic	1.0	...

...	Satisfactory	Shipping State	Sum of Monthly Personal Debt	Time In Business Actual	Type	Volume - 4 Months Ago	Volume - 6 Months Ago	Volume - Three Months Ago	Yearly Total Sales	Outcome
...	40.0	NJ	4792.0	24.0	New Deal	0.00	0	11605.73	128688.72	Neg
...	24.0	OH	2867.0	17.7	New Deal	35205.42	29288.31	30209.32	351830.88	Neg
...	15.0	NY	1331.0	17.5	New Deal	6299.58	4799.65	9281.73	132113.4	Neg
...	8.0	MN	16491.0	5.1	New Deal	0.00	0	44114.52	448156.8	Pos
...	22.0	PA	1373.0	1.5	New Deal	0.00	0	16168.17	205465.32	Pos

Рис. 2.1. Перші 5 рядків даних

Отримана розмірність даних має такі значення 5 рядків і 45 стовпців (параметрів, змінних).

Наступним кроком на цьому етапі є отримання інформації про склад даних (Рис. 2.2). Ця інформація надасть чітке уявлення про склад даних і що саме потрібно робити на наступному кроці.

```
Output exceeds the size limit. Open the full output data in a text editor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9795 entries, 0 to 9794
Data columns (total 45 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Application: Buy Rate                 9795 non-null   float64
 1   Application: Funded Amount            9795 non-null   float64
 2   Application: Origination Fee          9795 non-null   float64
 3   Application: Remittance Frequency     9795 non-null   object
 4   Primary Contact Gender                9438 non-null   object
 5   Application: Close Date               9795 non-null   object
 6   customer Age                          9608 non-null   float64
 7   Amount                                9786 non-null   float64
 8   crime_record                          3678 non-null   object
 9   Applications received All Time        9793 non-null   float64
 10  Applications received by last 1 Month 9793 non-null   float64
 11  Applications received by last 3 Months 9793 non-null  float64
 12  Applications received by last 6 Months 9793 non-null  float64
 13  Average Daily Negatives               9795 non-null   float64
 14  Average Monthly Sales                 9795 non-null   object
 15  Avg Daily Bank Balance                9795 non-null   object
 16  Avg Number of Monthly Deposits        9795 non-null   float64
 17  Brokers submitted All Time            9793 non-null   float64
 18  Brokers submitted last 1 month        9793 non-null   float64
 19  Brokers submitted last 3 months       9793 non-null   float64
...
 43  Yearly Total Sales                    9795 non-null   object
 44  Outcome                               9795 non-null   object
dtypes: float64(29), object(16)
```

Рис. 2.2. Склад даних

Можна легко побачити, що у деяких колонках невірно визначений тип даних, що там зберігається. Наприклад, колонка з назвою «Average Monthly Sales» має тип даних object, що не відповідає дійсноті. Тип даних, що там зберігається повинен визначатися як float64.

Відразу можна однозначно додати першу задачу на наступний етап. Необхідно змінити тип даних з object на float64.

Поглянемо також на статистичний опис набору даних для отримання розуміння структури і характеристик даних. Статистичний опис перших 17 стовпців, які є числовими, набору даних представлено нижче на рис. 2.3. Продовження рис. 2.3. наведено (додаток Г, рис. Г.1).

	count	mean	std	min	25%	50%	75%	max
Application: Buy Rate	9795.0	1.30	0.06	1.09	1.25	1.28	1.38	1.50
Application: Funded Amount	9795.0	32622.28	49488.47	5000.00	9500.00	15000.00	30000.00	250000.00
Application: Origination Fee	9795.0	2.80	0.40	0.00	2.50	3.00	3.00	3.00
customer Age	9608.0	48.09	12.91	0.00	40.00	47.00	55.00	178.00
Amount	9786.0	32642.12	49551.65	56.00	9500.00	15000.00	30000.00	250000.00
Applications received All Time	9793.0	2.48	2.43	1.00	1.00	2.00	3.00	41.00
Applications received by last 1 Month	9793.0	1.14	0.44	1.00	1.00	1.00	1.00	8.00
Applications received by last 3 Months	9793.0	1.30	0.69	1.00	1.00	1.00	1.00	10.00
Applications received by last 6 Months	9793.0	1.59	1.02	1.00	1.00	1.00	2.00	26.00
Average Daily Negatives	9795.0	0.60	1.03	0.00	0.00	0.00	0.79	12.33
Avg Number of Monthly Deposits	9795.0	27.84	33.52	0.00	9.33	20.75	35.82	969.67
Brokers submitted All Time	9793.0	1.78	1.65	1.00	1.00	1.00	2.00	34.00
Brokers submitted last 1 month	9793.0	1.09	0.37	1.00	1.00	1.00	1.00	6.00
Brokers submitted last 3 months	9793.0	1.18	0.55	1.00	1.00	1.00	1.00	9.00
Brokers submitted last 6 months	9793.0	1.31	0.77	1.00	1.00	1.00	1.00	22.00
Credit Score	9767.0	610.37	70.08	0.00	557.00	615.00	662.00	816.00
Daily Bank Balance v/s Daily Payment	8472.0	45.59	66.98	0.02	14.38	27.33	51.88	1848.93

Рис. 2.3. Статистичний опис даних

Можна наочно побачити, що у стовпці під назвою Daily Bank Balance v/s Daily Payment кількість даних значно менша, ніж у інших стовпцях. Тому друге завдання буде формуватися стосовно того чи є необхідним даний стовпець.

А у стовпці «customer Age» можна спостерігати аномалію. Максимальне значення у цьому стовпці рівне 178, а мінімальне 0.

Наступним кроком на цьому етапі є пошук і відповідно обробка пропущених значень, що є найнеобхіднішим кроком через те, що багато інструментів для аналізу даних, просто не пристосовані для обробки виключень з пустими даними.

Перші 14 стовпців, у яких було помічено найбільшу кількість пропусків, представлено нижче на рисунку 2.4, а на рисунку 2.5 представлена графічна інтерпретація отриманих результатів. Продовження рис. 2.4 знаходиться відповідно до (додатку Д, рис. Д.1).

	Columns	Count	Percentage(%)
7	crime_record	4916	66.91
40	Volume - 6 Months Ago	2603	35.43
39	Volume - 4 Months Ago	1992	27.11
33	Sales to Payment	1379	18.77
21	Daily Bank Balance v/s Daily Payment	1232	16.77
37	Time In Business Actual	399	5.43
4	Primary Contact Gender	353	4.80
41	Volume - Three Months Ago	243	3.31
5	customer Age	185	2.52
26	Inquiry Count	48	0.65
32	Public Records	48	0.65
20	Credit Score	25	0.34
22	Days	9	0.12
23	Factor Rate	9	0.12

Рис. 2.4. Кількість пропущених значень

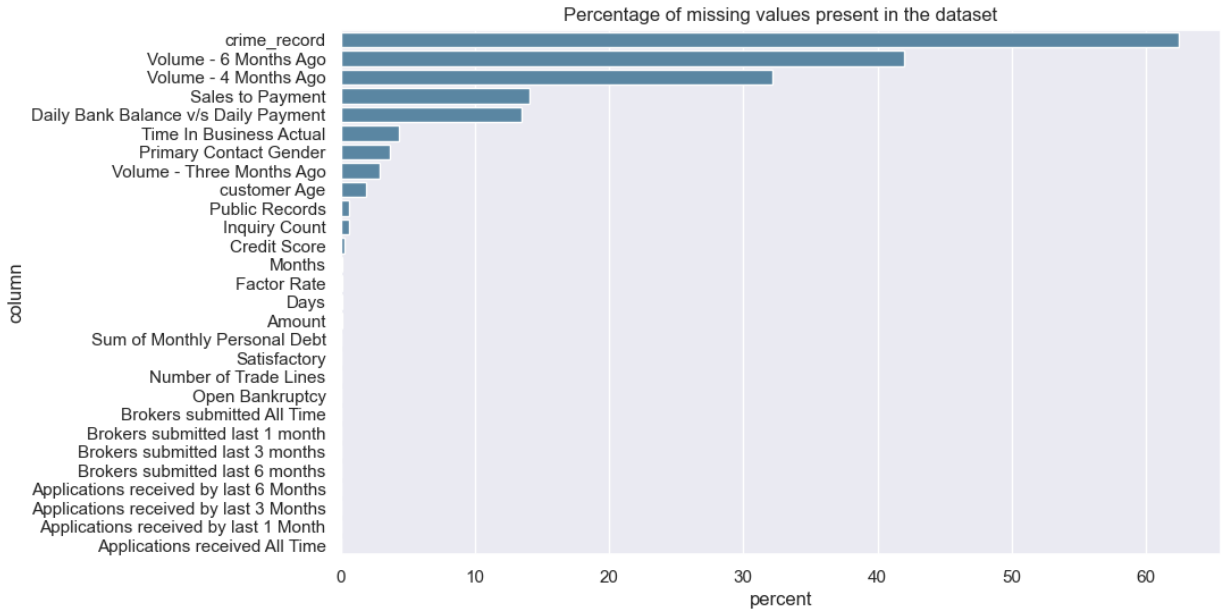


Рис. 2.5. Графічна інтерпретація результатів

Добре видно, що колонка під назвою «crime_record» є рекордсменом з пропущених значень. Більше ніж 50% значень є пустими.

Останнім кроком на даному етапі буде обробка пропущених значень. Категоріальні дані будуть заповнюватися модою, а числові – медіаною.

Також у наборі даних, що розглядається, було помічено те, що у деяких колонках є проблем з типом даних, а саме у колонках «Average Monthly Sales», «Avg Daily Bank Balance», «Volume. 4 Months Ago», «Volume. 6 Months Ago», «Volume. Three Months Ago», «Yearly Total Sales». На цьому кроці також зробимо приведення типів.

Після обробки пропущених значень маємо наступний набір даних (Рис. 2.6). На рисунку зображено перши 13 стовпчиків.

	Column	Sum
0	Application: Buy Rate	0
1	Application: Funded Amount	0
2	Application: Origination Fee	0
3	Application: Remittance Frequency	0
4	Primary Contact Gender	0
5	customer Age	0
6	Amount	0
7	crime_record	0
8	Applications received All Time	0
9	Applications received by last 1 Month	0
10	Applications received by last 3 Months	0
11	Applications received by last 6 Months	0
12	Average Daily Negatives	0

Рис. 2.6. Кількість пропущених значень

Даний етап вважається завершеним, так як виконані необхідні початкові маніпуляції з даними, тобто первинний аналіз, обробка пропущених значень. Наступний етап буде включати в себе повний статистичний аналіз даних.

2.4 Дослідницький аналіз даних (EDA)

На цьому етапі проводиться більш глибинний аналіз і проводиться формування гіпотез.

Так як у наборі даних представлено 45 змінних, тому графічне відображення немає сенсу. Але є сенс подивитися на розподіл (Рис. 2.7) класів відносно цільової змінної, яка зберігається у колонці під назвою «Outcome».

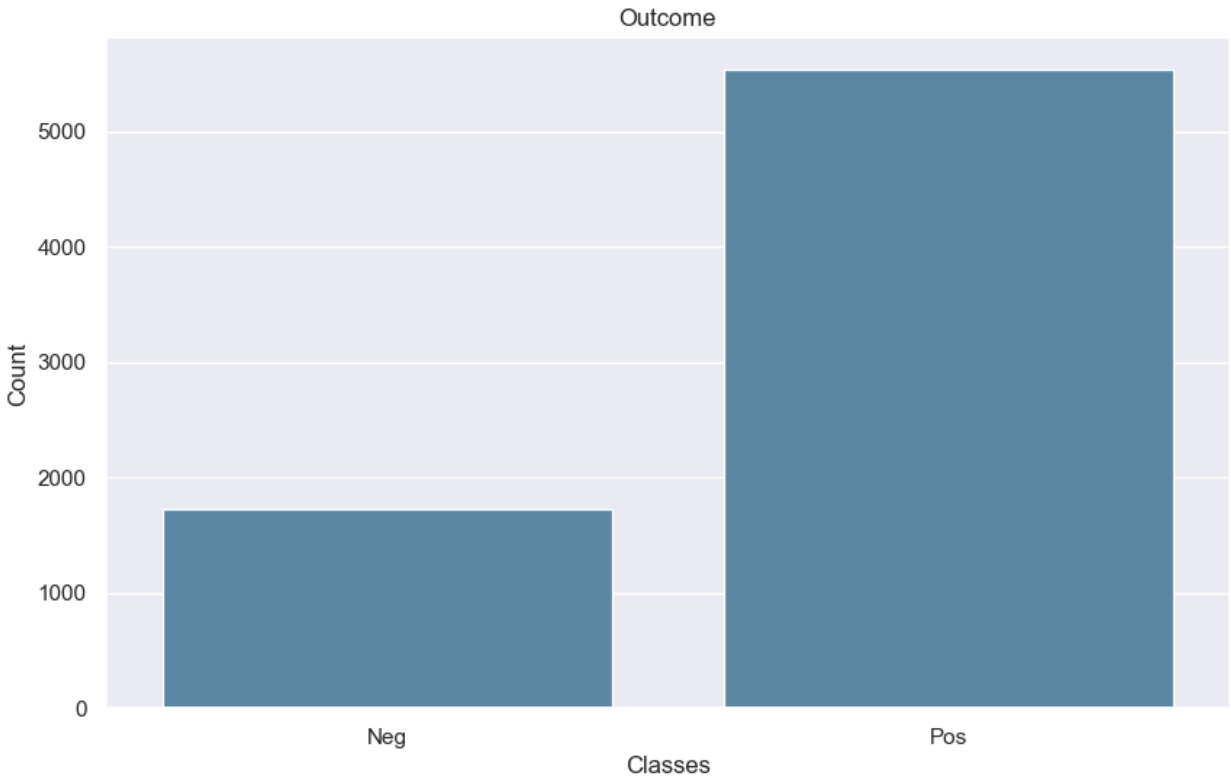


Рис. 2.7. Розподіл класів

У таблиці 2.2 відповідно наведено кількісний розподіл класів.

Таблиця 2.2

Розподіл класів

Клас	Кількість
Pos	5534
Neg	1732

Наочно можна впевнитися, що у даному наборі присутня незбалансованість класів. Відношення кількості позитивних екземплярів до негативних екземплярів рівна 4. Це говорить про невеликий дисбаланс даних.

Перевіримо чи існує мультиколінеарність між змінними у наборі даних. За допомогою пакета Pandas знайдемо кореляційну матрицю, а для обчислення кореляції буде використовуватися коефіцієнт Кендалла. Так як даний ранговий коефіцієнт є більш стійким до аномалій і викидів, тобто не потребує нормального ропозділу. Будуємо кореляційну карту (Рис. 2.8) за допомогою пакета seaborn.

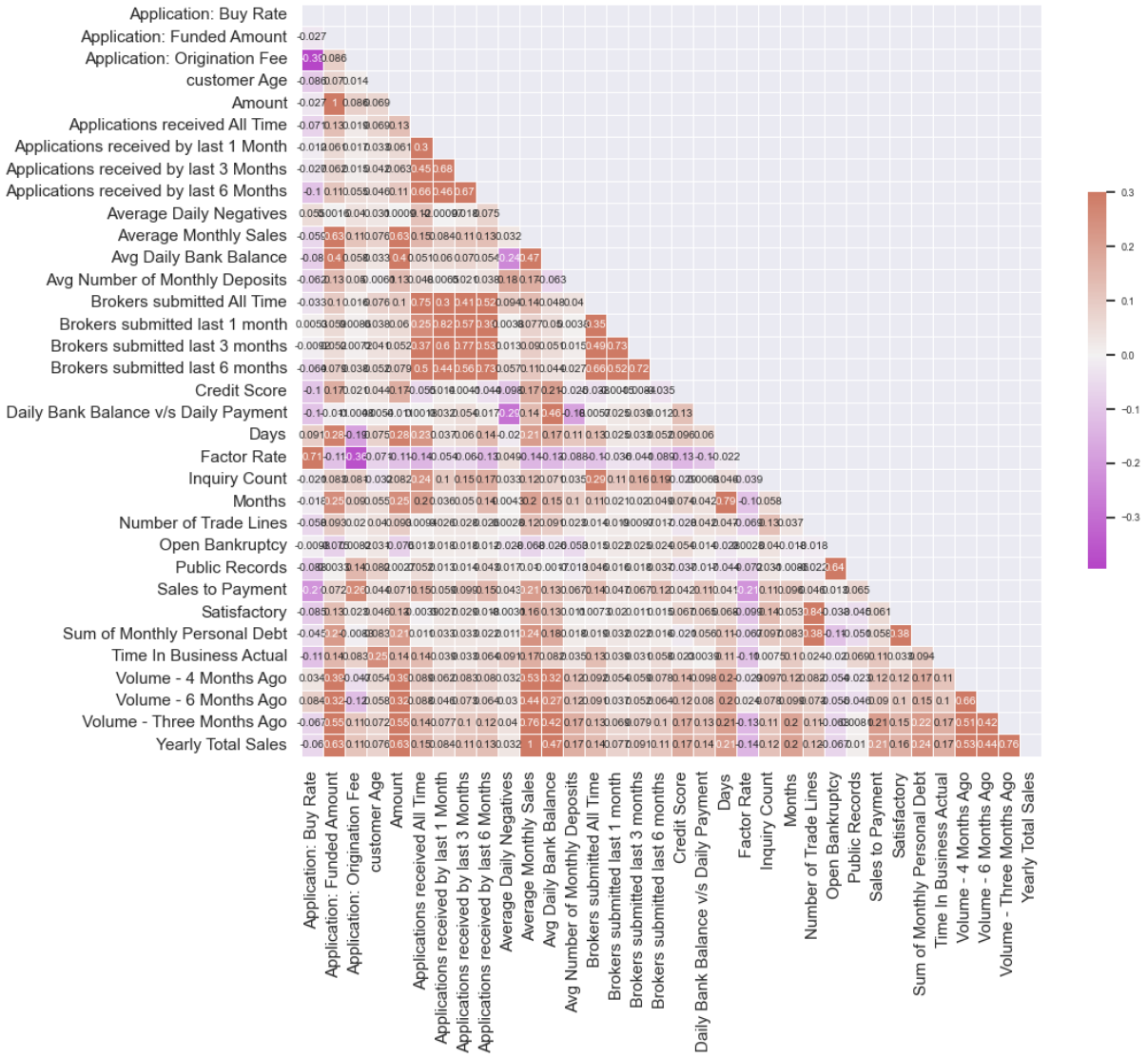


Рис. 2.8. Кореляційна карта

Можна помітити, що існують сильні зв’язки між «Amount» та «Application: Funded Amount», а також між «Yearly Total Sales» та «Average Monthly Sales», а також між «Months» та «Days», а також між «Satisfactory» та «Number Trade Lines», що рівні 1;1;0,84;0,83 відповідно.

Відповідно до цих зв’язків поставлені питання: «Чи мають вони сенс?», «Чи дублюється інформація у цих двох стовпчиках?». Будуємо діаграму розкиду для першого зв’язку (Рис. 2.9) за допомогою пакету seaborn.

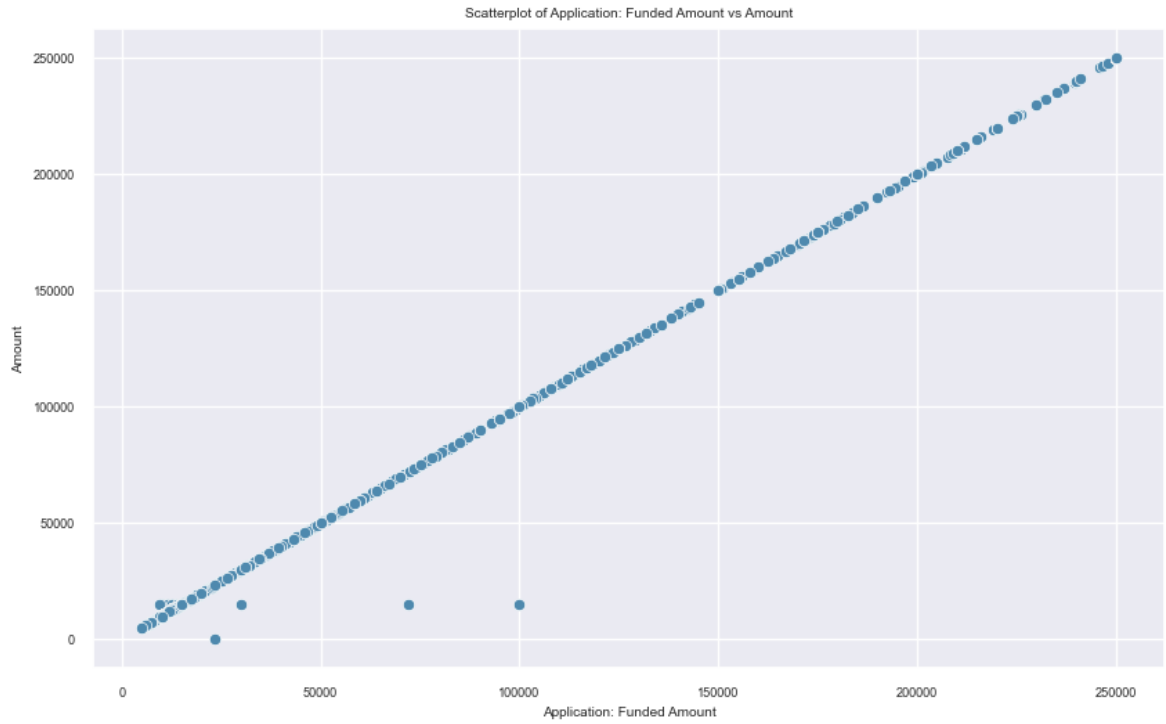


Рис. 2.9. Діаграма розкиду «Application:Funded Amount» і «Amount»

Можна побачити що точки утворюють пряму лінію з дуже високим кутом нахилу. Це свідчить про те, що змінні несуть однакову інформацію.

На рис. 2.10 представлена діаграма розкидку «Yearly Total Sales» та «Average Monthly Sales»

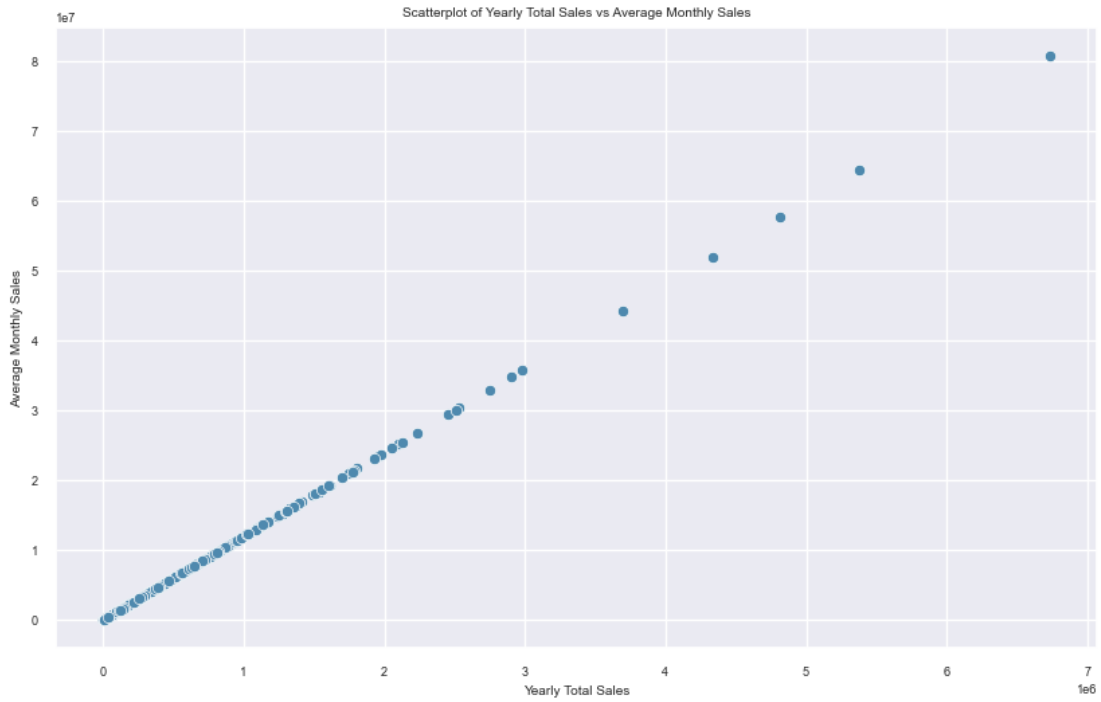


Рис. 2.10. Діаграма розкиду «Yearly Total Sales» і «Average Monthly Sales»

Чітко спостерігається інформація про те, що змінні несуть дублюючу інформацію.

На рис. 2.11 зображено діаграму розкиду для «Months» та «Days»

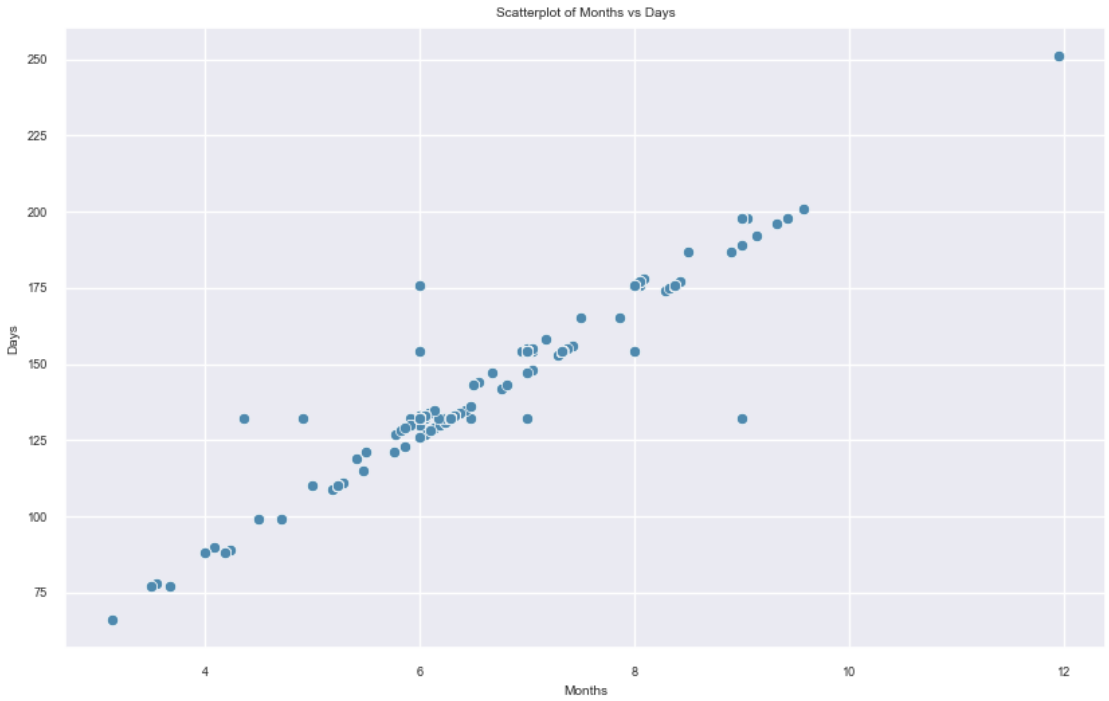


Рис. 2.11. Діаграма розкиду «Months» і «Days»

Можна також побачити, точки утворили пряму, але деякі з точок розміщено над прямою, що свідчить про те, що одна зі змінних має тенденцію зростати швидше, ніж інша. Тобто, якщо одна змінна збільшується, інша змінна також має тенденцію збільшуватися, але можливо, із різним коефіцієнтом зростання.

Розглянемо останню діаграму розкиду (Рис. 2.12) для «Satisfactory» та «Number of Trade Lines».

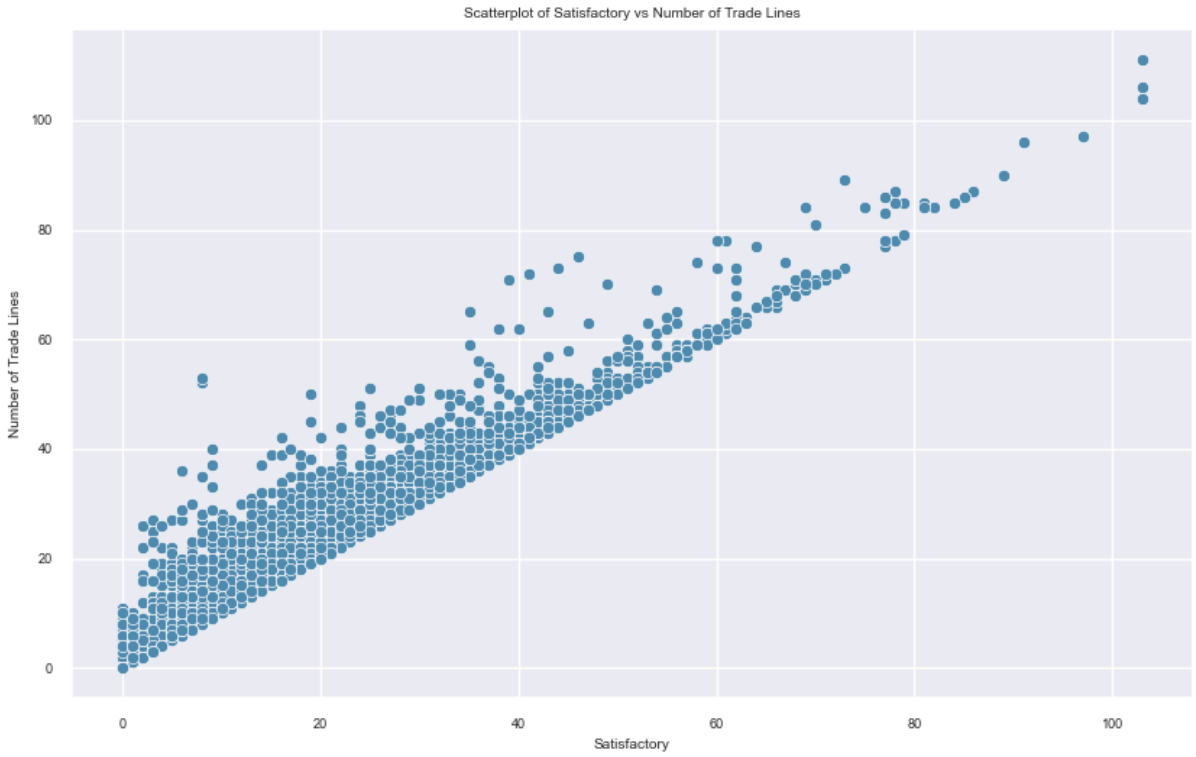


Рис. 2.12. Діаграма розкиду «Satisfactory» і «Number of Trade Lines»

На цьому рисунку добре видно, що також утворюється пряма, але, як же було запропоновану гіпотезу вище, можна зробити висновок, що тут одна зі змінних зростає швидше, ніж інша.

Подивимося на коефіцієнт інфляції дисперсії (рис. 2.13) для того, щоб впевнитися, що дійсно у наборі даних присутня мультиколінеарність. Якщо так, то і перевірити чи не існує можливої взаємодії між трьома або більше змінними.

На рис. 2.13 зображено перші 15 змінних.

	VIF Factor	features
10	828566.782079	Average Monthly Sales
33	828551.193380	Yearly Total Sales
4	2528.337076	Amount
1	2527.988511	Application: Funded Amount
19	2040.437014	Days
22	1975.572327	Months
0	1604.436045	Application: Buy Rate
20	1529.914805	Factor Rate
17	92.052053	Credit Score
14	78.063597	Brokers submitted last 1 month
15	65.940325	Brokers submitted last 3 months
23	59.161583	Number of Trade Lines
6	55.657530	Applications received by last 1 Month
27	51.664572	Satisfactory
7	45.948983	Applications received by last 3 Months

Рис. 2.13. VIF

Можна спостерігати, що перші дві змінні дійсно корелюють між собою, як і другі дві (Days, Months) тому на наступному кроці одну з них буде однозначно вилучено при подальшому дослідженні.

Подивимося на відношення клієнтів з різними видами власності «Office space» до «Outcome» (рис. 2.14).

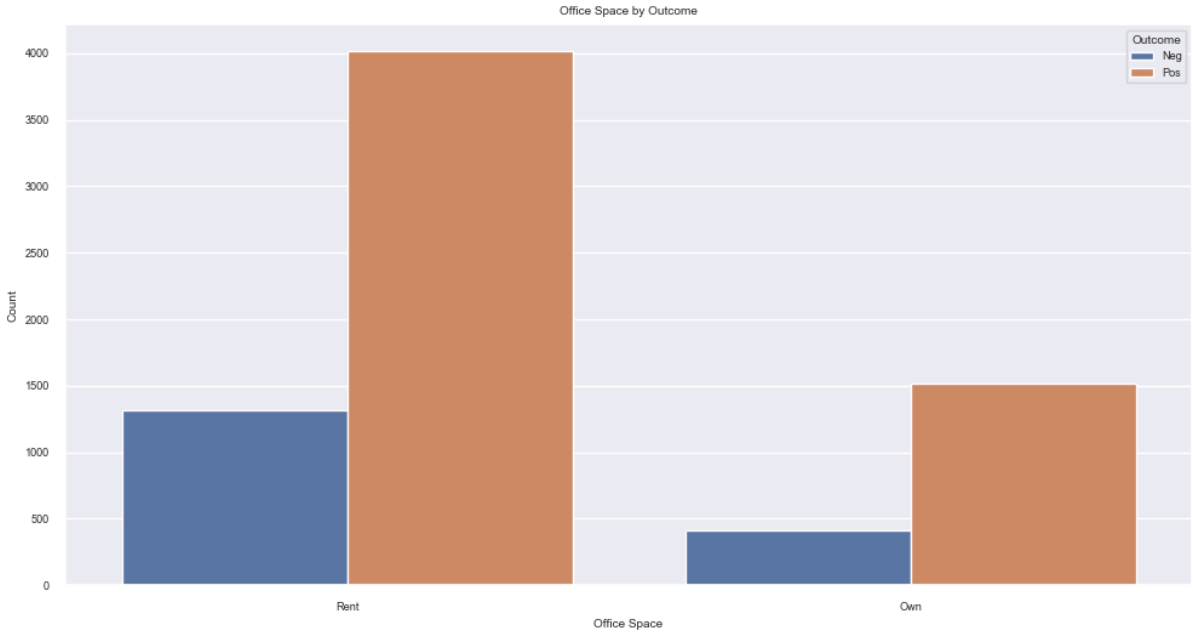


Рис. 2.14. Office space до Outcome

Можна побачити, що люди, які маю не орендований офіс мають:

По-перше, більше записів у наборі даних, а по-друге повертали позику найчастіше. Тому цю інформацію слід врахувати.

Також слід переглянути інформацію про кількість «public records» у клієнта (рис. 2.15). Ця змінна також включає у себе записи про об’явлене банкрутство.

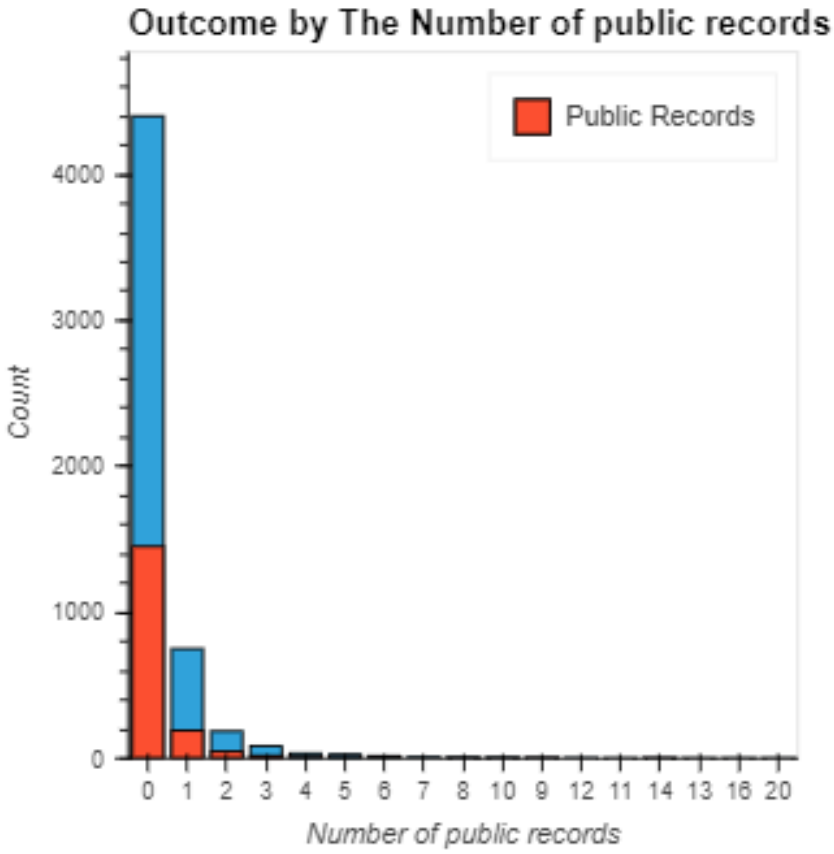


Рис 2.15. Кількість public records до Outcome

Можна побачити (помічено синім), що клієнт про якого є достатньо інформації щодо «public records» частіше повертали позику.

Також розглянемо змінну «Yearly Total Sales» по відношенню до «Outcome» (рис. 2.16).

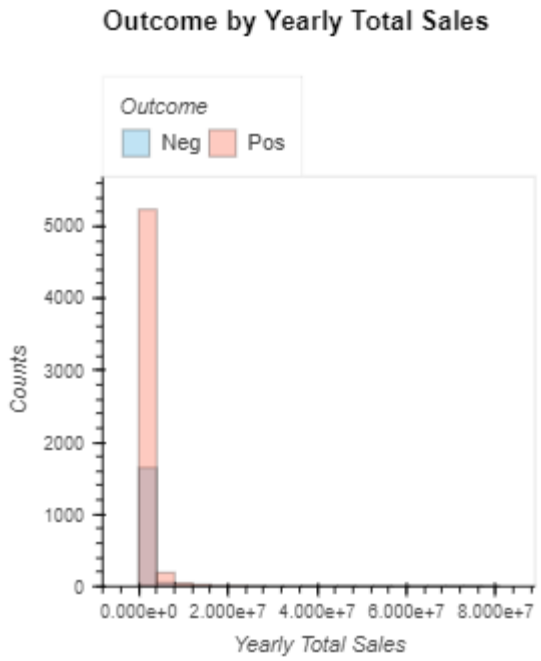


Рис. 2.16. Outcome від Yearly Total Sales

Можна побачити, що позичальники з річним доходом більше або рівний 2000000 грошових одиниць найчастіше повертали бізнес позику. Але, якщо поглянувши на розподіл позичальників з річним доходом менше або рівному 300000 грошових одиниць (рис. 2.17), то можна побачити, що позичальники з доходом у 110000 грошових одиниць не повернули позику. Всього таких позичальників 4.

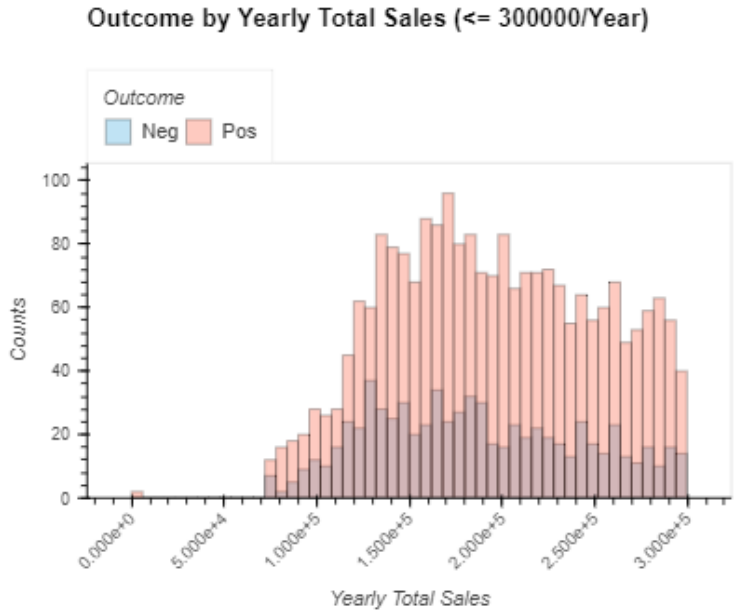


Рис. 2.17. «Outcome» від «Yearly Total Sales» <= 300000

Подивимося на розподіл позичальників за віком (рис. 2.18).

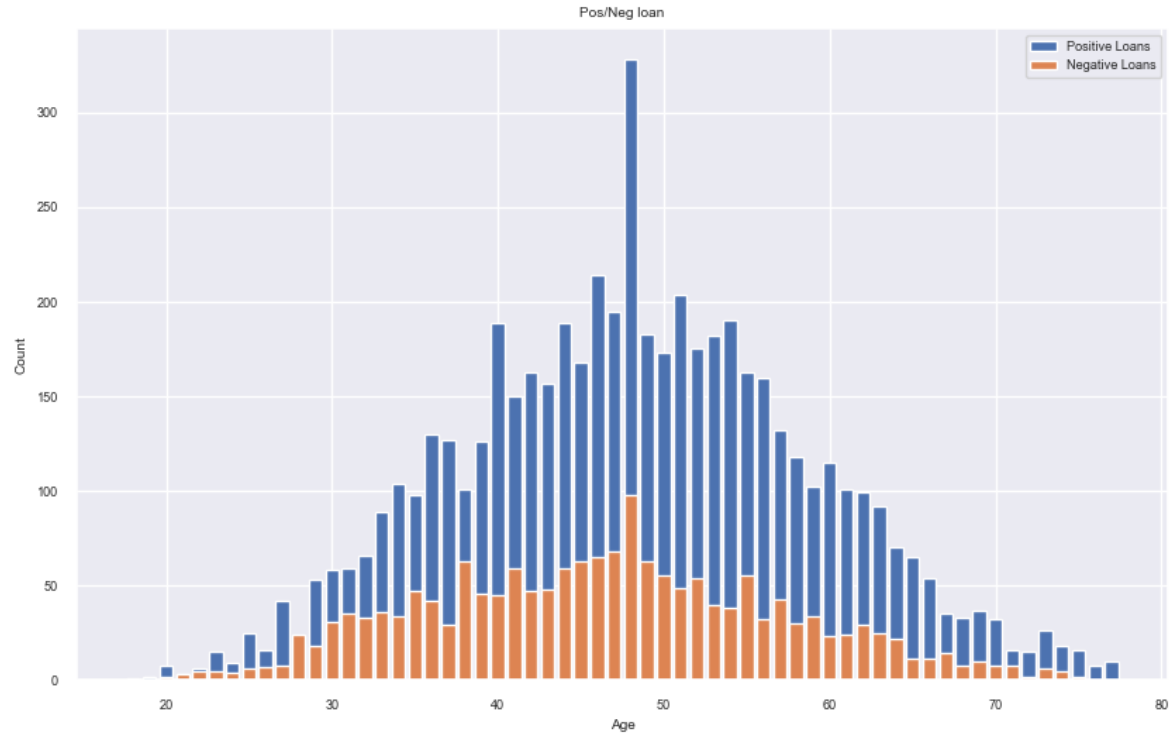


Рис. 2.18. Age від Outcome

Можна зробити висновок про те, що найчастіше повертали позику люди від 47 до 55 років, тобто люди, які вже мають якесь фінансове підґрунтя.

Розглянемо 10 бізнес сфер, які найчастіше не повертали позику (рис. 2.19).

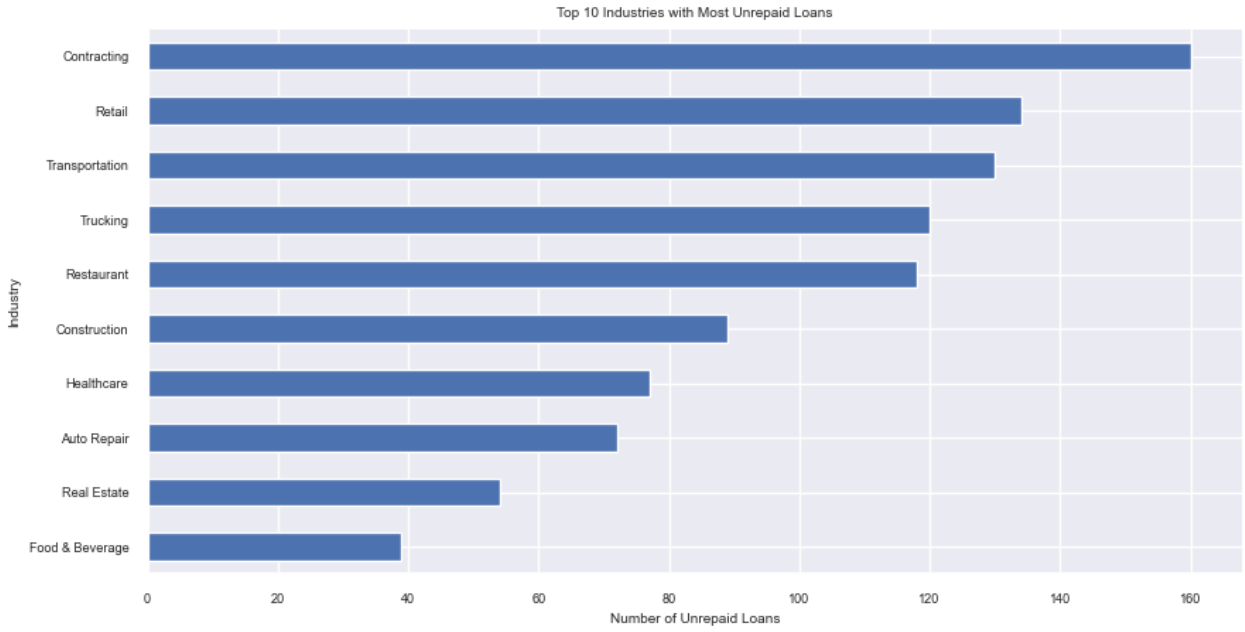


Рис. 2.19. Сфери, що не повертали позику

З усього набору даних найбільше користувачів банку не повертали позику зі ресторанної сфери бізнесу, хоча середній річний заробіток у цій сфері рівний 765241,60 грошовим одиницям.

Перевіримо чи присутня у таблиці інформація про змінні, які мають найбільший рейтинг серед не повернутих до повернутих позик (рис. 2.20).

Outcome	Neg	Pos
Industry		
Bike shops	1	0
Nursing homes	1	0
Environmental	2	1
Not For Profit	0	1
Building Material Suppliers	2	1
Bookkeeping	0	1
Mini Market	1	1
Pubs	1	1
Electronics	4	1
Barbershops	1	1

Рис. 2.20. Повернуті / не повернуті позики за сферою бізнеса

Можна побачити, що позичальники зі сфери «Electronics» найбільше всього не повертали позику, тому ця сфера буде вилучена з розгляду.

Розглянемо зв’язок між двома змінними «Outcome» і «crime_record» для того, щоб зрозуміти зв’язок між ними. Зробимо це за допомогою перехресної таблиці . Перехресна таблиця – це таблиця, яка показує зв'язок між двома або більше змінними.

crime_record	Missing	drug	fraud	traffic
Outcome				
Neg	1155	286	29	262
Pos	3705	920	3	906

Рис. 2.21. Перехресна таблиця «crime_record»

Можна побачити, що майже всі, хто має запис «fraud» не виплачували позику. Тобто користувачів з цим записом можна відкинути з розгляду.

Тепер можна розглянути розподіл даних. Подивитися на частоту значень змінної, а також на частоту, з якою ці значення зустрічаються. Це необхідно для того щоб проаналізувати дані, що зберігаються у змінних, на виявлення аномалій або пропущених значень.

Розглянемо перші 3 графіки розподілів даних (Рис. 2.22), а інші див. додаток Е, рисунки Е.1 – Е.5.

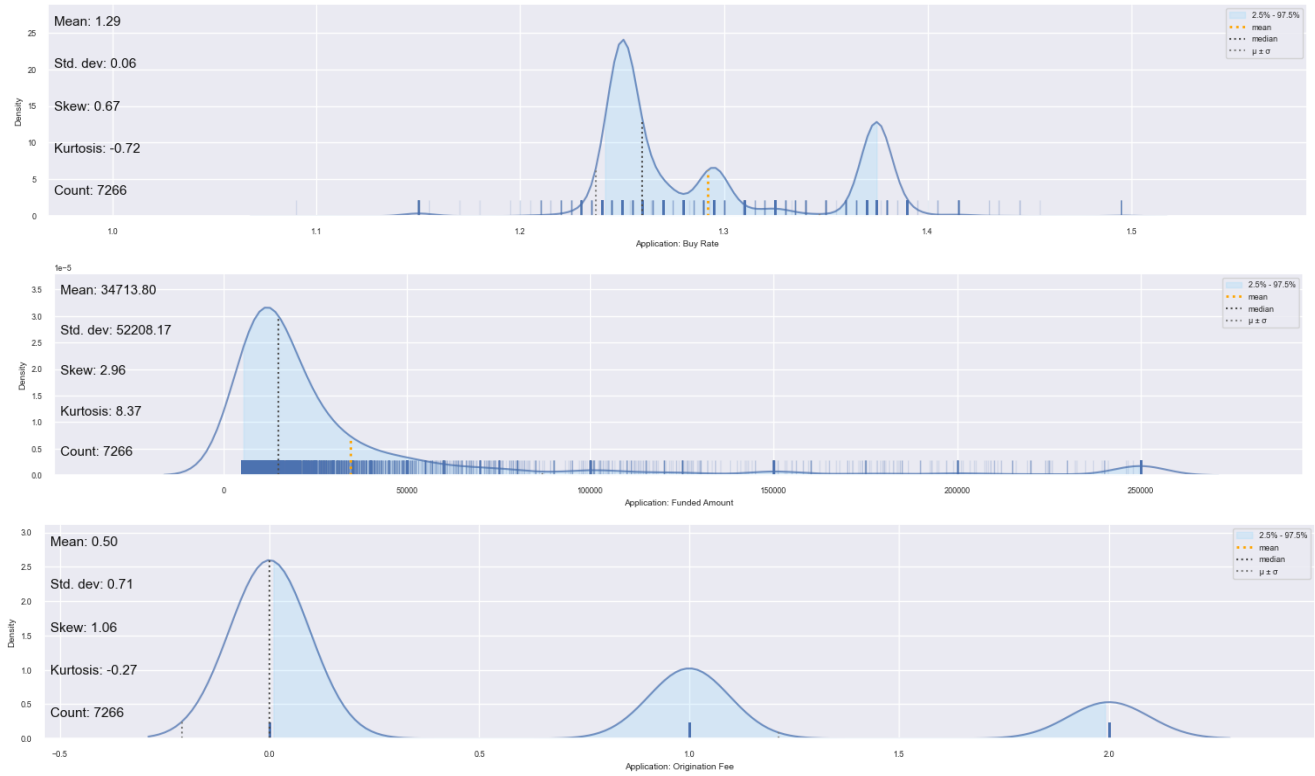


Рис. 2.22. Розподіл даних по стовпцях

З наведених вище рисунків можна зробити висновок про розподіл даних, а саме відносно інформації розподілу щодо асиметрії. У перших двох змінних рисунка 2.22 розподіл розміщується праворуч щодо середнього значення, а у останньої змінної розподіл розміщується трішки ліворуч.

Для розподілу з позитивною асиметрію результат трактується наступним чином: більшість даних зосереджена на менших значеннях. А для розподілу з негативною асиметрію відповідно навпаки.

На рис. 2.23 добре видно, що більшість позик, що було видано, є новими угодами з банком.

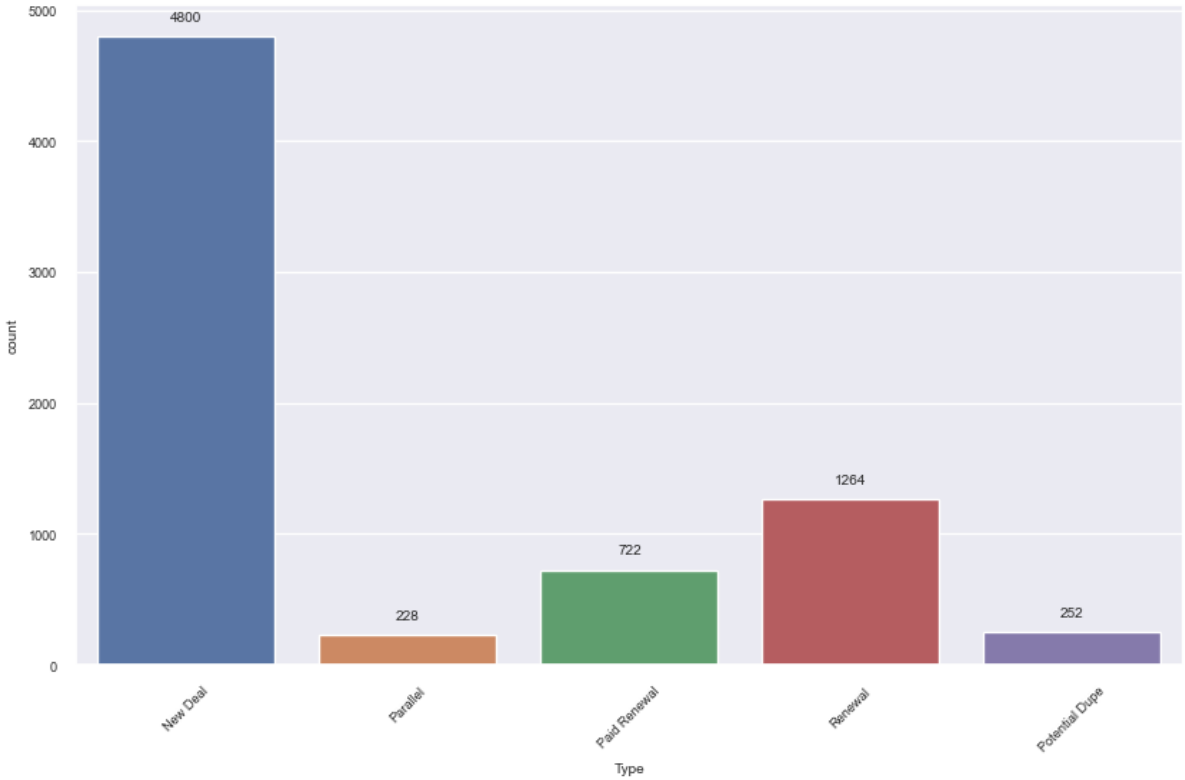


Рис. 2.23. Тип угоди

Далі нас цікавить складова категоріальних змінних (Таблиця 2.3), а саме кількість і склад унікальних змінних, так як у наборі даних представлений стовпець, який містить назву штату позичальника.

Таблиця 2.3

Опис категоріальних змінних

Ім'я змінної	Значення
-1-	**-2-**
Unique values in Application: Remittance Frequency:	2
Unique values of Application: Remittance Frequency:	['Daily' 'Weekly']
Unique values in Primary Contact Gender:	3

Продовження таблиці 2.3

-1-	-2-
Unique values in crime_record:	3
Unique values of crime_record:	['Missing' 'traffic' 'drug']
Unique values in Has Website:	2
Unique values of Has Website:	['Yes' 'No']
Unique values in Industry:	113
Unique values in Office Space:	2
Unique values of Office Space:	['Rent' 'Own']
Unique values in Position:	4
Unique values of Position:	['First' 'Second' 'Third' 'Fourth']
Unique values in Shipping State:	53
Unique values in Type:	5
Unique values of Type:	['New Deal' 'Parallel' 'Paid Renewal' 'Renewal' 'Potential Dupe']
Unique values in Outcome:	2
Unique values of Outcome:	['Neg' 'Pos']

Кількість унікальних назв у змінній «Shipping State» складає 53, а наразі у США всього 50 штатів. Відповідно 3 штати, що зберігається у цій змінній не є валідними. Дописи, які містять ці 3 неіснуючі штати, будуть відкинути з розгляду.

Розглянемо скрипкові діаграми змінних з набору даних (рис. 2.24).

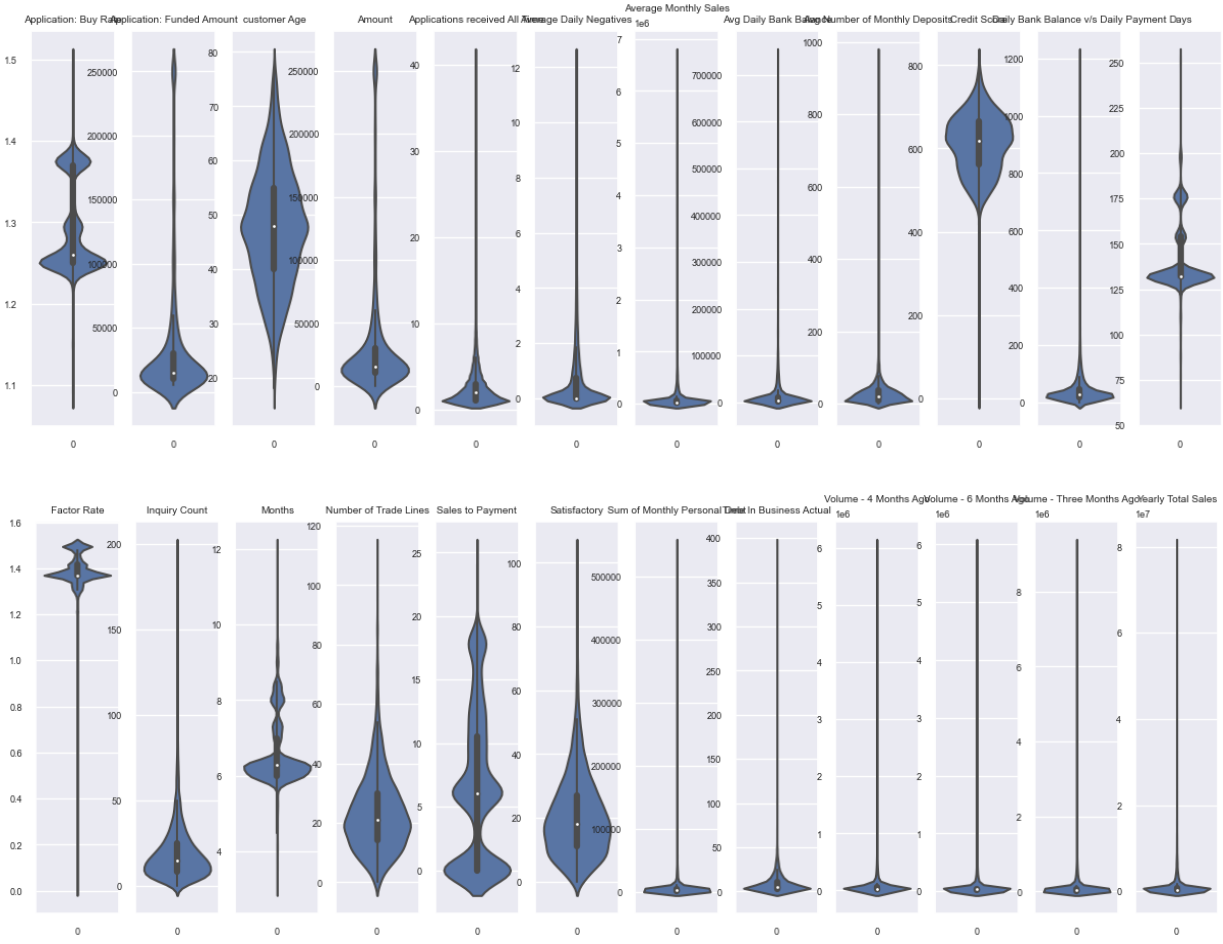


Рис. 2.24. Скрипкові діаграми змінних

Можна спостерігати тенденцію у наведених вище діаграмах щодо важків хвостів скрипкових діаграм – тобто, коли у даних є значення, які відхиляються від середнього більше від нормального розподілу.

Тепер подивимося на саму діаграму ядерного розподілу (Рис. 2.25).

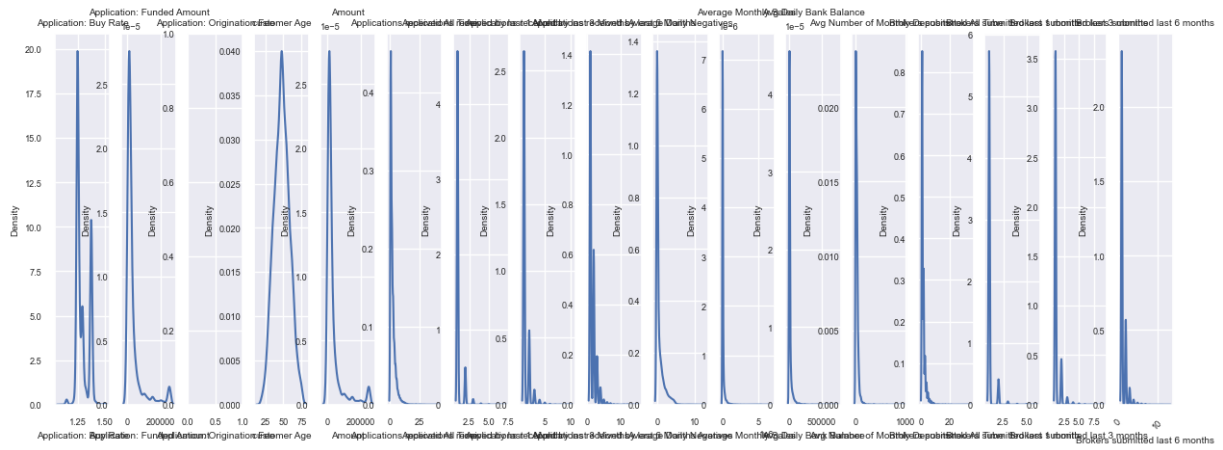
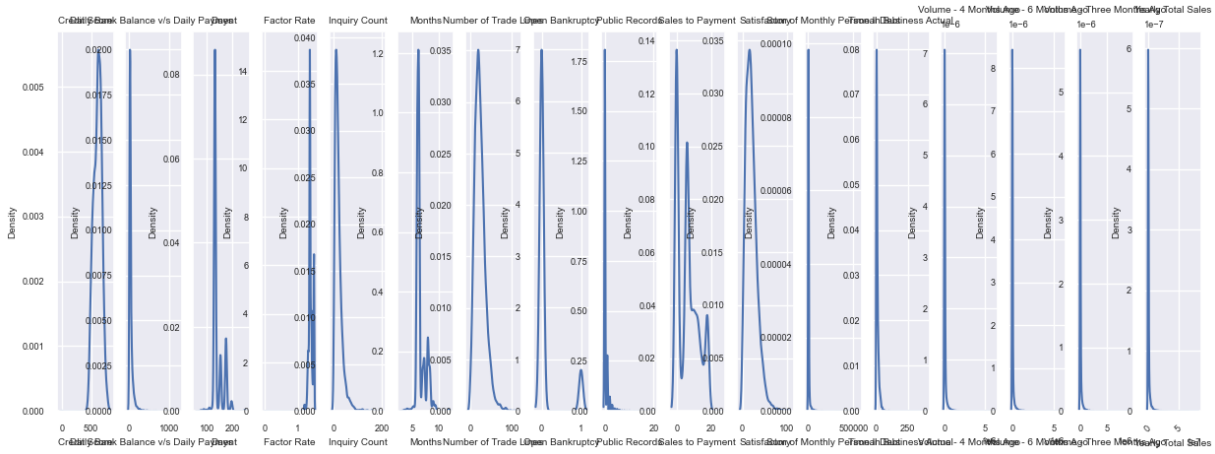


Рис. 2.25. Оцінка щільності ядра



Так як, скрипкова діаграма це поєднання ящика з вусам (boxplot) і оцінки щільності ядра (KDE), то відповідно і на рис. 2.25, аркуш 60, що наведено вище, можна побачити аналогічну поведінку розподілів змінних, але на рис. 2.25, аркуш 60 значно краще видно форму розподілу одновимірних даних.

Наступним кроком на даному етапі є позбавлення від мультиколінеарності.

Під відкидання попадають такі змінні: «Yearly Total Sales», «Months», «Factor Rate», «Amount», «Application: Buy Rate». Тоді коефіцієнти VIF мають наступні значення (Рис. 2.26)

	VIF Factor	features
2	940.165403	Application: Origination Fee
23	14.746299	Satisfactory
19	14.439096	Number of Trade Lines
14	8.961916	Brokers submitted last 6 months
7	8.577889	Applications received by last 6 Months
6	8.527877	Applications received by last 3 Months
13	8.406223	Brokers submitted last 3 months
9	7.077615	Average Monthly Sales
12	6.298937	Brokers submitted All Time
4	6.238587	Applications received All Time

Рис. 2.26. Коефіцієнт VIF після видалення комірок

Змінна «Application: Origination Fee» буде входити в кластер, який буде створено стосовно даних про заявку позики, а потім видалено.

При кодуванні змінних буде використовуватися Label Encoding.

Застосуємо Fuzzy Claster для додавання міток кластерів як нових ознак до набору даних, що розглядається. Fuzzy Claster використовується саме через те, що він дозволяє кожному об'єкту в набору даних належати до кожного кластера з певним ступенем належності, а не просто присвоювати кожен об'єкт до одного конкретного кластера, і тому це може поліпшити точність прогнозування. Розмита кластеризація може допомогти моделі краще впоратися з такою неоднозначністю, забезпечуючи більш ніжну і гнучку модель.

Для візуалізації кластерів (Рис. 2.27) використовується алгоритм t-SNE (t-Distributed Stochastic Neighbor Embedding). Даний алгоритм зменшує розмірність даних для їх відображення.

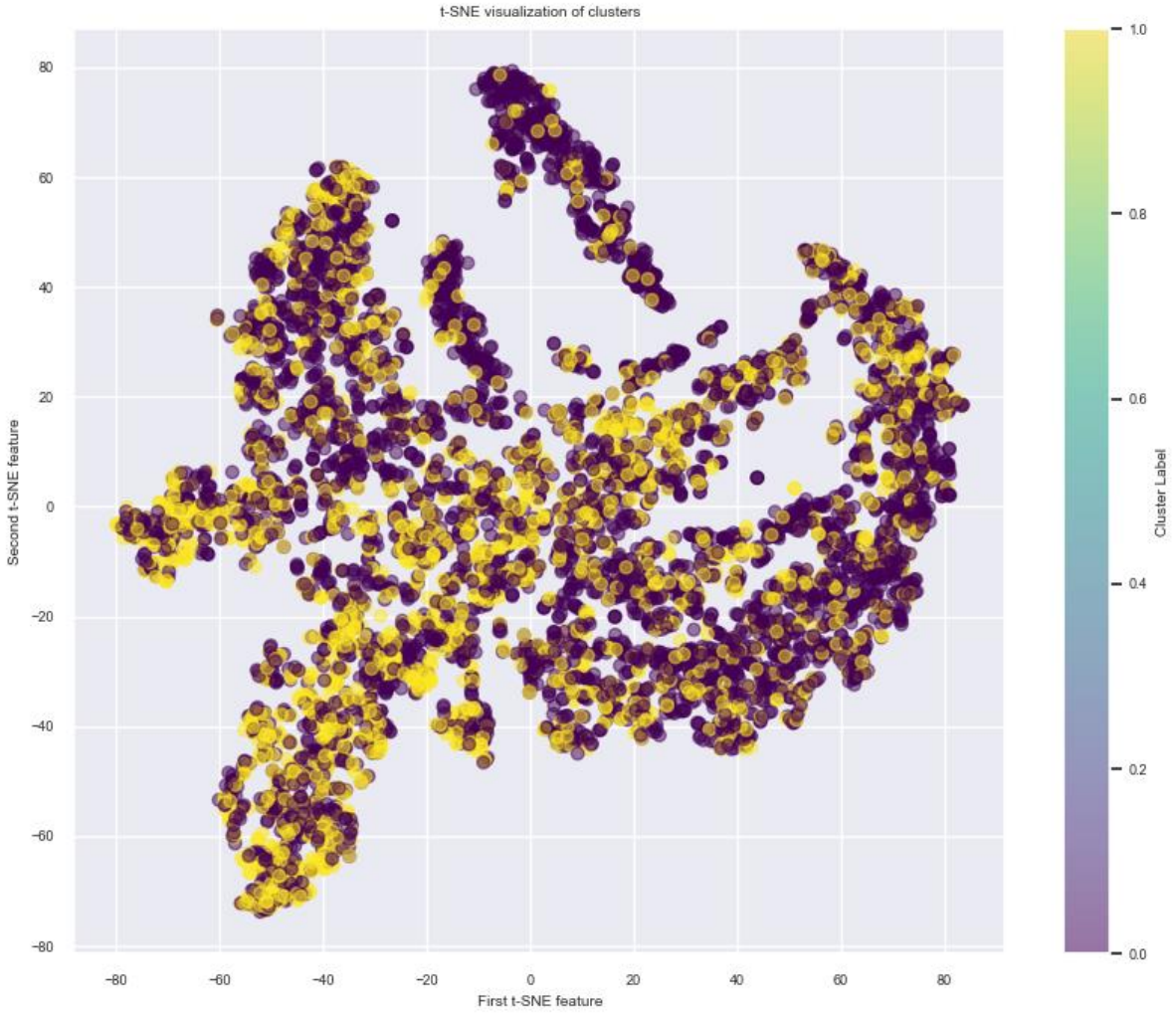


Рис. 2.27. Візуалізація кластерів даних

Знайдемо оптимальну кількість кластерів. Для цього зробимо декілька перетворень. Перетворимо ступені належності на жорсткі приналежності, виберемо кластер з найвищим ступенем належності для кожної точки, а потім застосуємо метод ліктя (Рис. 2.28) або силуетний аналіз до цих жорстких приналежностей.

На рисунку 2.28 зображено пошук оптимального числа кластерів. Можна спостерігати, що на рисунку 2.28 зображена спадаюча пряма. Це свідчить про те, що додавання більшої кількості кластерів не принесе значного поліпшення у внутрішньокластерній схожості даних.

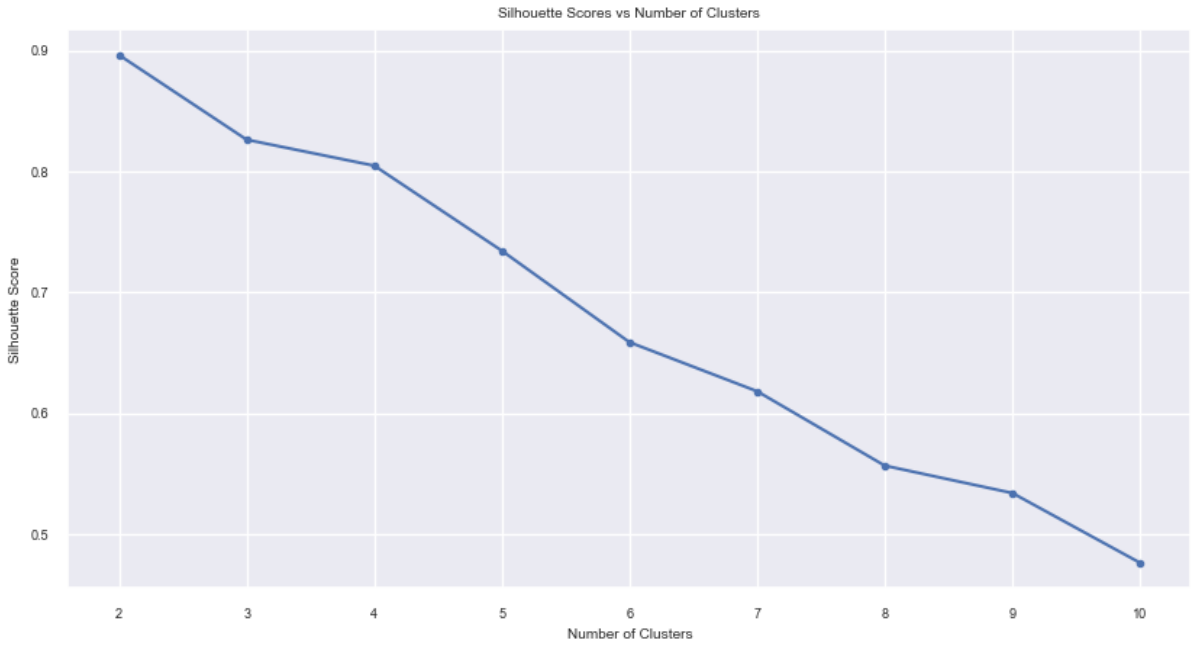


Рис 2.28. Оптимальне число кластерів

2.5 Підтверджувальний аналіз даних (CDA)

Після проведення дослідницького аналізу даних, наступним кроком є підтверджувальний аналіз даних (CDA), який часто включає моделювання для прогнозування. У цьому підрозділі буде використовуватися модель з XGBoost, високоефективною реалізацією алгоритму градієнтного бустінгу, який використовує дерева рішень як базові моделі.

Окрім цього, було застовано Fuzzy Clustering для створення нових ознак, які допомагають врахувати невизначеність та посилити модель. Fuzzy Clustering. це метод групування, який дозволяє кожному об'єкту належати до кожного кластера з певною ступеню належності, що відрізняється від традиційних методів групування, таких як k-means, де кожен об'єкт належить лише до одного кластера.

Щоб оптимізувати модель XGBoost, буде використовуватися Optuna для підбору гіперпараметрів. Optuna. це фреймворк для оптимізації гіперпараметрів, який включає ефективні алгоритми, такі як TPE (Tree-structured Parzen Estimator), і дозволяє здійснювати автоматичний підбір гіперпараметрів. Це надає перевагу над традиційними методами решітчатого пошуку або випадкового пошуку, що зазвичай вимагають більше часу та ресурсів.

Щодо метрик оцінювання, будуть використовуватися accuracy_score, precision_score, recall_score, f1_score та geometric_mean_score з бібліотеки sklearn.

- Accuracy (точність). це відношення правильно передбачених спостережень до загальної кількості спостережень.
- Precision (точність). це відношення правильно передбачених позитивних спостережень до загальної кількості передбачених позитивних спостережень.
- Recall (повнота). це відношення правильно передбачених позитивних спостережень до всіх дійсних позитивних спостережень.
- F1 Score. це середнє гармонічне між Precision та Recall.
- Geometric Mean Score. це корінь квадратний з добутку всіх значень метрики, що є більш рівноважною метрикою для несбалансованих наборів даних.

Також буде використовуватися ConfusionMatrixDisplay для візуалізації результатів. ROC-крива (Receiver Operating Characteristic). це графік, який відображає співвідношення між чутливістю моделі (True Positive Rate) та специфічністю (1. False Positive Rate) при різних порогових значеннях. AUC-ROC (Area Under the ROC Curve). це числова характеристика якості бінарної класифікації, що розраховується як площа під ROC-кривою.

Матриця помилок, або confusion matrix,. це таблиця, яка дозволяє візуально оцінити роботу алгоритму класифікації. З її допомогою можна ідентифікувати, які класи алгоритм плутає.

2.5.1 Прогнозування бізнес позик на незбалансованому наборі даних, використовуючи користувацькі функції втрат

Для початку розглянемо модель, яка була навчена на незбалансованому наборі даних з використанням Focal Loss.

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

```
params = {
  'objective': focal_loss,
  'n_estimators': 102,
  'max_depth': 10,
  'learning_rate': 0.0011966131925523543,
  'subsample': 0.8006227968027733,
  'colsample_bytree': 0.9803398728188094,
  'gamma': 0.8386194241845784,
  'min_child_weight': 3}
```

де params – оптимальні підібрані параметри.

Після навчання моделі було отримано було отримано наступний словник з результатами класифікації.

```
{'Accuracy Score': 0.6373626373626373,
 'Precision Score': 0.3333333333333333,
 'Recall Score': 0.0038314176245210726,
 'F1 Score': 0.007575757575757576,
 'G-mean': 0.06148020852560959}
```

А також було побудовано графік візуалізації матриці помилок (Рис. 2.30) і ROC-кривої (Рис. 2.31) за допомогою класу «ConfusionMatrixDisplay» з бібліотеки sklearn і кастомної функції для графіка ROC-кривої. Перший клас графічно відображає матрицю помилок.

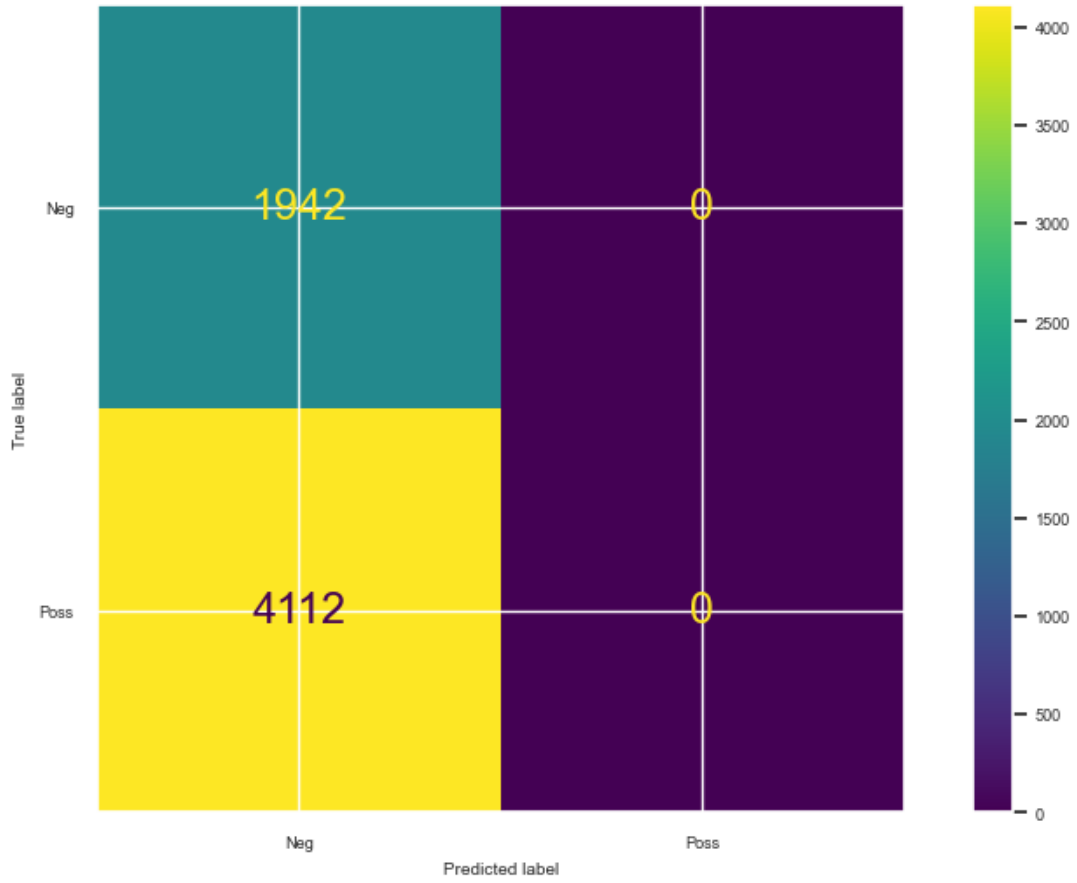


Рис. 2.30. Матриця помилок незбалансованого набору даних

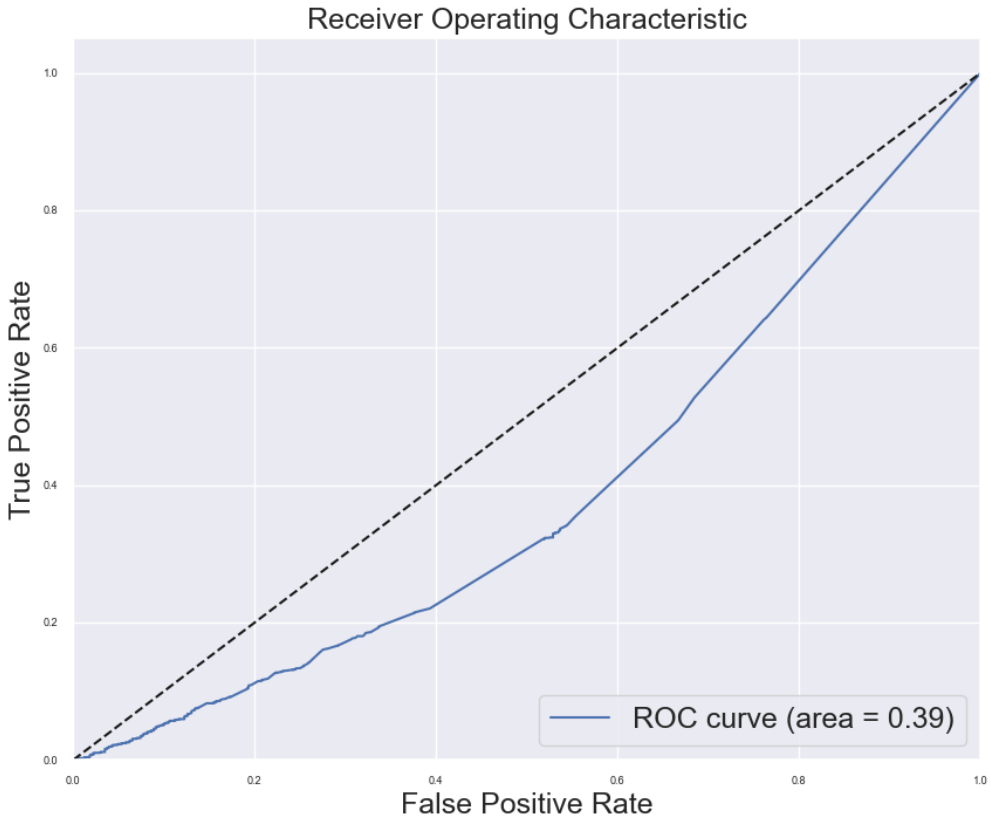


Рис. 2.31. Графік ROC-кривої незбалансованого набору даних

Наступною користувацькою функцією втрат, яка буде розглядатися є Weighted Cross Entropy. Параметри для моделі не змінюються.

Тому отримуємо такі розраховані метрики:

{'Accuracy Score': 0.6373626373626373,

'Precision Score': 0.6411740388590327,

'Recall Score': 0.9904214559386973,

'F1 Score': 0.7784190715181932,

'G-mean': 0.15989507462567792}

Графіки ROC-кривої і матриці помилок можна переглянути у додатку Ж, Рис. 1. Ж.

2.5.1.1 Аналіз результатів

Accuracy Score для обох моделей точність складає 0.637, тобто приблизно 64% передбачень моделі є вірними.

Precision Score у моделі з користувацькою функцією Focal Loss ця метрика дорівнює 0,33, що модель має обмежену здатність правильно класифікувати позитивні приклади. У моделі з Weighted Cross Entropy ця метрика дорівнює 0.64, свідчить про помірну здатність моделі правильно класифікувати позитивні приклади.

Recall Score у моделі з Focal Loss моделі ця метрика дорівнює 0, що означає, що модель не змогла правильно передбачити жодного позитивного випадку. У моделі з Weighted Cross Entropy ця метрика дорівнює 1, що означає, що модель правильно передбачила всі позитивні випадки.

F1 Score у моделі з Focal Loss моделі ця метрика дорівнює 0, що відображає погану продуктивність моделі. У Weighted Cross Entropy моделі ця метрика дорівнює 0.81, що відображає добру продуктивність моделі.

G-mean у Focal Loss ця метрика дорівнює 0.0614 показує, що модель має дуже низький рівень згоди між цими двома метриками.

На основі цих метрик можна зробити висновок, що модель з Weighted Cross Entropy виявилася кращою, ніж модель з Focal Loss, оскільки вона має вищі показники точності, повернення та F1-оцінки. Проте обидві моделі виявилися слабкими у вирішенні проблеми незбалансованості класів, що відображається в нульовому значенні G-mean.

2.5.2 Застосування методів балансування даних

Будуть застосовані наступні методи для балансування даних, які було розглянуто у інформаційно-аналітичному розділі: Random Under Sampling, Random Over Sampling, SMOTE, ADASYN, SMOTETOMEK, SMOTEENN.

1. Random Under Sampling

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

```
params = {
   'n_estimators': 221,
    'max_depth': 10,
    'learning_rate': 0.00041821413923095774,
    'subsample': 0.8742097017145511,
    'colsample_bytree': 0.5936492805139413,
    'gamma': 0.9463373617864398,
    'min_child_weight': 7}
```

Після навчання моделі було отримано було отримано наступний словник з результатами класифікації.

```
{'Accuracy Score': 0.6373626373626373,
 'Precision Score': 0.7463556851311953,
```

'Recall Score': 0.4904214559386973,

'F1 Score': 0.5919075144508671,

'G-mean': 0.5888655581821884}

А також було побудовано графіки візуалізації матриці помилок (Рис. 2.32) і ROC-кривої (Рис. 2.33).

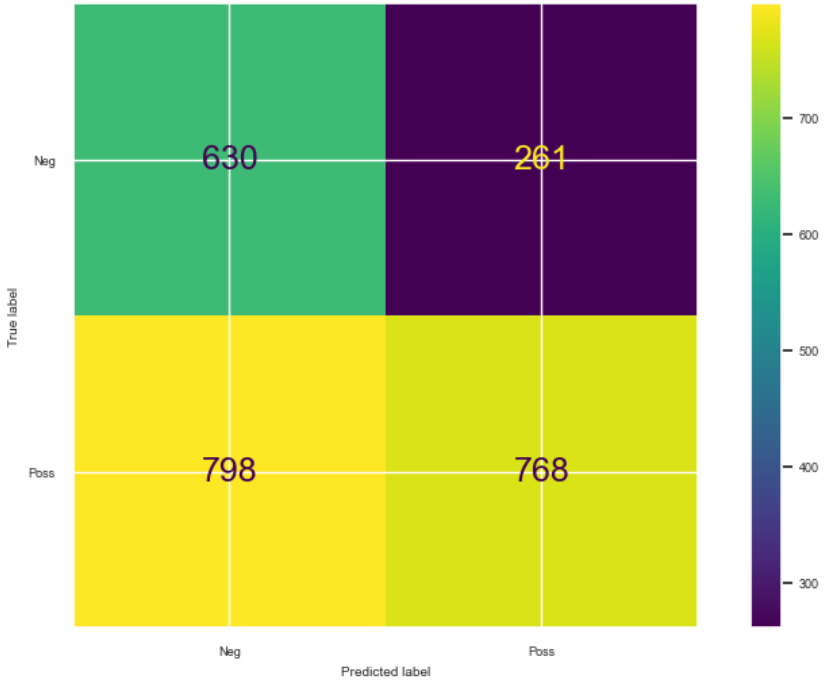


Рис. 2.32. Матриця помилок збалансованого набору даних за допомогою RUS

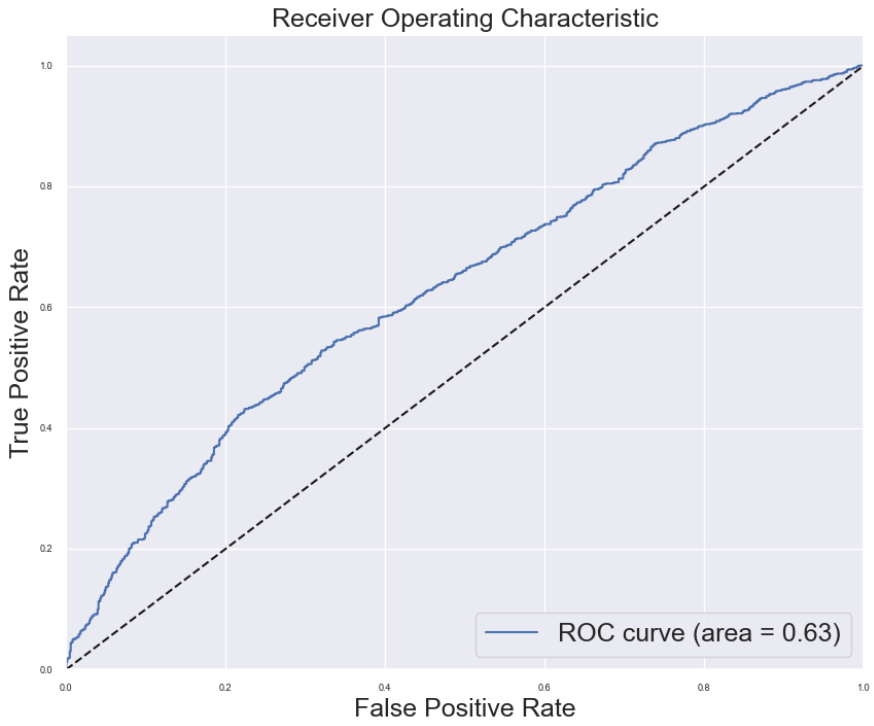


Рис. 2.33. Графік ROC-кривої збалансованого набору даних за допомогою RUS

2. Random Over Sampling

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

params = {

'n_estimators': 258,

'max_depth': 10,

'learning_rate': 0.07331865190064507,

'subsample': 0.7678677499159607,

'colsample_bytree': 0.596916820568302,

'gamma': 0.3378097005569924,

'min_child_weight': 3}

Після навчання моделі було отримано було отримано наступний словник з результатами класифікації.

{'Accuracy Score': 0.6487586487586487,

'Precision Score': 0.6747285291214216,

'Recall Score': 0.8729246487867177,

'F1 Score': 0.7611358574610245,

'G-mean': 0.4767530925375613}

А також було побудовано графіки візуалізації матриці помилок (Рис. 2.34) і ROC-кривої (Рис. 2.35).

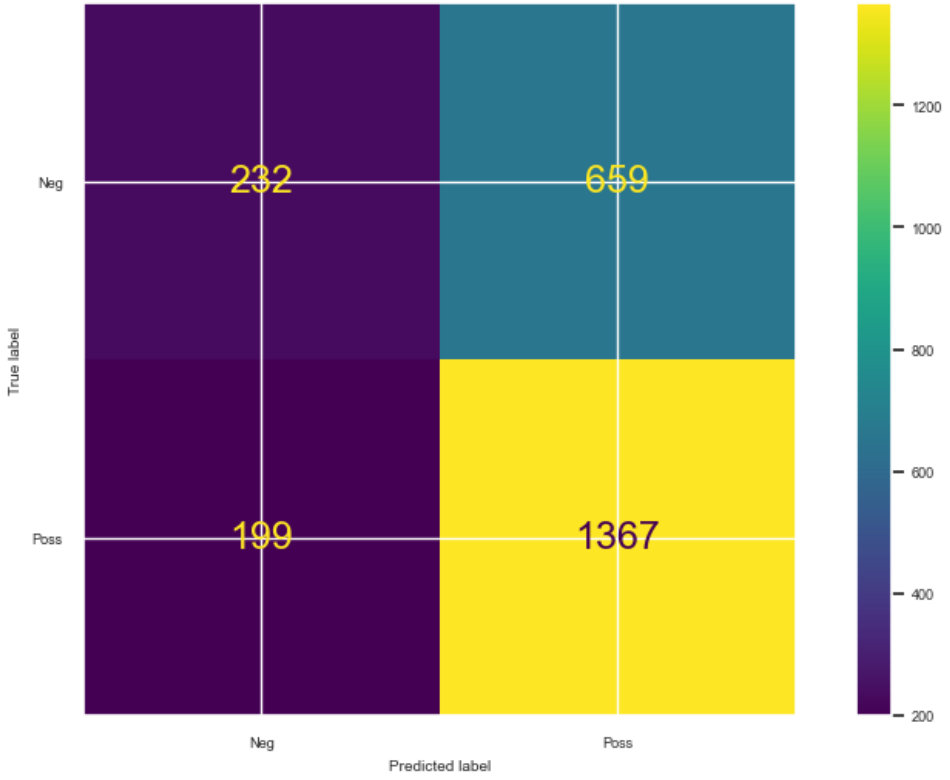


Рис. 2.34. Матриця помилок збалансованого набору даних за допомогою ROS

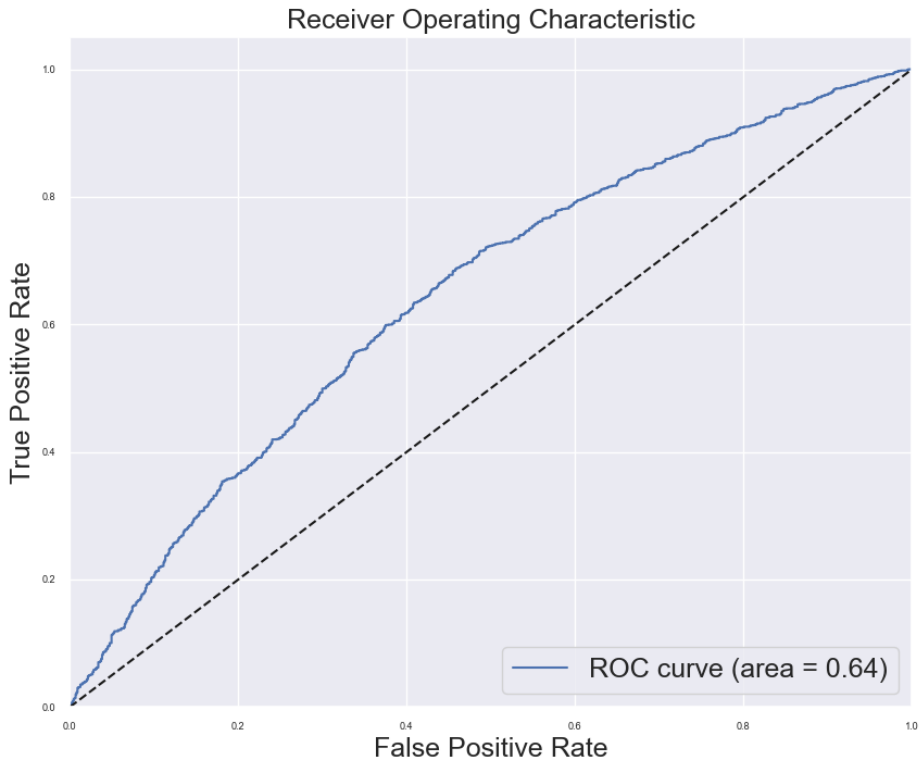


Рис. 2.35. Графік ROC-кривої збалансованого набору даних за допомогою ROS

3. SMOTE

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

params = {

'n_estimators': 143,

'max_depth': 10,

'learning_rate': 0.0012381122229706693,

'subsample': 0.8764884885259474,

'colsample_bytree': 0.5122076493704834,

'gamma': 0.42260708044796647,

'min_child_weight': 6}

Після навчання моделі було отримано було отримано наступний словник з результатами класифікації.

{'Accuracy Score': 0.6373626373626373,

'Precision Score': 0.6601178781925344,

'Recall Score': 0.8582375478927203,

'F1 Score': 0.7462520821765687,

'G-mean': 0.4378158113389911}

А також було побудовано графіки візуалізації матриці помилок (Рис. 2.36) і ROC-кривої (Рис. 2.37).

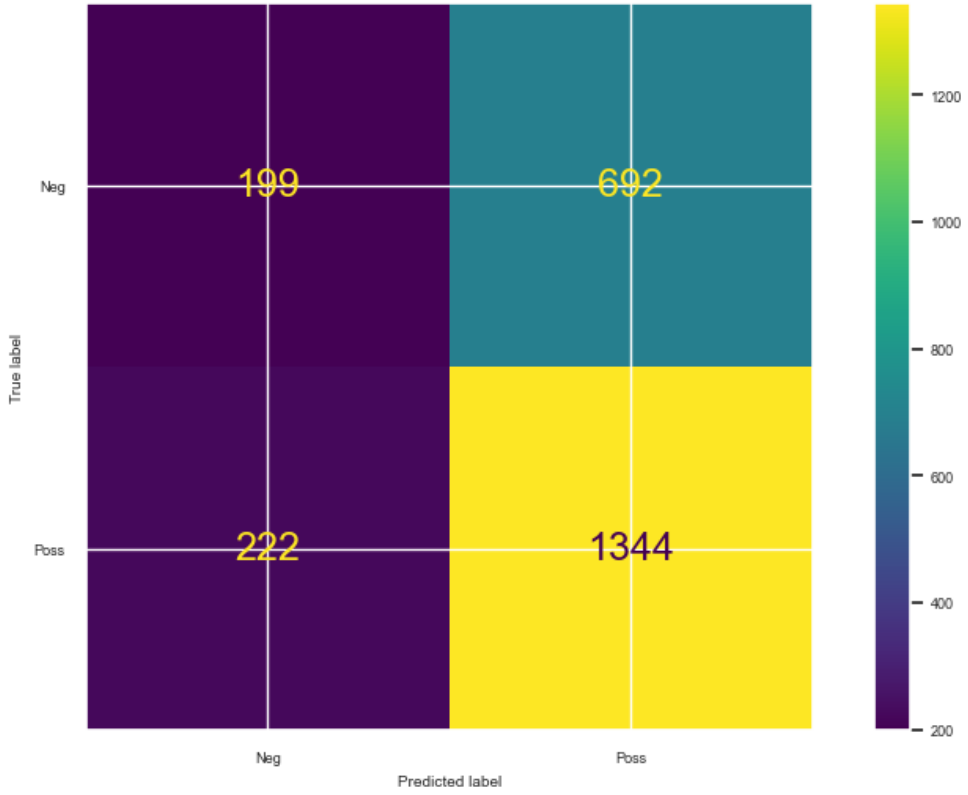


Рис. 2.36. Матриця помилок збалансованого набору даних за допомогою SMOTE

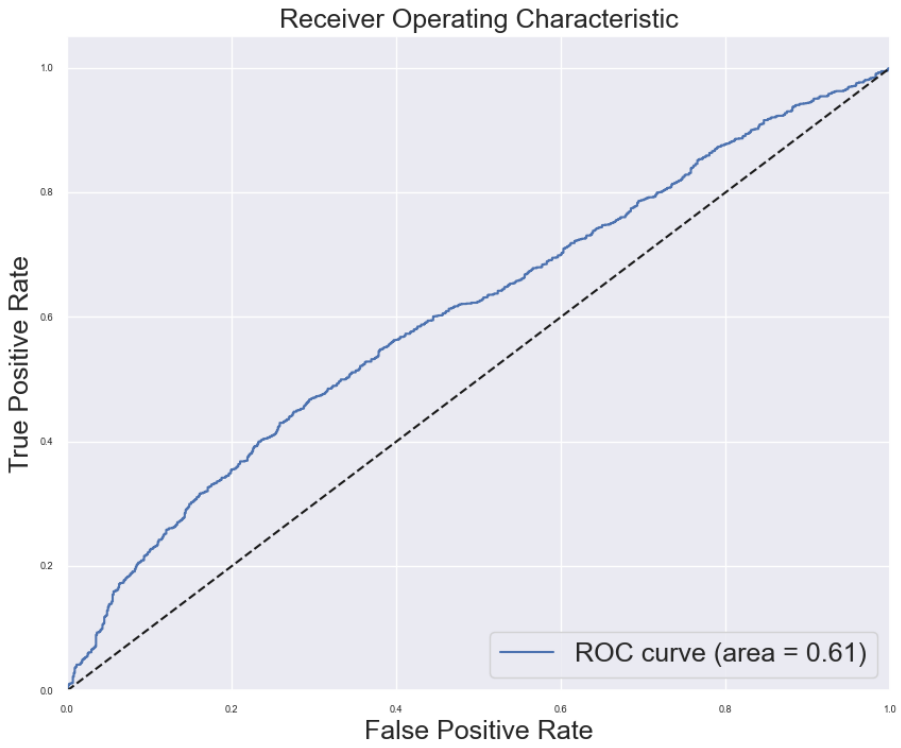


Рис. 2.37. Графік ROC-кривої збалансованого набору даних за допомогою SMOTE

4. ADASYN

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

params = {

'n_estimators': 190,

'max_depth': 7,

'learning_rate': 0.13836053931252873,

'subsample': 0.6016105058143335,

'colsample_bytree': 0.8645099541402509,

'gamma': 0.7162834664094448,

'min_child_weight': 3}

Після навчання моделі було отримано було отримано наступний словник з результатами класифікації.

{'Accuracy Score': 0.6398046398046398,

'Precision Score': 0.6873508353221957,

'Recall Score': 0.735632183908046,

'F1 Score': 0.7106724244293645,

'G-mean': 0.5504584473311485}

А також було побудовано графіки візуалізації матриці помилок (Рис. 2.38) і ROC-кривої (Рис. 2.39).

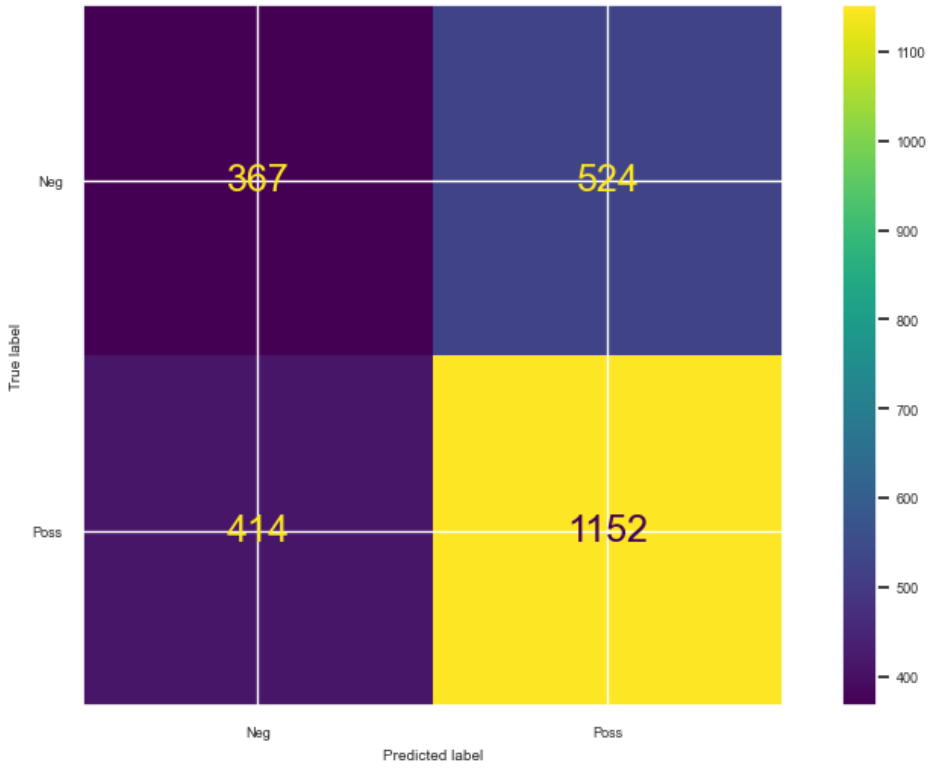


Рис. 2.38. Матриця помилок збалансованого набору даних за допомогою ADASYN

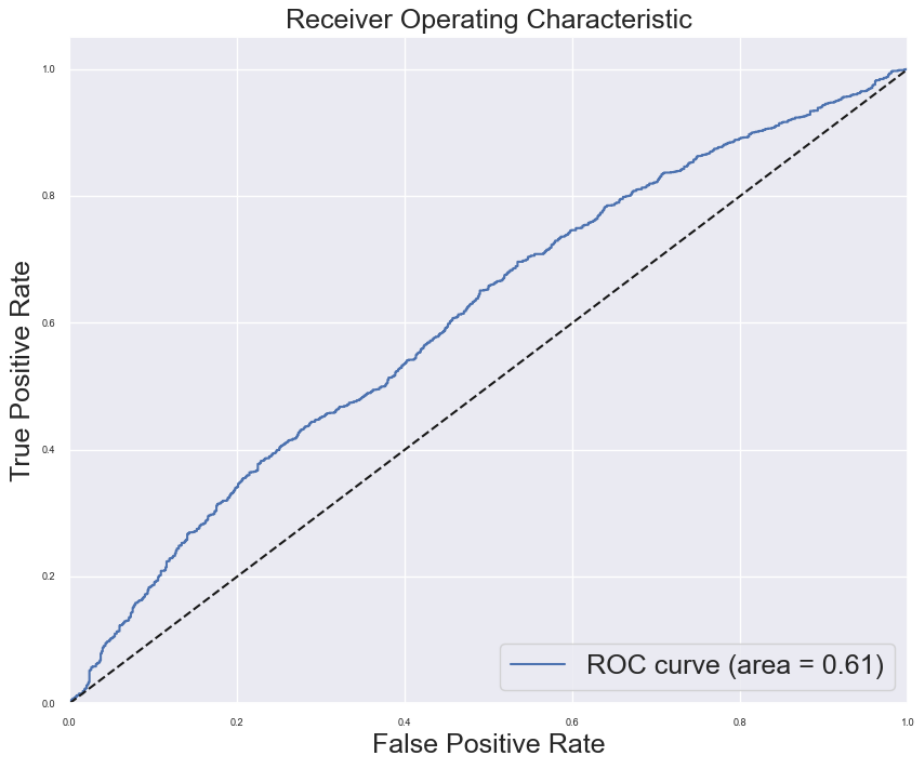


Рис. 2.39. Графік ROC-кривої збалансованого набору даних за допомогою ADASYN

5. SMOTETOMEK

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

params = {

'max_depth': 10,

'n_estimators': 150,

'learning_rate': 0.5,

'subsample': 0.8,

'colsample_bytree': 0.9,

'seed': 12}

Після навчання моделі було отримано було отримано наступний словник з результатами класифікації.

{'Accuracy Score': 0.6398046398046398,

'Precision Score': 0.6873508353221957,

'Recall Score': 0.735632183908046,

'F1 Score': 0.7106724244293645,

'G-mean': 0.5504584473311485}

А також було побудовано графіки візуалізації матриці помилок (Рис. 2.40) і ROC-кривої (Рис. 2.41).

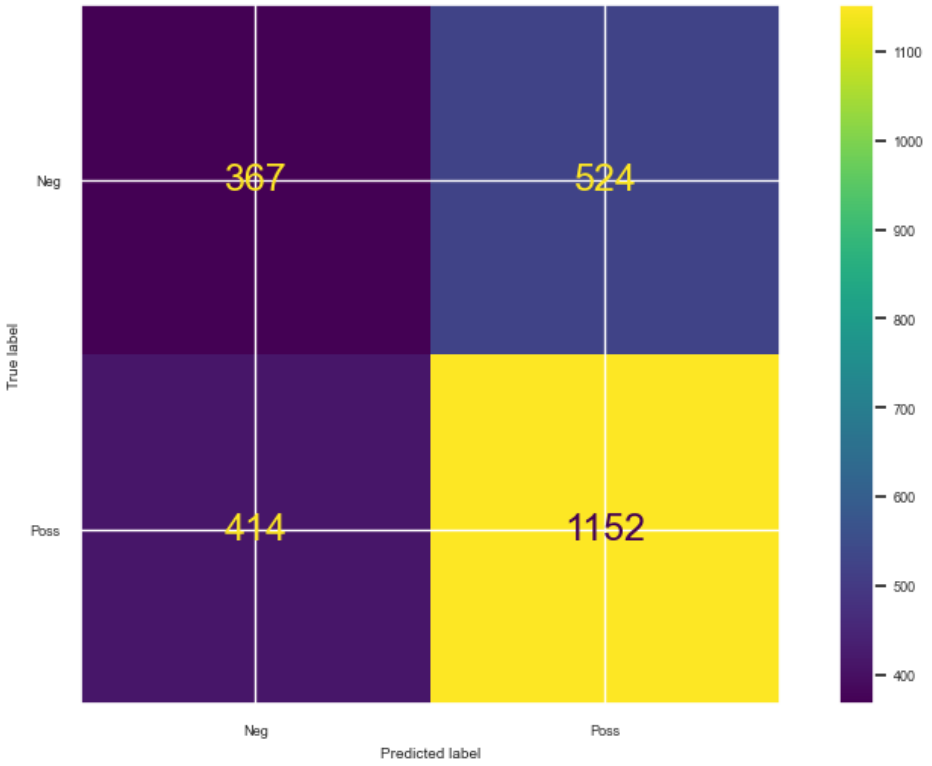


Рис. 2.40. Матриця помилок збалансованого набору даних за допомогою SMOTETOMEK

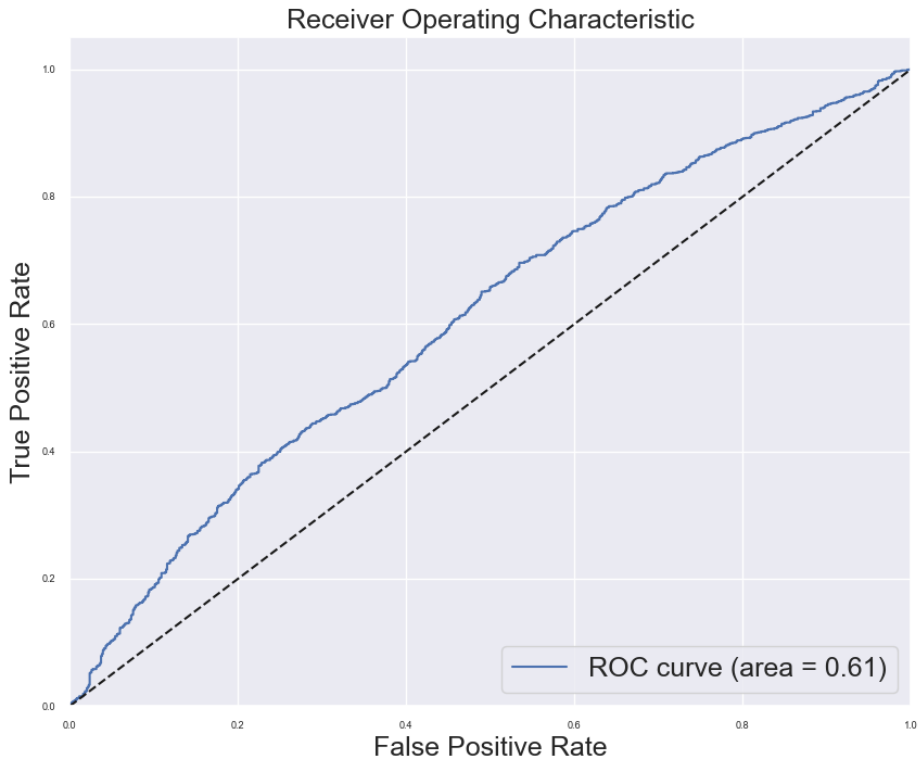


Рис. 2.41. Графік ROC-кривої збалансованого набору даних за допомогою SMOTETOMEK

6. SMOTEENN

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

params = {

'n_estimators': 168,

'max_depth': 6,

'learning_rate': 0.08509848665588302,

'subsample': 0.9239135350691549,

'colsample_bytree': 0.9043052520550187,

'gamma': 0.1889174325488193,

'min_child_weight': 1}

Після навчання моделі було отримано було отримано наступний словник з результатами класифікації.

{'Accuracy Score': 0.6398046398046398,

'Precision Score': 0.6873508353221957,

'Recall Score': 0.735632183908046,

'F1 Score': 0.7106724244293645,

'G-mean': 0.5504584473311485}

А також було побудовано графіки візуалізації матриці помилок (Рис. 2.42) і ROC-кривої (Рис. 2.43).

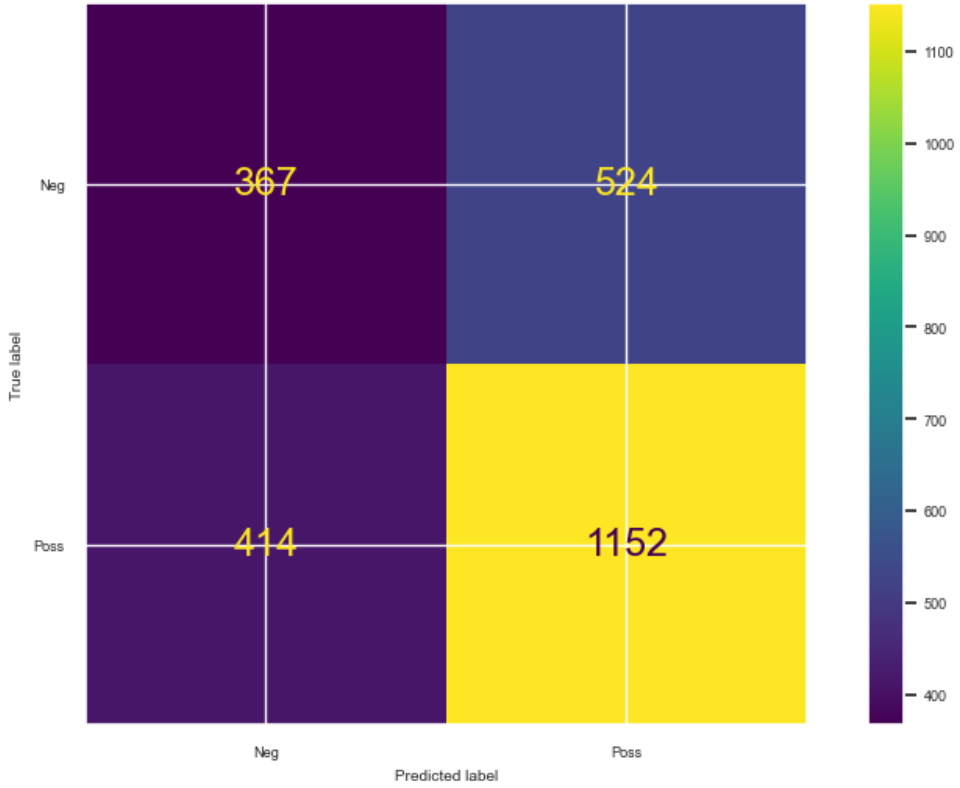


Рис. 2.42. Матриця помилок збалансованого набору даних за допомогою SMOTEENN

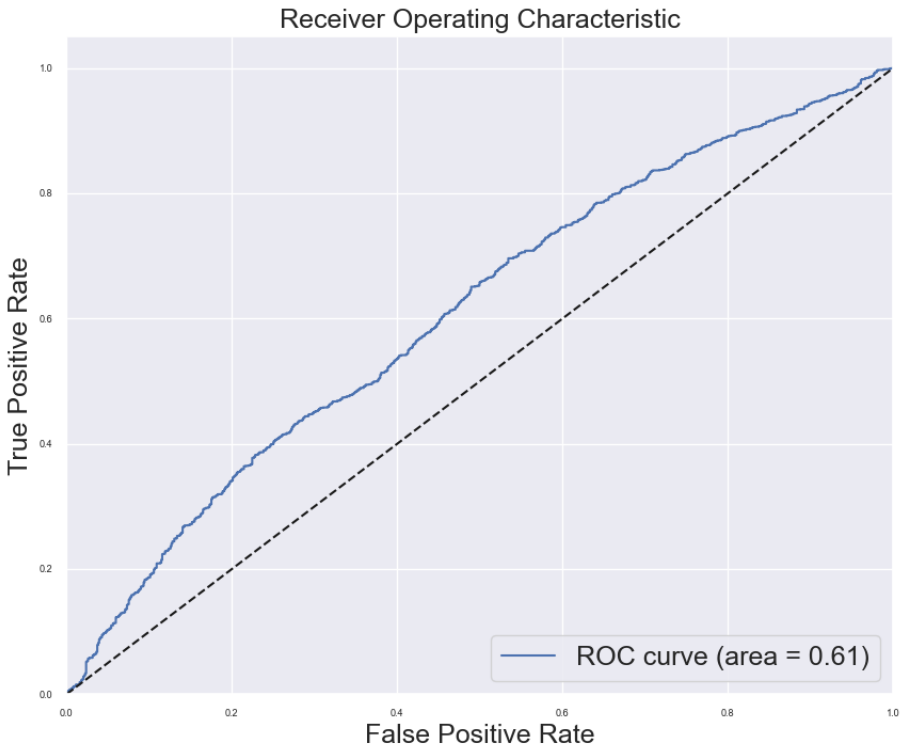


Рис. 2.43. Графік ROC-кривої збалансованого набору даних за допомогою SMOTEENN

2.5.2.1 Аналіз результатів

Значення обрахованих метрик зведено у таблицю 2.4.

Таблиця 2.4

Значення метрик по кожному з методів балансування

	RUS	ROS	SMOTE	ADASYN	SMOTETOMEK	SMOTEENN
-1-	**-2-**	**-3-**	**-4-**	**-5-**	**-6-**	**-7-**
Accuracy Score	0.637363	0.648759	0.637363	0.639805	0.630851	0.631665
Precision Score	0.746356	0.674729	0.660118	0.687351	0.676383	0.704663
Recall Score	0.490421	0.872925	0.858238	0.735632	0.726054	0.607918

Продовження таблиці 2.4

-1-	-2-	-3-	-4-	-5-	-6-	-7-
G-mean	0.588866	0.476753	0.437816	0.550458	0.531753	0.579384

На основі отриманих метрик можна зробити наступний висновок:

Метод Random Over-sampling (ROS) має найвище значення для чутливості (Recall Score), що означає його здатність ефективно виявляти позитивні приклади. Однак, він також має найнижчу точність (Precision Score), що може вказувати на більшу кількість помилкових позитивних класифікацій.

Методи ADASYN і SMOTE також показують гарні результати, з вищими значеннями як для чутливості, так і для точності в порівнянні з RUS і SMOTEENN.

Методи SMOTETOMEK і SMOTEENN мають середні результати, з проміжними значеннями для всіх метрик.

З огляду на отримані значення метрик, найкращим методом балансування даних є ROS. Цей метод демонструє високу точність (Precision Score), чутливість (Recall Score) та F1 Score, що означає його здатність ефективно балансувати класи і досягати якісних результатів класифікації.

2.5.3 Комбінація застосування методів балансування і користувацької функції втрат

1. Random Under Sampling + (Focal Loss / Weighted Cross Entropy)

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

```
params_focal = {
   'objective': focal_loss,
   'n_estimators': 221,
```

```
    'max_depth': 10,
    'learning_rate': 0.00041821413923095774,
    'subsample': 0.8742097017145511,
    'colsample_bytree': 0.5936492805139413,
    'gamma': 0.9463373617864398,
    'min_child_weight': 7}
params_weighted = {
    'objective': weighted_binary_cross_entropy,
    'n_estimators': 221,
    'max_depth': 10,
    'learning_rate': 0.00041821413923095774,
    'subsample': 0.8742097017145511,
    'colsample_bytree': 0.5936492805139413,
    'gamma': 0.9463373617864398,
    'min_child_weight': 7}
```

Де params_focal – використовується для Focal Loss, а params_weighted – для Weighted Cross Entropy

Після навчання моделі було отримано було отримано наступну таблицю (Таблиця 2.5) з результатами класифікації.

Таблиця 2.5

Метрики для моделі з балансованими даними за допомогою RUS і FL / WCE

	RUS FL	RUS WCE
Accuracy Score	0.637363	0.637363
Precision Score	0.507463	0.747317
Recall Score	0.15198	0.489144
F1 Score	0.233907	0.591277

G-mean	0.335526	0.589031

А також було побудовано графіки візуалізацій матриць помилок див. додаток Є, Рис. Є.1 і див. ROC-кривої додаток Є, Рис. Є.2

2. Random Over Sampling + (Focal Loss / Weighted Cross Entropy)

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

```
params_focal = {
  'objective': focal_loss,
  'n_estimators': 258,
  'max_depth': 10,
  'learning_rate': 0.07331865190064507,
  'subsample': 0.7678677499159607,
  'colsample_bytree': 0.596916820568302,
  'gamma': 0.3378097005569924,
  'min_child_weight': 3}
params_weighted = {
  'objective': weighted_binary_cross_entropy,
  'n_estimators': 258,
  'max_depth': 10,
  'learning_rate': 0.07331865190064507,
  'subsample': 0.7678677499159607,
  'colsample_bytree': 0.596916820568302,
  'gamma': 0.3378097005569924,
  'min_child_weight': 3}
```

Після навчання моделі було отримано було отримано наступну таблицю (Таблиця 2.6) з результатами класифікації.

Таблиця 2.6

Метрики для моделі з балансованими даними за допомогою ROS і FL / WCE

	ROS FL	ROS WCE
Accuracy Score	0.354904	0.643468
Precision Score	0.363636	0.672115
Recall Score	0.012771	0.866539
F1 Score	0.024676	0.757043
G-mean	0.110769	0.471925

А також було побудовано графіки візуалізацій матриць помилок див. додаток Г, Рис. Г.3 і див. ROC-кривої додаток Г, Рис. Г.4

3. SMOTE + (Focal Loss / Weighted Cross Entropy)

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

```
params_focal = {
    'objective': focal_loss,
    'n_estimators': 143,
    'max_depth': 10,
    'learning_rate': 0.0012381122229706693,
    'subsample': 0.8764884885259474,
```

'colsample_bytree': 0.5122076493704834,

'gamma': 0.42260708044796647,

'min_child_weight': 6}

params_weighted = {

'objective': weighted_binary_cross_entropy,

'n_estimators': 143,

'max_depth': 10,

'learning_rate': 0.0012381122229706693,

'subsample': 0.8764884885259474,

'colsample_bytree': 0.5122076493704834,

'gamma': 0.42260708044796647,

'min_child_weight': 6}

Після навчання моделі було отримано було отримано наступну таблицю (Таблиця 2.7) з результатами класифікації.

Таблиця 2.7

Метрики для моделі з балансованими даними за допомогою SMOTE і FL / WCE

	SMOTE FL	SMOTE WCE
Accuracy Score	0.637363	0.637363
Precision Score	0.65	0.660118
Recall Score	0.066411	0.858238
F1 Score	0.12051	0.746252
G-mean	0.249474	0.437816

А також було побудовано графіки візуалізацій матриць помилок див. додаток Є, Рис. Є.5 і див. ROC-кривої додаток Є, Рис. Є.6

4. ADASYN + (Focal Loss / Weighted Cross Entropy)

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

```
params_focal = {
    'objective': focal_loss,
    'n_estimators': 190,
    'max_depth': 7,
    'learning_rate': 0.13836053931252873,
    'subsample': 0.6016105058143335,
    'colsample_bytree': 0.8645099541402509,
    'gamma': 0.7162834664094448,
    'min_child_weight': 3}
params_weighted = {
    'objective': weighted_binary_cross_entropy,
    'n_estimators': 190,
    'max_depth': 7,
    'learning_rate': 0.13836053931252873,
    'subsample': 0.6016105058143335,
    'colsample_bytree': 0.8645099541402509,
    'gamma': 0.7162834664094448,
    'min_child_weight': 3}
```

Після навчання моделі було отримано було отримано наступну таблицю (Таблиця 2.8) з результатами класифікації.

Таблиця 2.8

Метрики для моделі з балансованими даними за допомогою ADASYN і FL / WCE

	ADASYN FL	ADASYN

		WCE
Accuracy Score	0.363451	0.643468
Precision Score	1	0.680498
Recall Score	0.000639	0.733078
F1 Score	0.001276	0.70581
G-mean	0.02527	0.538155

А також було побудовано графіки візуалізацій матриць помилок див. додаток Є, Рис. 7. Є і див. ROC-кривої додаток Є, Рис. 8.Є.

5. SMOTETOMEK + (Focal Loss / Weighted Cross Entropy)

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

```
params_focal = {
    'objective': focal_loss,
    'max_depth': 10,
    'n_estimators': 150,
    'learning_rate': 0.5,
    'subsample': 0.8,
    'colsample_bytree': 0.9,
    'seed': 12}
params_weighted = {
    'objective': weighted_binary_cross_entropy,
    'max_depth': 10,
    'n_estimators': 150,
    'learning_rate': 0.5,
    'subsample': 0.8,
    'colsample_bytree': 0.9,
```

'seed': 12}

Після навчання моделі було отримано було отримано наступну таблицю (Таблиця 2.9) з результатами класифікації.

Таблиця 2.9

Метрики для моделі з балансованими даними за допомогою SMOTETOMEK і FL / WCE

	SMOTETOMEK FL	SMOTETOMEK WCE
Accuracy Score	0.36223	0.630851
Precision Score	0	0.676383
Recall Score	0	0.726054
F1 Score	0	0.700339
G-mean	0	0.531753

А також було побудовано графіки візуалізацій матриць помилок див. додаток Є, Рис. Є.9 і див. ROC-кривої додаток Є, Рис. Є.10

6. SMOTEENN + (Focal Loss / Weighted Cross Entropy)

Дана модель буде навчатися залучаючи наступні підібрані параметри за допомогою optuna:

params_focal = {

'objective': focal_loss,

'n_estimators': 168,

'max_depth': 6,

'learning_rate': 0.08509848665588302,

```
    'subsample': 0.9239135350691549,
    'colsample_bytree': 0.9043052520550187,
    'gamma': 0.1889174325488193,
    'min_child_weight': 1}
params_weighted = {
    'objective': weighted_binary_cross_entropy,
    'n_estimators': 168,
    'max_depth': 6,
    'learning_rate': 0.08509848665588302,
    'subsample': 0.9239135350691549,
    'colsample_bytree': 0.9043052520550187,
    'gamma': 0.1889174325488193,
    'min_child_weight': 1}
```

Після навчання моделі було отримано було отримано наступну таблицю (Таблиця 2.10) з результатами класифікації.

Таблиця 2.10

Метрики для моделі з балансованими даними за допомогою SMOTEENN і FL / WCE

	SMOTEENN FL	SMOTEENN WCE
Accuracy Score	0.420431	0.606024
Precision Score	0.520788	0.712707
Recall Score	0.15198	0.494253
F1 Score	0.235294	0.58371
G-mean	0.338562	0.566728

А також було побудовано графіки візуалізацій матриць помилок див. додаток Є, Рис. 11. Є і див. ROC-кривої додаток Є, Рис. 12. Є.

2.5.3.1 Аналіз результатів

Повна таблиця розрахованих метрик див. таблиця 2.11.

Таблиця 2.11

Результати

	Accuracy Score	Precision Score	Recall Score	F1 Score	G-mean
RUS FL	0.637363	0.507463	0.15198	0.233907	0.335526
RUS WCE	0.637363	0.747317	0.489144	0.591277	0.589031
ROS FL	0.354904	0.363636	0.012771	0.024676	0.110769
ROS WCE	0.643468	0.672115	0.866539	0.757043	0.471925
SMOTE FL	0.637363	0.65	0.066411	0.12051	0.249474
SMOTE WCE	0.637363	0.660118	0.858238	0.746252	0.437816
ADASYN FL	0.363451	1	0.000639	0.001276	0.02527
ADASYN WCE	0.643468	0.680498	0.733078	0.70581	0.538155
SMOTETOMEK FL	0.36223	0	0	0	0
SMOTETOMEK WCE	0.630851	0.676383	0.726054	0.700339	0.531753
SMOTEENN FL	0.420431	0.520788	0.15198	0.235294	0.338562
SMOTEENN WCE	0.606024	0.712707	0.494253	0.58371	0.566728

Згідно таблиці 2.11 можна зробити висновок, що:

Модель, що побудована на збалансованих даних методом RUS FL досягає помірних результатів. Вона має середню Accuracy Score, проте низькі значення Precision Score, Recall Score, F1 Score та G-mean. Це свідчить про недостатню здатність моделі виявляти позитивні класи і здатність до помилкових класифікацій.

Модель, що побудована на збалансованих даних методом RUS WCE показує кращі результати порівняно з RUS FL. Вона має високі значення Precision Score та F1 Score, що свідчить про її здатність точно визначати позитивні класи. Однак, Recall Score залишається помірним, що може означати пропуски у виявленні позитивних класів.

Модель, що побудована на збалансованих даних методом ROS FL показує незадовільні результати. Вона має низькі значення для всіх метрик, що вказує на її недостатню здатність враховувати дисбаланс у даних та виявляти позитивні класи.

Модель, що побудована на збалансованих даних методом ROS WCE досягає найкращих результатів порівняно з іншими методами. Вона має високу Accuracy Score, Precision Score, Recall Score та F1 Score, що свідчить про її здатність до точного класифікації позитивних класів. G-mean також є прийнятним, що вказує на здатність враховувати дисбаланс у даних.

Обидві моделі на збалансованих даних методом SMOTE досягають помірних результатів. Вони мають високі значення Precision Score, але низькі значення Recall Score, що свідчить про пропуски виявлення позитивних класів.

Модель, що побудована на збалансованих даних методом SMOTETOMEK FL показує дуже низькі значення для всіх метрик. Вона не здатна ефективно вирішити дисбаланс класів та проводити правильну класифікацію.

Модель, що побудована на збалансованих даних методом SMOTETOMEK WCE показує показує прийнятні результати. Вона має високі значення Precision Score та F1 Score, але помірне значення Recall Score, що вказує на її здатність до точного класифікації позитивних класів.

Модель, що побудована на збалансованих даних методом SMOTEENN FL показує помірні результати. Вона має високе значення Precision Score, але низьке значення Recall Score, що вказує на пропуски виявлення позитивних класів.

Модель, що побудована на збалансованих даних методом SMOTEENN WCE показує прийнятні результати. Вона має високі значення Precision Score та F1 Score, але помірне значення Recall Score, що вказує на її здатність до точного класифікації позитивних класів.

Отже, на основі порівняння метрик можна виділити три найкращі методи для балансування датасету, що розглядався у роботі:

1. ROS WCE. має високу точність, здатність до точного визначення позитивних класів та ефективну роботу з дисбалансом даних.
2. RUS WCE. має високу точність та гарну здатність до точного визначення позитивних класів, але може мати пропуски у виявленні позитивних класів.
3. SMOTEENN WCE. має хороші результати щодо точності та здатність до точного визначення позитивних класів, але може мати пропуски у виявленні позитивних класів.

Висновки

В розділі було проведено глибоке дослідження бізнес-кредитів, зокрема було зосереджено увагу на особливостях набору даних. Було застосовано три етапи аналізу даних: IDA, EDA і CDA, що допомогло нам зрозуміти основні характеристики, структуру та залежності в даних.

Основна увага була приділена проблемі незбалансованості даних у контексті класифікації. Було розглянуто та проаналізовано різні методи балансування даних, включаючи SMOTE, ADASYN, Random UnderSampling (RUS), Random OverSampling (ROS) і комбіновані методи, такі як SMOTETomek та SMOTEENN.

Було проведено оцінку якості моделей класифікації за допомогою різних метрик, включаючи Accuracy, Precision, Recall, F1 Score та G-mean. Результати показали, що модель, збалансована за допомогою методу ROS і тренована з використанням вагової крос-ентропії як функції втрат, показала найкращі результати за всіма метриками.

Все це свідчить про те, що робота з незбалансованими даними вимагає відмінного від типового підходу, і вибір правильної стратегії балансування даних може значно покращити якість моделі класифікації. Зважаючи на значну незбалансованість у бізнес-кредитах, ці висновки особливо важливі для подальшого дослідження та розробки моделей прогнозування

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено детальне дослідження проблеми класифікації незбалансованих даних на прикладі прогнозування дефолту бізнес-позик. Робота базувалася на глибокому вивченні теоретичних аспектів класифікації та балансування даних, а також на аналізі існуючих методів класифікації незбалансованих даних.

Зокрема, було вивчено специфіку бізнес-кредитів та особливості набору даних, проведено три етапи аналізу даних (IDA, EDA, CDA), та проаналізовано методи підвищення якості моделей класифікації. Значну увагу було приділено аналізу різних методів балансування даних та їх застосуванню для класифікації незбалансованих даних.

В результаті дослідження було встановлено, що модель, збалансована за допомогою методу ROS WCE і тренована з використанням вагової крос-ентропії як функції втрат, показала найкращі результати за всіма метриками.

Ці висновки можуть бути використані банками та іншими фінансовими установами для покращення їх моделей прогнозування дефолту бізнес-позик. Зокрема, вони можуть враховувати специфіку незбалансованих даних та використовувати ефективні методи балансування для підвищення якості своїх прогнозних моделей.

В майбутньому, ці висновки можуть стати основою для подальших досліджень у цій області, зокрема, для розробки нових методів класифікації незбалансованих даних та їх застосування у різних областях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lin T.Y., Goyal P., Girshick R., He K., Dollar P. Focal Loss for Dense Object Detection // Facebook AI Research. – 2018, 1-10, https://arxiv.org/pdf/1708.02002.pdf

2. Zhou, Z.Y., Huang, H. and Fang, B.H. (2021) Application of Weighted Cross-Entropy Loss Function in Intrusion Detection. Journal of Computer and Communications, 9, 1-21. https://doi.org/10.4236/jcc.2021.911001

3. Fern´andez, A. SMOTE for Learning from Imbalanced Data: Progress andChallenges, Marking the 15-year Anniversary / A. Fern´andez, S. Garc´ıa, F. Herrera // Journal of Artificial Intelligence Research : Електронний журнал. – URL: https://www.jair.org/index.php/jair/article/view/11192/26406. – Дата публікації: 2018.

4. Akosa, J. S. Predictive Accuracy: A Misleading Performance Measure for Highly Imbalanced Data / J. S. Akosa // Paper 942-2017 . – Oklahoma : Oklahoma State University, 2017. – С. 2-4.

5. Aman Arora’s Blog : Електронний ресурс. – URL: https://amaarora.github.io/posts/2020-06-29-FocalLoss.html#so-why-did-that-work-what-did-focal-loss-do-to-make-it-work (дата звернення: 11.05.2023)

6. Breiman, L., Random Forests, Machine Learning 45(1), 5-32, 2001.

7. Random Forests // UCB Statistics: Електронний ресурс. – URL: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#workings (дата звернення: 11.05.2023)

8. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting / Y. Freund,, R. E. Schapire // Journal of Computer and System Sciences. – 1997. – Т. 55, № 1. – С. 119-139. – ISSN 0022-0000

9. Drucker, Harris Boosting Decision Trees / Harris Drucker, Cotia tortes // Advances in Neural Information Processing Systems · . – New Jersey : 1995. – С. 4-6.

10. Bentéjac, C., Csörgő, A. & Martínez-Muñoz, G. A comparative analysis of gradient boosting algorithms. Artif Intell Rev 54, 1937–1967 (2021). https://doi.org/10.1007/s10462-020-09896-5

11. Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

12. «Multi-class AdaBoos»t / Ji Zhu, Hui Zou, Saharon Rosset // Statistics and Its Interface. – 2009. – Т. 2, № 60. – С. 2-12.

13. Wikipedia : Електронний ресурс. – URL: https://en.wikipedia.org/wiki/XGBoost (дата звернення: 11.05.2023)

14. RandomForests // Index of /~matthewb/pages/notes/pdf/ensembles : Електронний ресурс. – URL: https://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/ensembles/ (дата звернення: 11.05.2023)

15. Z. Wang, C. Wu, K. Zheng, X. Niu and X. Wang, «SMOTETomek-Based Resampling for Personality Recognition» in IEEE Access, vol. 7, pp. 129678-129689, 2019, doi: 10.1109/ACCESS.2019.2940061.

16. Haibo He, Yang Bai, E. A. Garcia and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 1322-1328, doi: 10.1109/IJCNN.2008.4633969.

17. Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. Newsl. 6, 1 (June 2004), 20–29. https://doi.org/10.1145/1007730.1007735

18. A hybrid sampling algorithm combining M-SMOTE and ENN based on Random forest for medical imbalanced data / Zhaozhao Xu, Derong Shen, Tiezheng

Nie, Yue Kou // Journal of Biomedical Informatics. – 2020. – № 107. – ISSN 1532-0464

19. Behrens, John T.. Principles and procedures of exploratory data analysis. Psychological Methods 2 (1997): 131-160.

20. Baillie M, le Cessie S, Schmidt CO, Lusa L, Huebner M, for the Topic Group "Initial Data Analysis" of the STRATOS Initiative (2022) Ten simple rules for initial data analysis. PLoS Comput Biol 18(2): e1009819. https://doi.org/10.1371/journal.pcbi.1009819

21. A Note About Gradients in Classification Problems // bokbokbok doks URL: https://orchardbirds.github.io/bokbokbok/derivations/note.html (дата звернення: 11.05.2023).

22. Ho Yu, C., Exploratory data analysis in the context of data mining and resampling.. International Journal of Psychological Research, –2010. Т–3. – № 1, 9-22.

23. McKinney, Wes. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. 1 : O'Reilly Media, 2013.

24. Lutz, Mark. Learning Python. 5 Beijing: O'Reilly, 2013.

ДОДАТКИ

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Найменування	Кількість аркушів	Примітки
1				
2		Документація		
3				
4	САУ.КР.23.01.ПЗ	Пояснювальна записка	159	Формат А4
5				
6	САУ.КР.23.01.ДМ	Демонстраційний матеріал	1	Презентація PowerPoint
7				
8	САУ.КР.23.01.КР	Копія роботи	1	Диск CD-R
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

					САУ.КР.23.01.ДА.ПЗ.
Змін.	Аркуш	№ докум.	Підпис	Дата	

Розроб.	Бутенко Н.В.			**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Аркушів
К. розд.	Купенко О.П.						
Керівн.	Купенко О.П.				НТУ «ДП», 12; 124-19-2		
Н.контр.	Хом’як Т.В.						
Зав. каф.	Желдак Т.А.						

ДОДАТОК Б. Відгук на кваліфікаційну роботу

Відгук

на кваліфікаційну роботу бакалавра

студента(ки) групи *124 – 19– 2*

спеціальності *124 Системний аналіз*

Тема кваліфікаційної роботи: «Аналіз методів класифікації незбалансованих даних на прикладі прогнозу дефолту бізнес займів»

Обсяг кваліфікаційної роботи 159 стор.

Мета кваліфікаційної роботи: шляхом поєднання методів передобробки вхідних даних та методів і прийомів класифікації незбалансованих даних обрати ефективний спосіб прогнозування дефолту бізнес-позик, оцінити ефективність різних підходів на основі наявних даних.

Актуальність теми пов’язана зі зменшенням ризиків неповернення бізнес-позик банківським та кредитним установам.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра спеціальності 124 Системний аналіз, оскільки системний підхід до методології попередньої обробки даних та їх підготовки до навчання якісних класифікаційних моделей є одним з найважливіших аспектів роботи інформаційних, та аналітичних систем.

Виконані в кваліфікаційній роботі завдання відповідають вимогам освітньо-кваліфікаційного рівня бакалавра. Оригінальність наукових рішень полягає в сучасному та системному погляді на область проблем, які пов’язані з обробкою даних для задачі прогнозування неповернення бізнес-займів.

Практичне значення результатів кваліфікаційної роботи полягає у реалізації поєднання розвідувального аналізу даних з методами підвищення точності у задачах класифікації незбалансованихданих.

Висновки підтверджують можливість використання результатів роботи в банківських та кредитних установах.

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

У роботі відзначено такі недоліки: очікувався більш творчий підхід до обробки пропущених значень та категорійних даних.

Кваліфікаційна робота в цілому заслуговує оцінки: 93 – «відмінно».

З урахуванням висловлених зауважень автор заслуговує присвоєння кваліфікації «бакалавр з системного аналізу».

Керівник кваліфікаційної роботи бакалавра,
доктор фізико-математичних наук,
професор кафедри САУ Купенко Ольга Петрівна

ОДАТОК В. Рецензія

РЕЦЕНЗІЯ

на випускну бакалаврську роботу студентки групи 124 – 19– 2

спеціальності 124 Системний аналіз

Бутенко Надії Віталіївни

«АНАЛІЗ МЕТОДІВ КЛАСИФІКАЦІЇ НЕЗБАЛАНСОВАНИХ ДАНИХ НА ПРИКЛАДІ ПРОГНОЗУ ДЕФОЛТУ БІЗНЕС ЗАЙМІВ»

Безумовно, самим значущим видом фінансового ризику є саме кредитний ризик, адже він є часткою невизначеності через те, що клієнт фінансової організації може не повернути кошти кредитору. Інакше кажучи, для кредитора можлива ситуація, коли він втратить частину або повну суму кредиту та процентів за користування коштами з причини впливу зовнішніх обставин, таких як невиконання іншою стороною договору своїх зобов'язань. З іншого боку у реальних банківських даних часто виникає проблема незбалансованості класів, коли кількість невідшкодованих кредитів значно менша за кількість відшкодованих. У зв'язку з цим тема випускної роботи Н.В. Бутенко безперечно є актуальною.

Зміст і обсяг роботи відповідають вимогам до випускних робіт бакалаврів. Матеріал добре організовано і викладено технічно грамотною мовою. Поставлені в кваліфікаційній роботі задачі виконані в повному обсязі. Автором вивчено специфіку бізнес-кредитів та особливості набору даних, проведено три етапи аналізу даних (IDA, EDA, CDA), та проаналізовано методи підвищення якості моделей класифікації. Значну увагу було приділено аналізу різних методів балансування даних та їх застосуванню для класифікації незбалансованих даних. В результаті дослідження було встановлено, що модель, збалансована за допомогою методу ROS WCE і тренована з використанням вагової крос-ентропії як функції втрат, показала найкращі результати за всіма метриками.

До недоліків можна віднести орфографічні помилки у тексті пояснювальної записки. Крім того, нажаль деякі практичні аспекти врахування специфіки незбалансованих даних та використання відповідних методів балансування автором розглянуто в обмеженому обсязі.

Вважаю, що вказані недоліки в цілому не впливають на високу оцінку роботи. Випускна робота може бути оцінена на 95 «відмінно», а студентка Бутенко Надія Віталіївна заслуговує присвоєння кваліфікації «бакалавр з системного аналізу».

Рецензент:
д.т.н., професор,
завідувач кафедри інформаційних технологій
та комп'ютерної інженерії _____ Володимир ГНАТУШЕНКО

ДОДАТОК В.

ДОДАТОК Г

	count	mean	std	min	25%	50%	75%	max
Application: Buy Rate	9795.0	1.30	0.06	1.09	1.25	1.28	1.38	1.50
Application: Funded Amount	9795.0	32622.28	49488.47	5000.00	9500.00	15000.00	30000.00	250000.00
Application: Origination Fee	9795.0	0.38	0.66	0.00	0.00	0.00	1.00	2.00
customer Age	9608.0	48.09	12.91	0.00	40.00	47.00	55.00	178.00
Amount	9786.0	32642.12	49551.65	56.00	9500.00	15000.00	30000.00	250000.00
Applications received All Time	9793.0	2.48	2.43	1.00	1.00	2.00	3.00	41.00
Applications received by last 1 Month	9793.0	1.14	0.44	1.00	1.00	1.00	1.00	8.00
Applications received by last 3 Months	9793.0	1.30	0.69	1.00	1.00	1.00	1.00	10.00
Applications received by last 6 Months	9793.0	1.59	1.02	1.00	1.00	1.00	2.00	26.00
Average Daily Negatives	9795.0	0.60	1.03	0.00	0.00	0.00	0.79	12.33
Average Monthly Sales	9795.0	81665.85	212771.03	0.00	16539.16	29269.45	62348.50	6728746.46
Avg Daily Bank Balance	9795.0	13501.37	35710.51	-316.00	1861.12	4211.07	10418.81	997212.25
Avg Number of Monthly Deposits	9795.0	27.84	33.52	0.00	9.33	20.75	35.82	969.67
Brokers submitted All Time	9793.0	1.78	1.65	1.00	1.00	1.00	2.00	34.00
Brokers submitted last 1 month	9793.0	1.09	0.37	1.00	1.00	1.00	1.00	6.00
Brokers submitted last 3 months	9793.0	1.18	0.55	1.00	1.00	1.00	1.00	9.00
Brokers submitted last 6 months	9793.0	1.31	0.77	1.00	1.00	1.00	1.00	22.00
Credit Score	9767.0	610.37	70.08	0.00	557.00	615.00	662.00	816.00
Daily Bank Balance v/s Daily Payment	8472.0	45.59	66.98	0.02	14.38	27.33	51.88	1848.93
Days	9786.0	143.22	22.42	66.00	132.00	132.00	154.00	378.00
Factor Rate	9786.0	1.40	0.06	0.00	1.37	1.38	1.44	1.50
Inquiry Count	9738.0	18.27	15.94	0.00	8.00	14.00	24.00	197.00
Months	9786.0	6.71	1.06	3.14	6.05	6.29	7.33	18.00
Number of Trade Lines	9791.0	22.80	13.30	0.00	13.00	21.00	30.00	111.00
Open Bankruptcy	9791.0	0.10	0.30	0.00	0.00	0.00	0.00	1.00
Public Records	9738.0	0.29	0.93	0.00	0.00	0.00	0.00	20.00
Sales to Payment	8416.0	5.46	6.23	0.00	0.00	2.20	10.50	24.07
Satisfactory	9791.0	19.86	13.00	0.00	10.00	18.00	27.00	103.00
Sum of Monthly Personal Debt	9791.0	5474.61	17354.90	0.00	1148.00	2542.00	4774.50	553673.00
Time In Business Actual	9373.0	9.60	22.03	0.00	2.20	5.30	11.50	394.20

	count	mean	std	min	25%	50%	75%	max
Volume - 4 Months Ago	6642.0	103698.19	279571.62	0.00	15919.35	33705.98	82351.30	6067111.23
Volume - 6 Months Ago	5682.0	99705.17	243595.22	0.00	13423.20	32551.24	81402.99	6020776.00
Volume - Three Months Ago	9514.0	97547.18	291754.56	0.00	16652.80	31169.22	71909.63	9290265.01
Yearly Total Sales	9795.0	979713.59	2552570.04	0.00	198404.82	350751.48	747944.16	80744957.52

Рис. Г.1. Статистичний опис змінних

ДОДАТОК Д

]:

	Columns	Count	Percentage(%)
7	crime_record	4916	66.91
40	Volume - 6 Months Ago	2603	35.43
39	Volume - 4 Months Ago	1992	27.11
33	Sales to Payment	1379	18.77
21	Daily Bank Balance v/s Daily Payment	1232	16.77
37	Time In Business Actual	399	5.43
4	Primary Contact Gender	353	4.80
41	Volume - Three Months Ago	243	3.31
5	customer Age	185	2.52
26	Inquiry Count	48	0.65
32	Public Records	48	0.65
20	Credit Score	25	0.34
22	Days	9	0.12
23	Factor Rate	9	0.12
27	Months	9	0.12
6	Amount	9	0.12
34	Satisfactory	3	0.04
30	Open Bankruptcy	3	0.04
28	Number of Trade Lines	3	0.04
36	Sum of Monthly Personal Debt	3	0.04
35	Shipping State	0	0.00
31	Position	0	0.00
29	Office Space	0	0.00
38	Type	0	0.00
42	Yearly Total Sales	0	0.00
0	Application: Buy Rate	0	0.00
25	Industry	0	0.00
24	Has Website	0	0.00
2	Application: Origination Fee	0	0.00
3	Application: Remittance Frequency	0	0.00
8	Applications received All Time	0	0.00
9	Applications received by last 1 Month	0	0.00
10	Applications received by last 3 Months	0	0.00
11	Applications received by last 6 Months	0	0.00
12	Average Daily Negatives	0	0.00
13	Average Monthly Sales	0	0.00
14	Avg Daily Bank Balance	0	0.00
15	Avg Number of Monthly Deposits	0	0.00

	Columns	Count	Percentage(%)
16	Brokers submitted All Time	0	0.00
17	Brokers submitted last 1 month	0	0.00
18	Brokers submitted last 3 months	0	0.00
19	Brokers submitted last 6 months	0	0.00
1	Application Funded Amount	0	0.00
43	Outcome	0	0.00

Рис. Д.1. Відсоток пропущених значень у наборі даних

ДОДАТОК Е

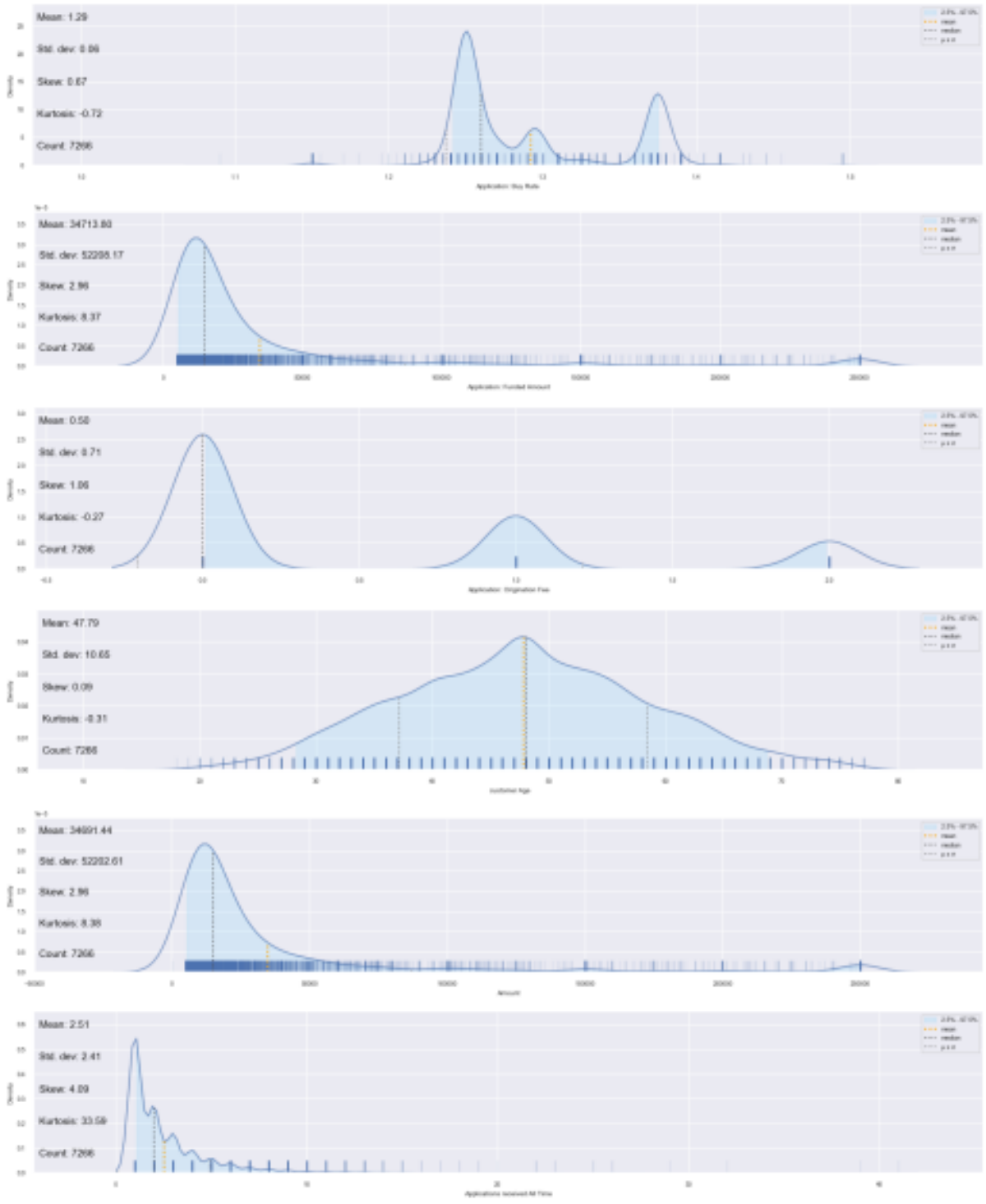


Рис. Е.1 – Розподіл даних перші 6 змінних

Рис. Е.2 – Розподіл даних

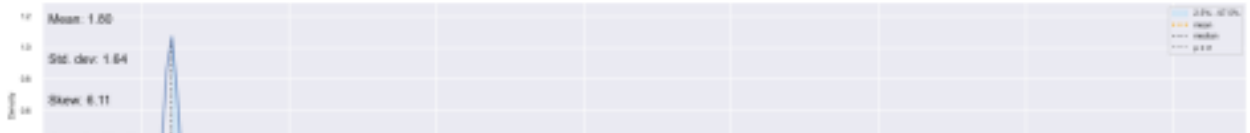


Рис. Е.3 – Розподіл даних

Рис. Е.4 – Розподіл даних

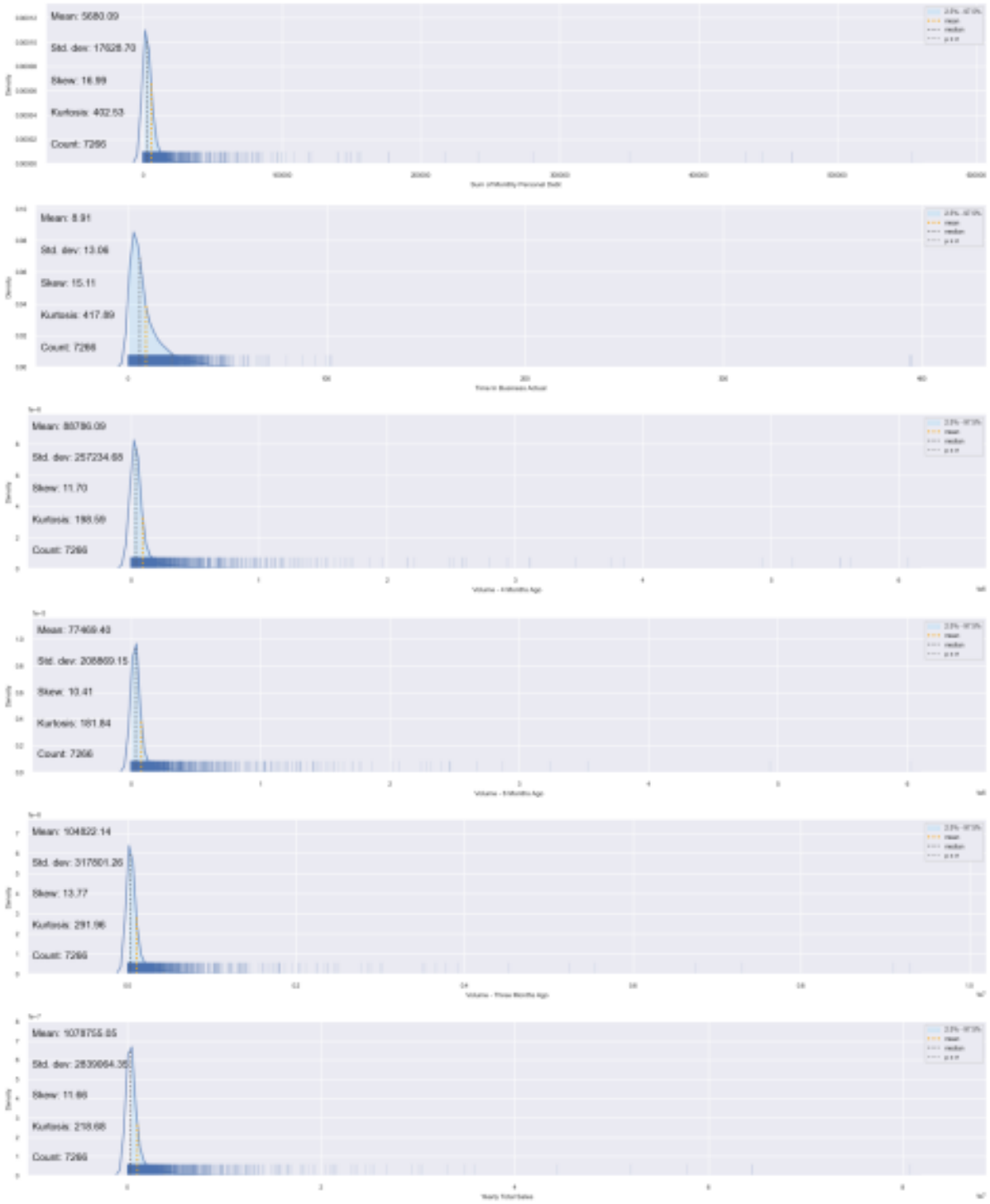
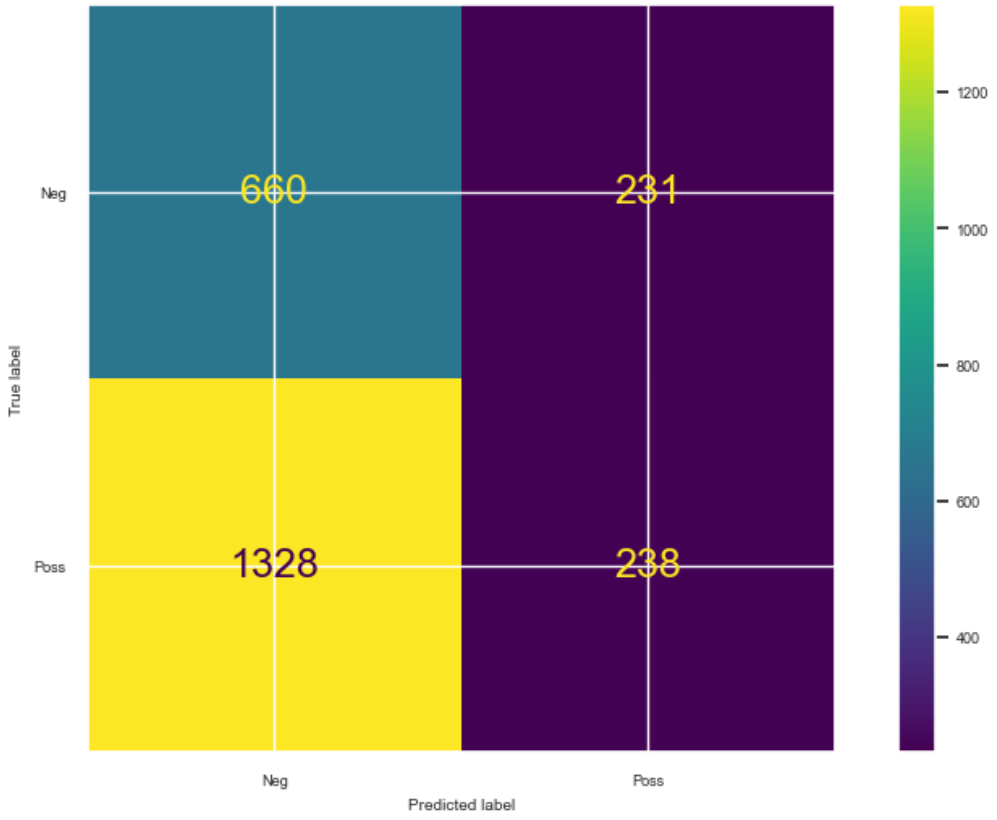


Рис. Е.5 – Розподіл даних

ДОДАТОК Є



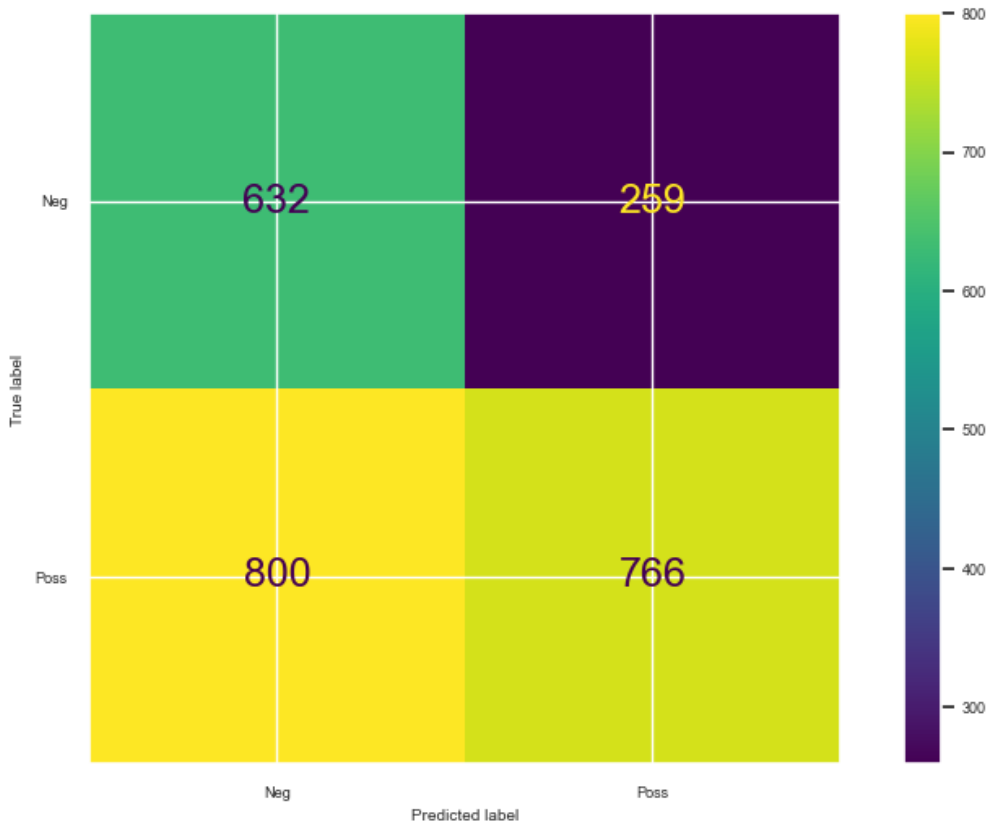
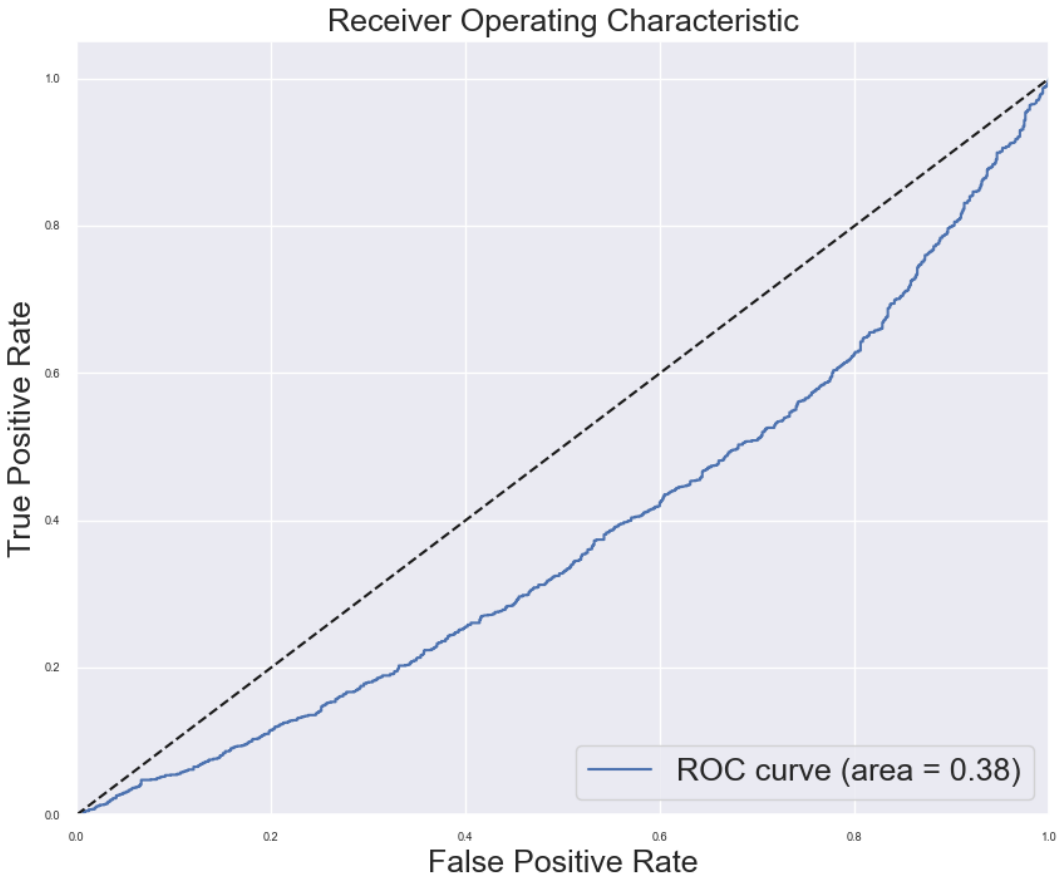


Рис. Є.1. Матриця помилок збалансованого набору даних RUS FL і RUS WCE відповідно



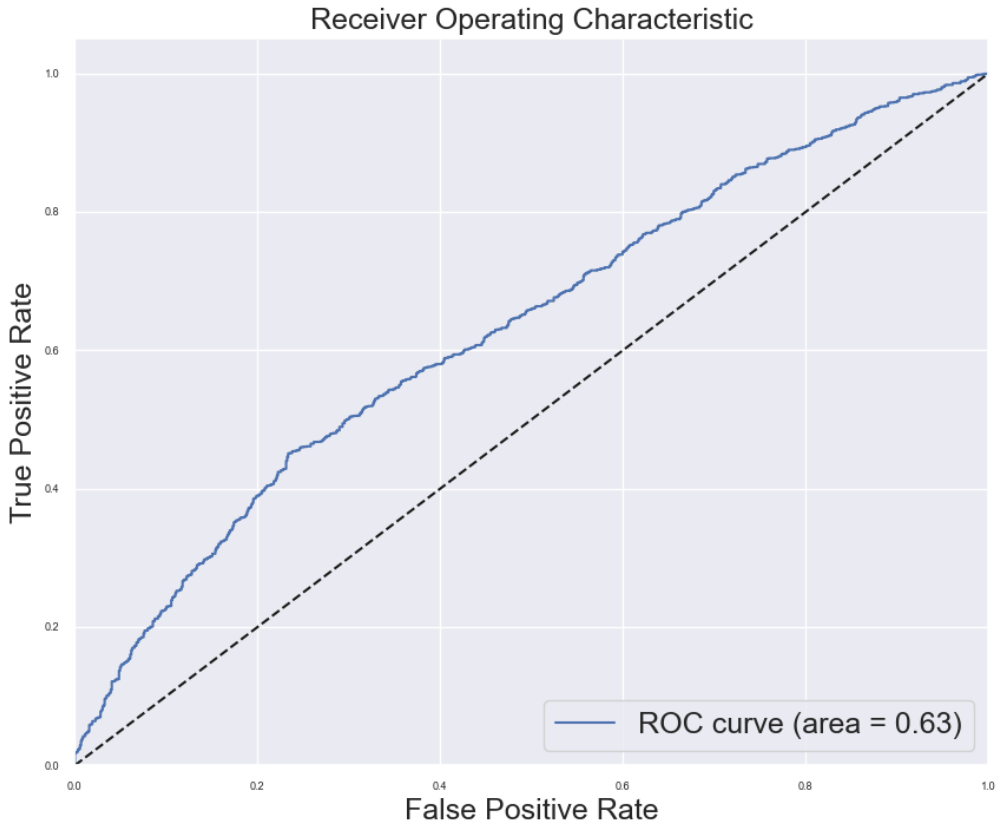


Рис. Є.2. Графік ROC-кривої збалансованого набору даних RUS FL і RUS WCE відповідно

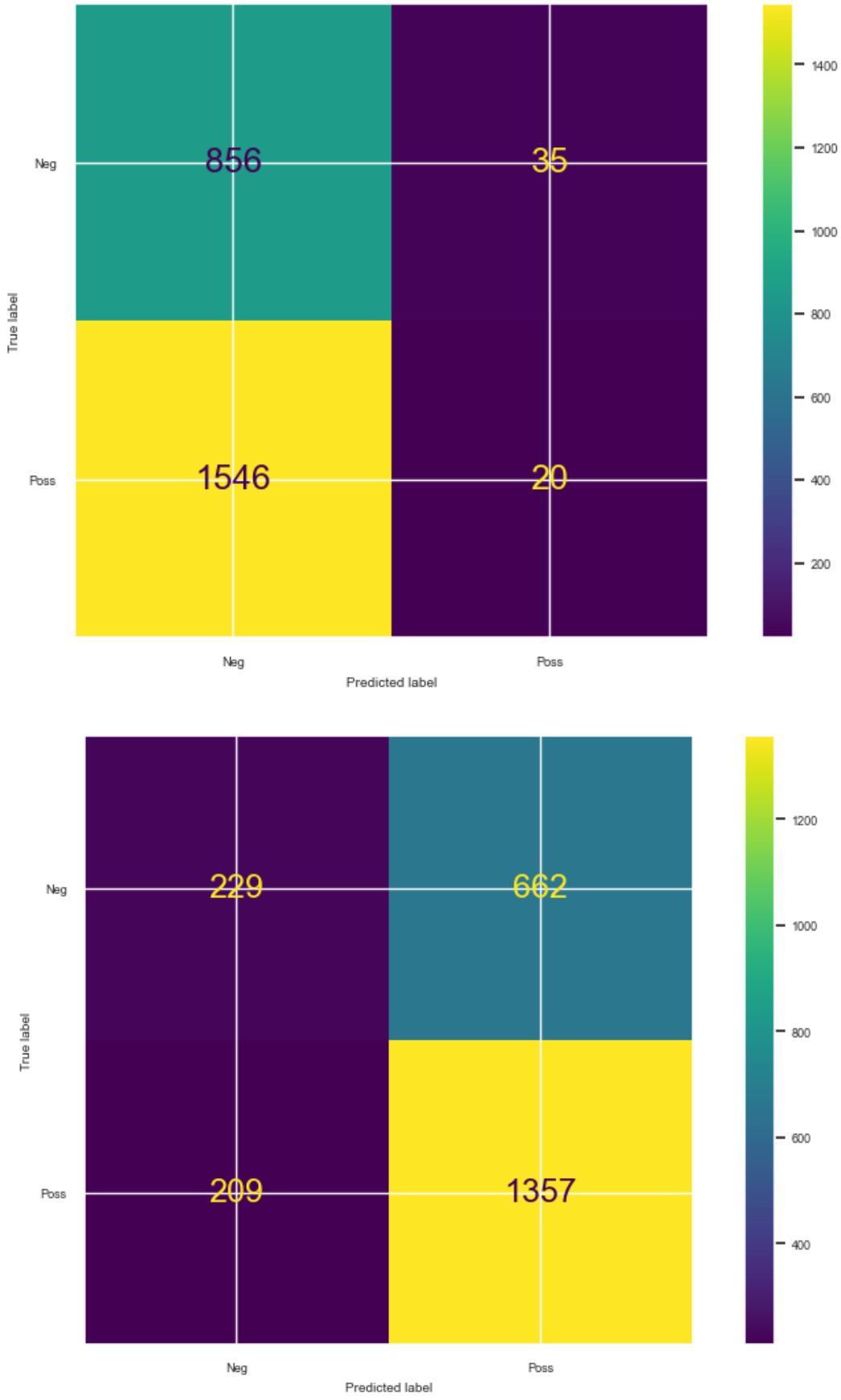


Рис. Є.3. Матриця помилок збалансованого набору даних ROS FL і ROS WCE відповідно

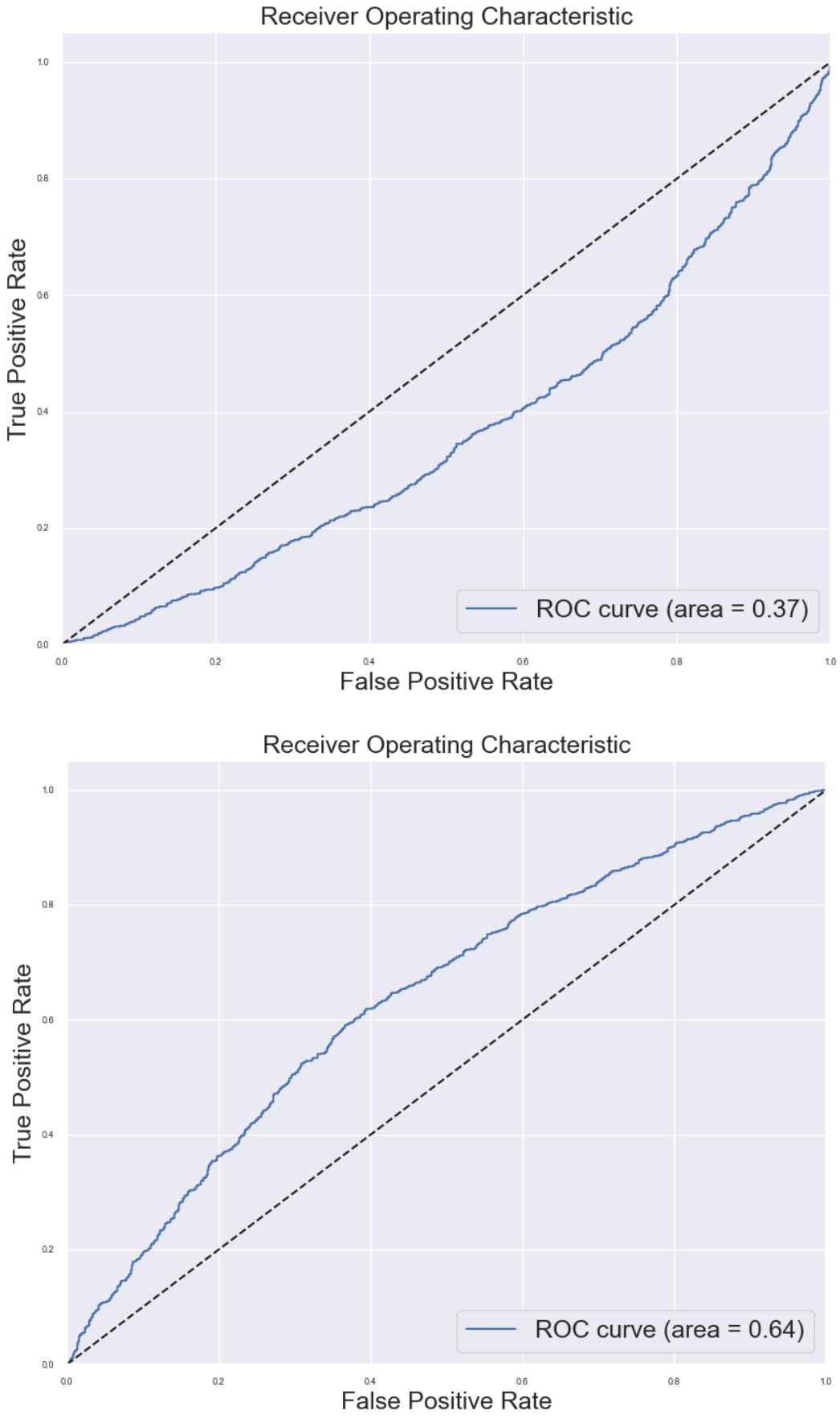


Рис. Є.4. Графік ROC-кривої збалансованого набору даних ROS FL і ROS WCE відповідно

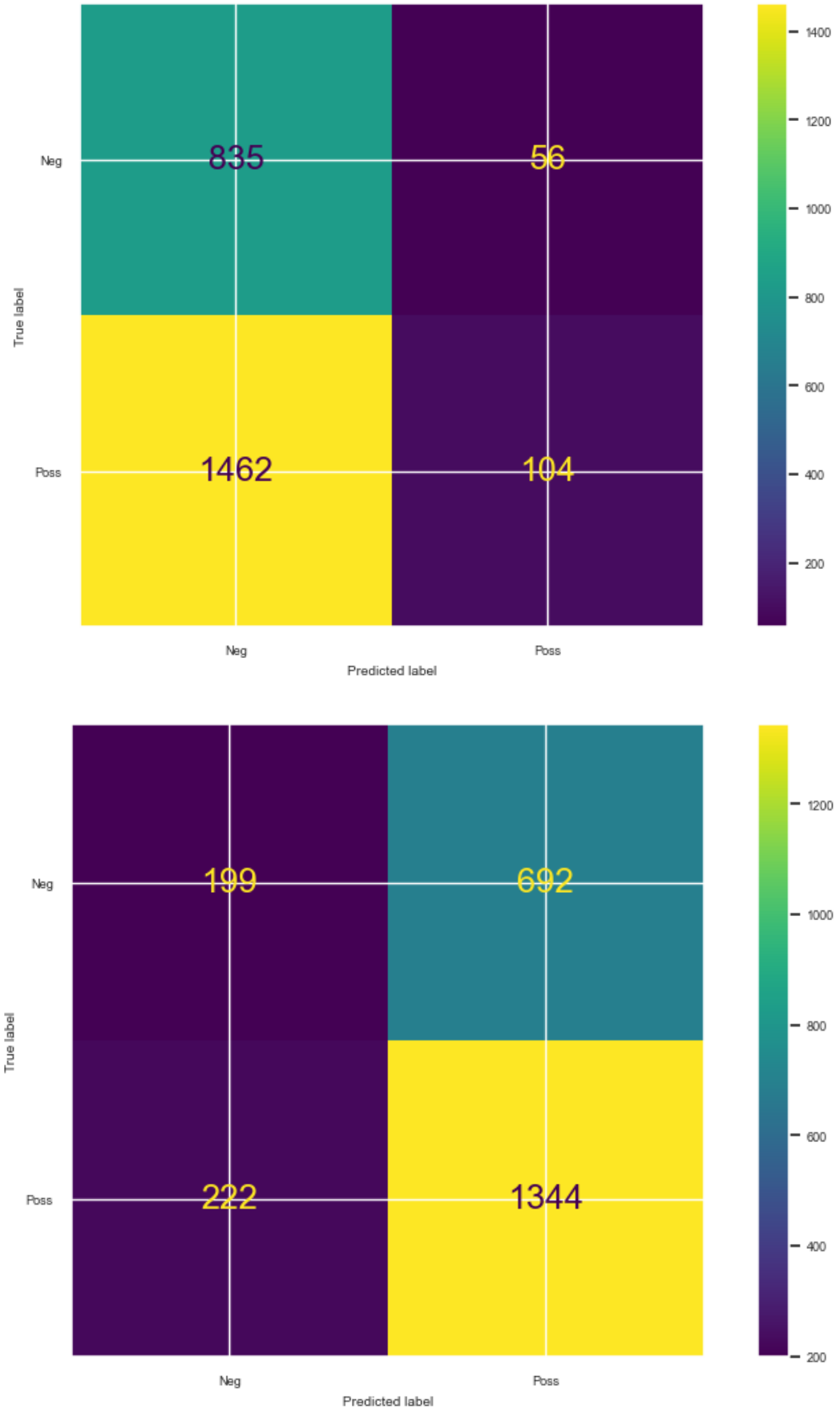
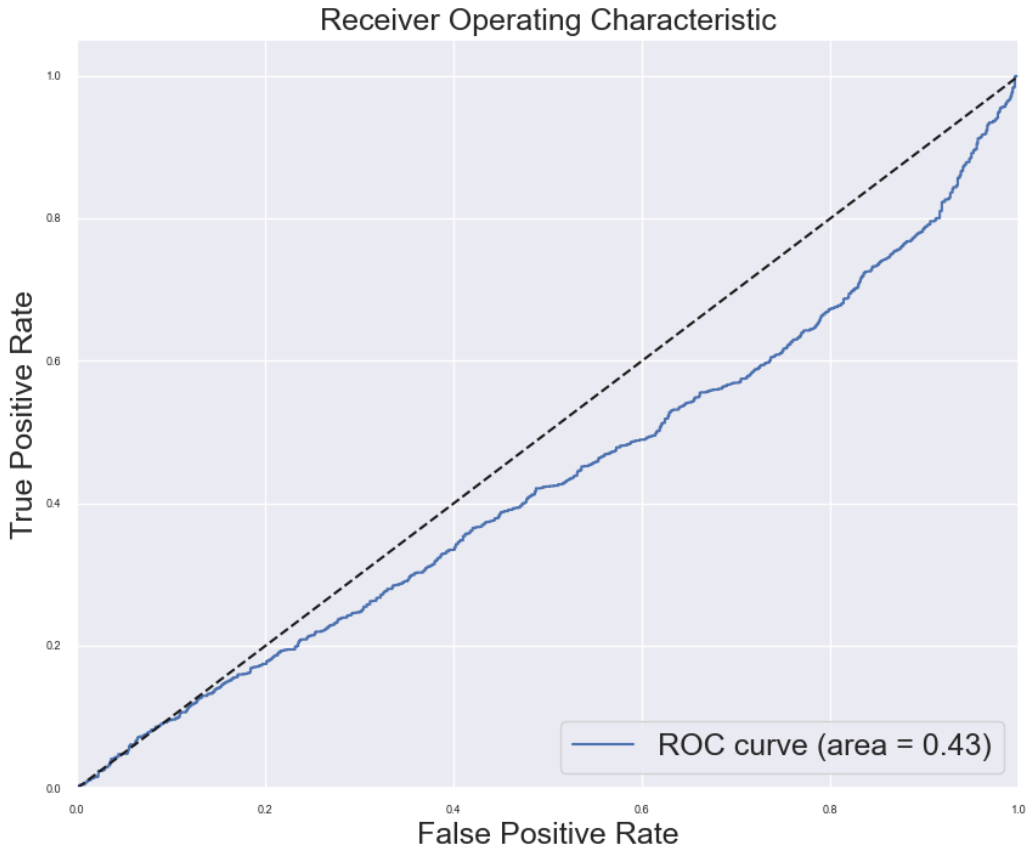


Рис. Є.5. Матриця помилок збалансованого набору даних SMOTE FL і SMOTE WCE відповідно



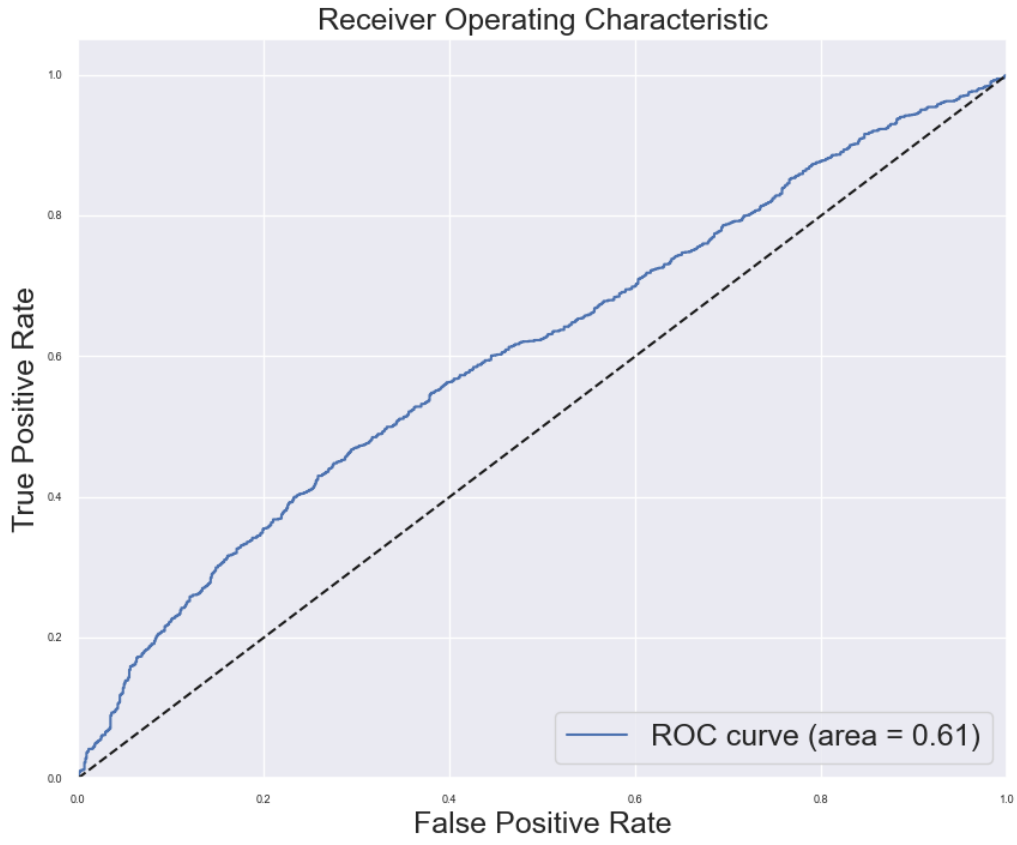


Рис. Є.6. Графік ROC-кривої збалансованого набору даних SMOTE FL і SMOTE WCE відповідно

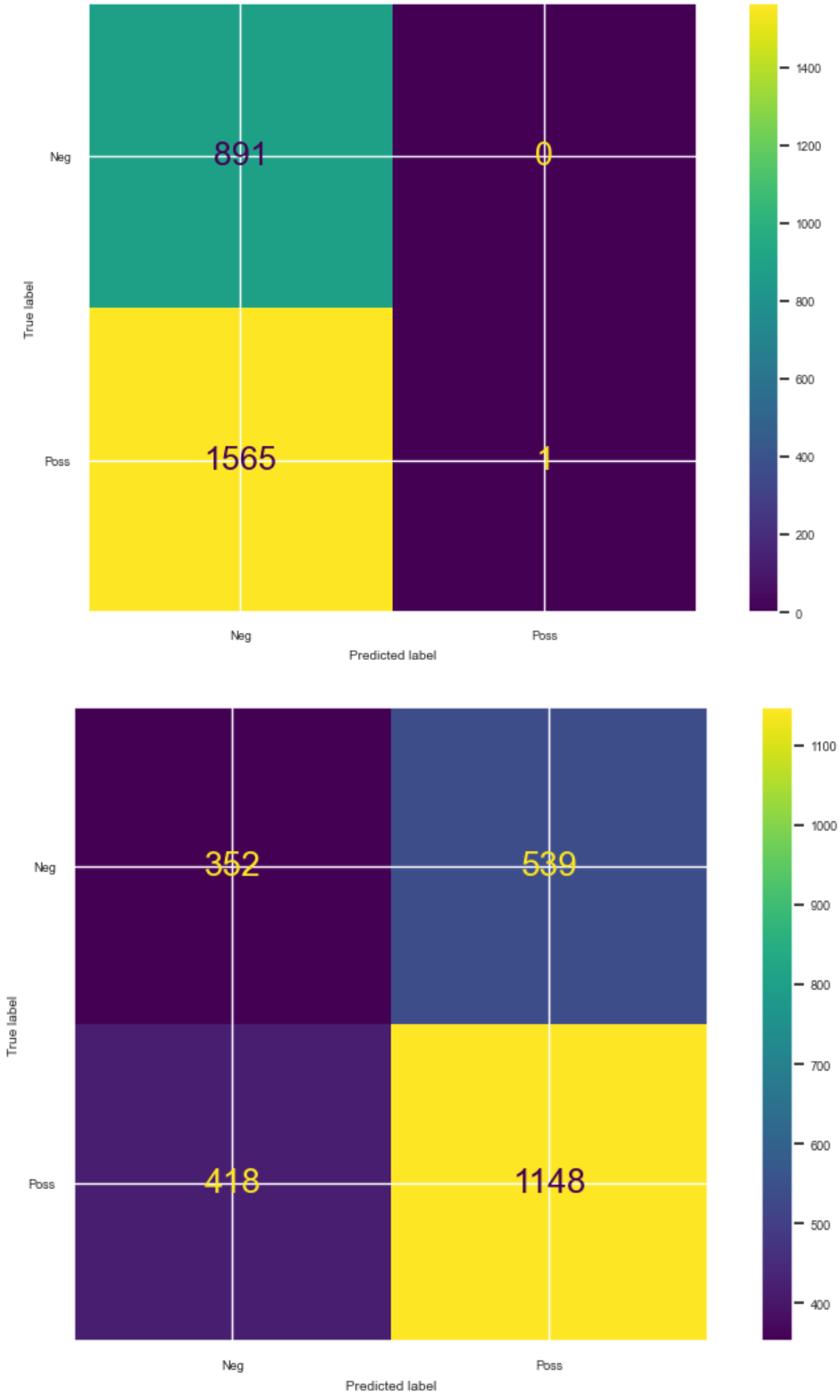


Рис. Є.7. Матриця помилок збалансованого набору даних ADASYN FL і ADASYN WCE відповідно

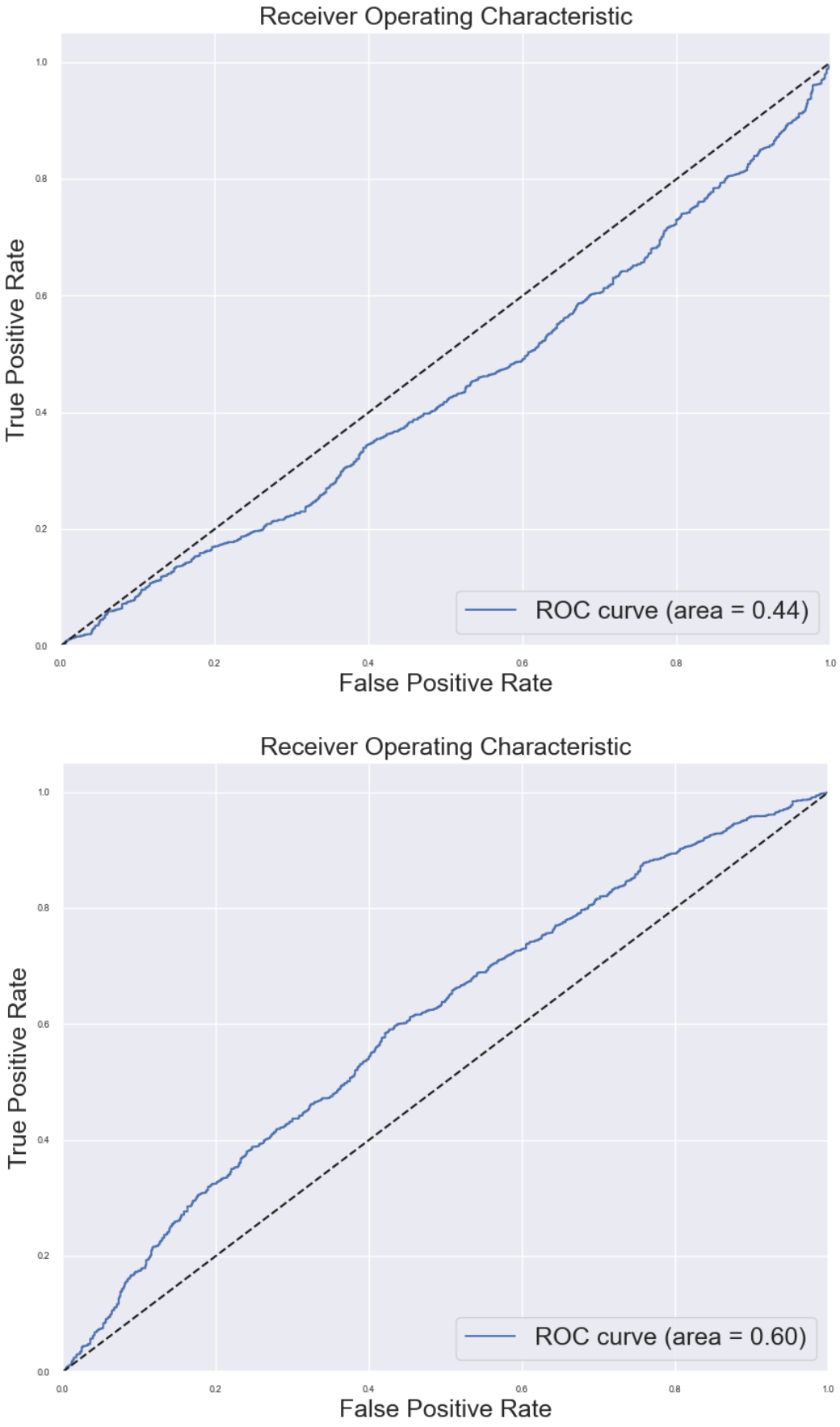


Рис. Є.8. Графік ROC-кривої збалансованого набору даних ADASYN FL і ADASYN WCE відповідно

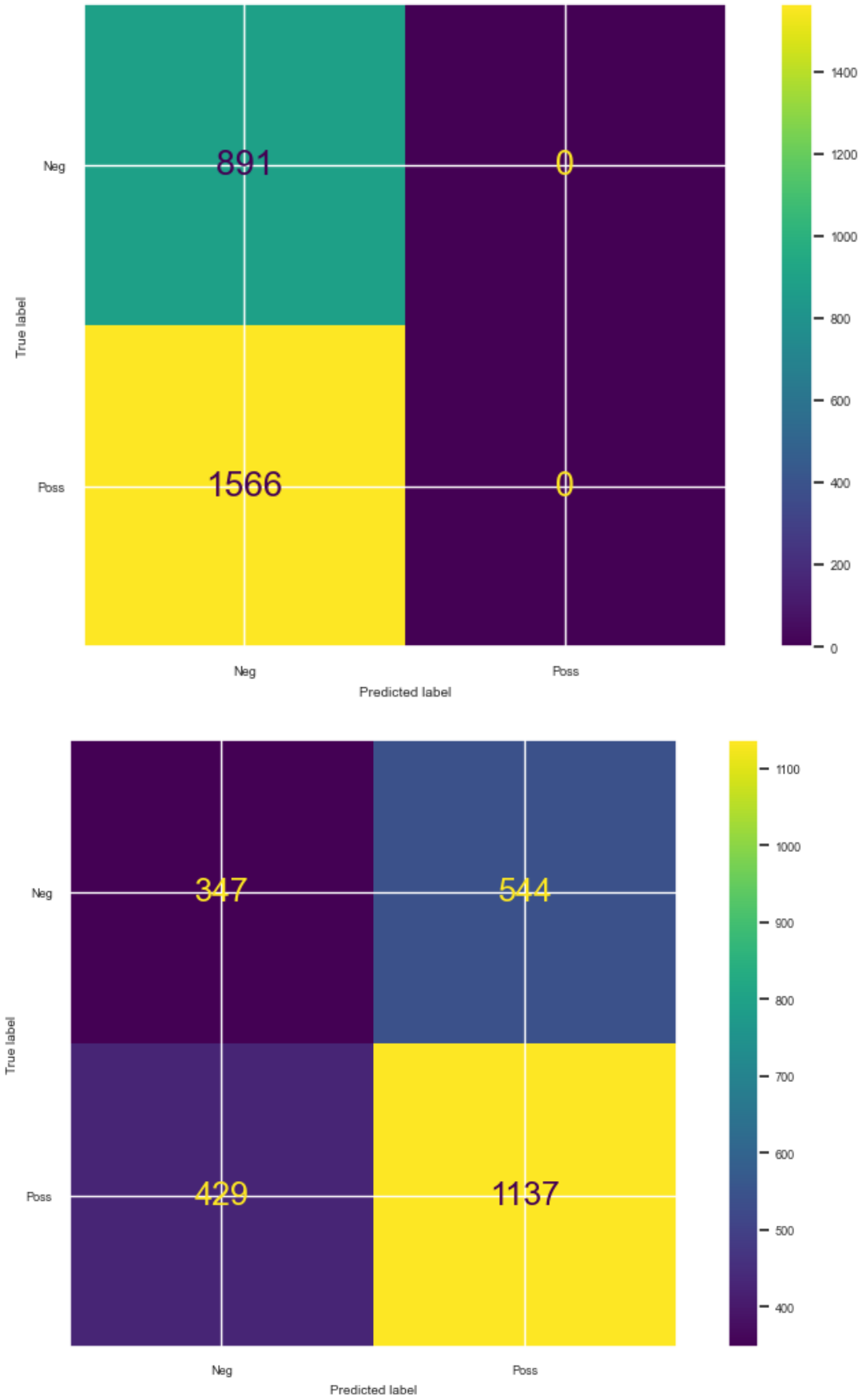


Рис. Є.9. Матриця помилок збалансованого набору даних SMOTETOMEK FL і SMOTETOMEK WCE відповідно

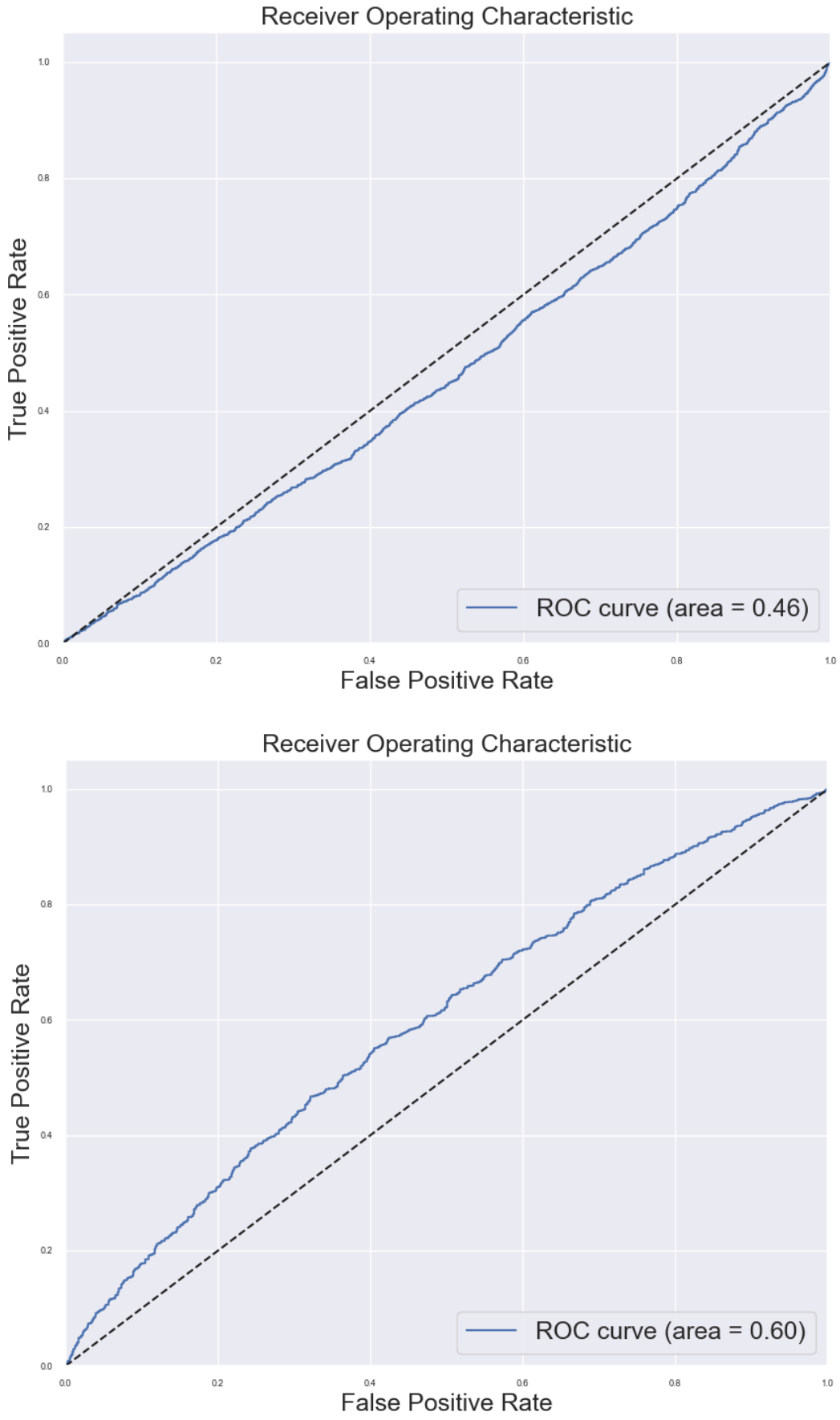


Рис. Є.10. Графік ROC-кривої збалансованого набору даних SMOTETOMEK FL і SMOTETOMEK WCE відповідно

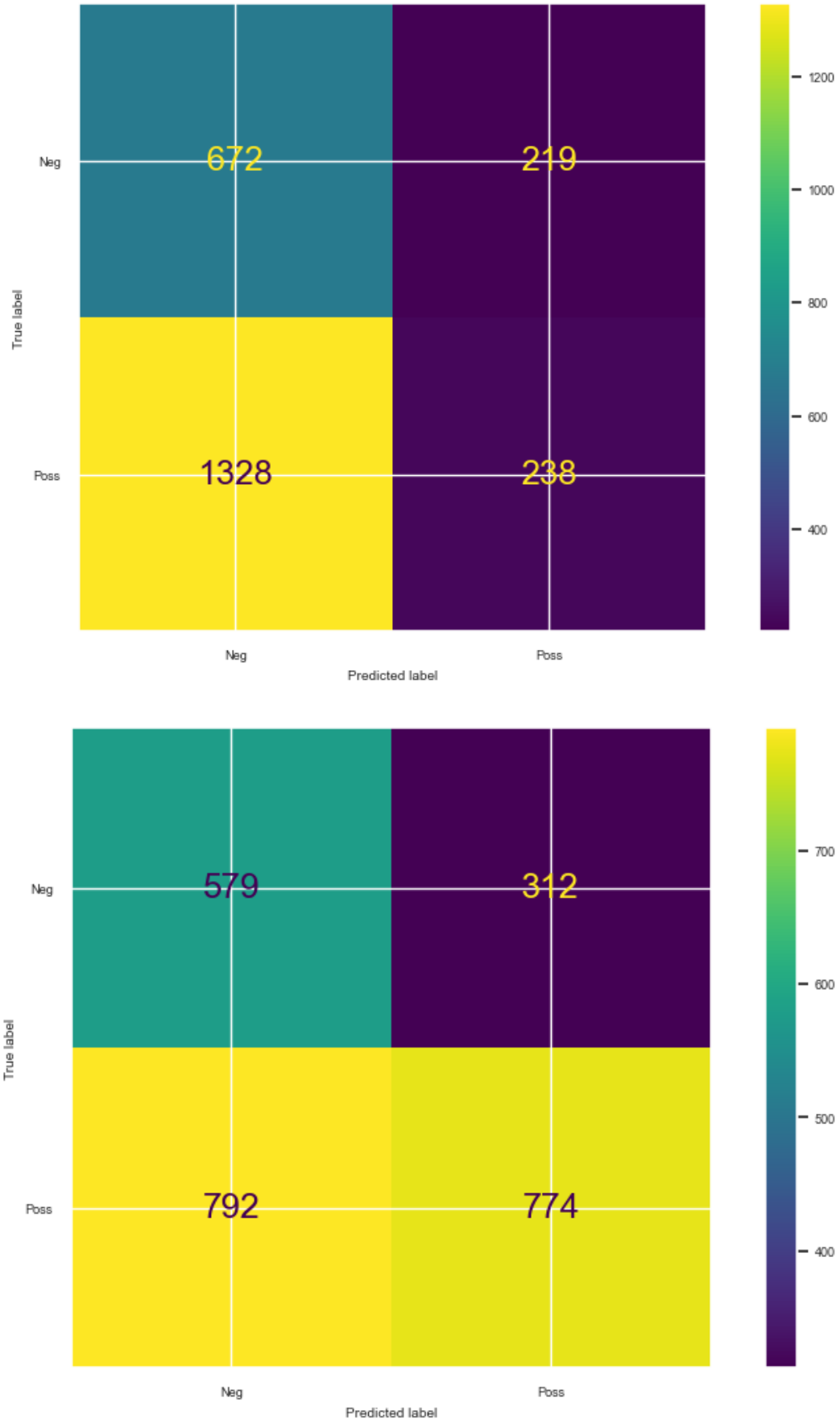


Рис. Є.11. Матриця помилок збалансованого набору даних SMOTEENN FL і SMOTEENN WCE відповідно

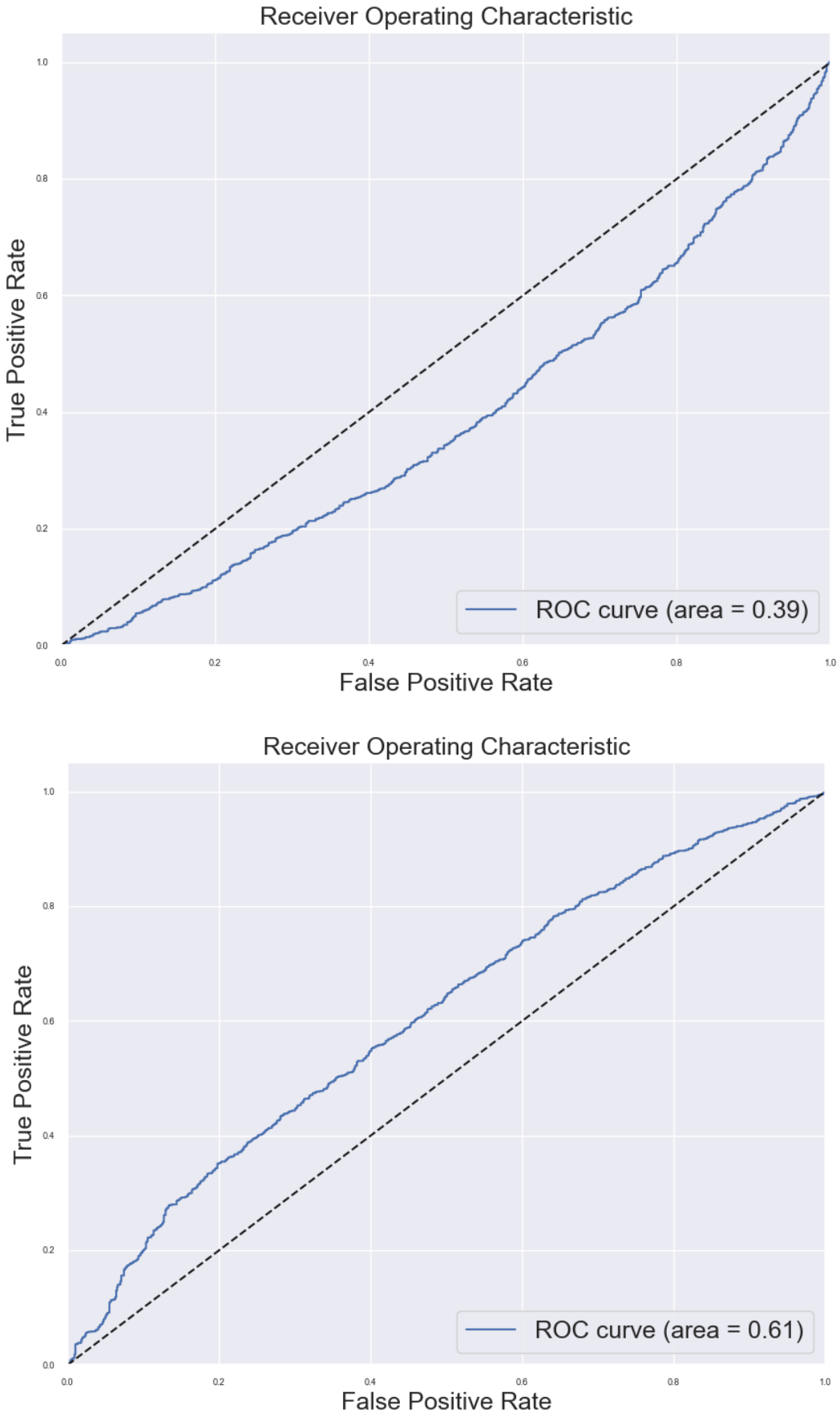


Рис. Є.12. Графік ROC-кривої збалансованого набору даних SMOTEENN FL і SMOTEENN WCE відповідно

ДОДАТОК Ж

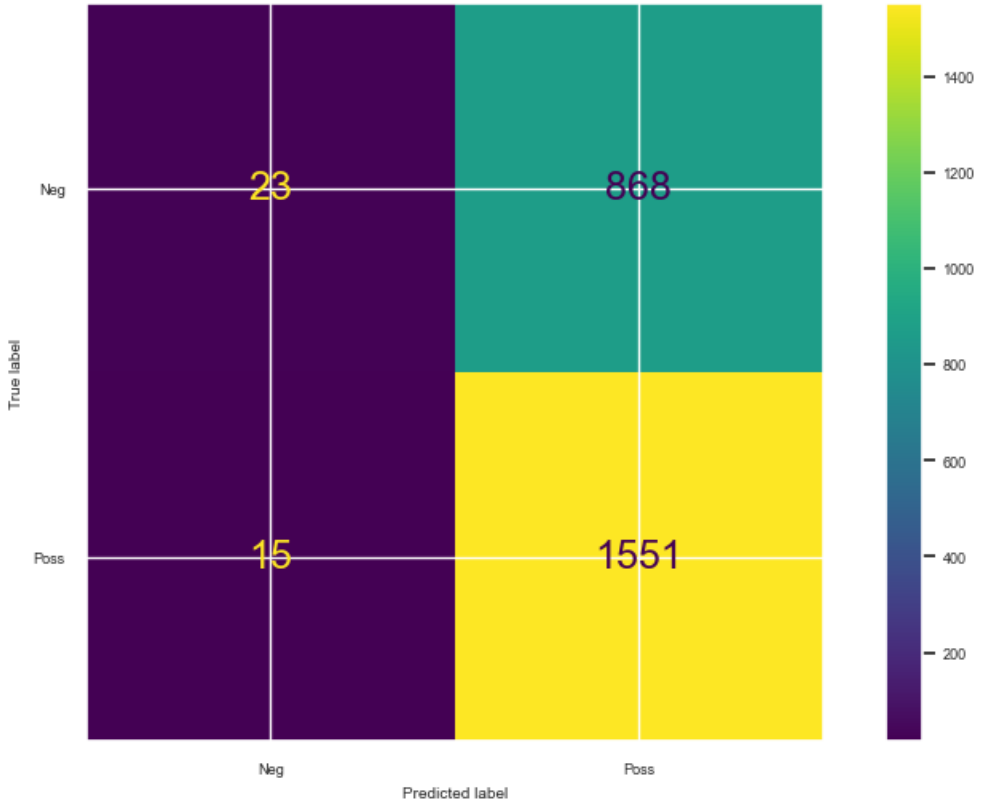


Рис. Ж.1. Матриця помилок збалансованого набору даних WCE

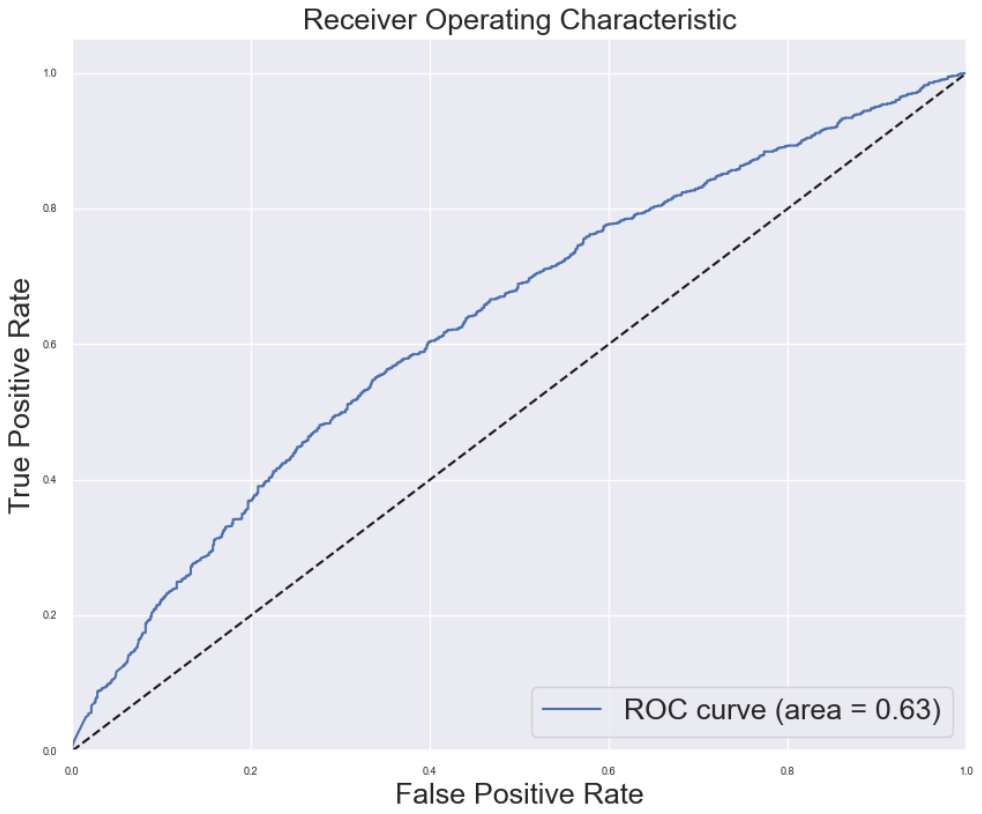


Рис. Ж.2. Графік ROC-кривої збалансованого набору даних WCE

ДОДАТОК З. Код програми

```
# %%
#Plot
import seaborn as sns
import matplotlib.pyplot as plt
import hvplot.pandas
from sklearn.manifold import TSNE

#Math
import pandas as pd
import numpy as np
import math

#For VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

#Instruments for data encoding
from sklearn.preprocessing import LabelEncoder

#Warnings ignore
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)

#Classification
import xgboost as xgb
from sklearn.metrics import accuracy_score,confusion_matrix, f1_score,
precision_score, recall_score,ConfusionMatrixDisplay
from imblearn.metrics import geometric_mean_score
from sklearn.preprocessing import StandardScaler
import skfuzzy as fuzz
import optuna

# %% [markdown]
# *2017 train set 2018 test set*

# %% [markdown]
# **IDA**

# %%
df = pd.DataFrame(pd.read_csv("D:\lab works\ДИПЛОМКА\diploma\loan_applic.csv"))
df = df.drop('Application Number',axis=1)
```

```
    df = df.drop('Application: BPA Broker Negotiation',axis=1)#Відкидуємо колонку, бо вона пуста

    custom_color = "#4e89ae"
    df.drop_duplicates()
    df.head(5)

    # %%
    df.info()

    # %% [markdown]
    # Фікс форматування

    # %%
    columns = ['Average Monthly Sales','Avg Daily Bank Balance'
                    ,'Volume. 4 Months Ago','Volume. 6 Months Ago','Volume. Three Months Ago','Yearly Total Sales']
    for i in columns:
        df[i] = df[i].apply(lambda x: float(x.split()[0].replace(",","") if type(x) is str else x))

    # %%
    df['Application: Close Date'] = pd.to_datetime(df['Application: Close Date'])
    df = df.set_index('Application: Close Date')

    # %%
    df['Application: Origination Fee'] = np.where(df['Application: Origination Fee'] == 3.00, 0, np.where(df['Application: Origination Fee'] == 2.50, 1, 2))

    # %%
    def missing_values(data):

        sns.set(rc={'figure.figsize':(10,6)})
        missed = pd.DataFrame()
        missed['column'] = data.columns

        missed['percent'] = [round(100* data[col].isnull().sum() / len(data), 2) for col in data.columns]
        missed = missed.sort_values('percent',ascending=False)
        missed = missed[missed['percent']>0]

        ax = sns.barplot(
            x=missed['percent'],
            y=missed["column"],
```

```
            orientation='horizontal',color=custom_color
        ).set_title('Percentage of missing values present in the dataset')
        plt.show()


    # %%
    pd.DataFrame(round(df[df.columns].describe().T,2))

    # %%
    train = df[pd.to_datetime('1/3/2017'):pd.to_datetime('9/4/2018')]
    test = df[pd.to_datetime('9/4/2018'):]

    # %%
    if train.isna().sum().sum()==0:
        print('\nNo missing values were present!')
    else:
        print('\nMissing values were present!')
        miss_df=pd.DataFrame(train.isna().sum()).reset_index()
        miss_df.columns=['Columns','Count']
        miss_df['Percentage(%)']=round(((miss_df['Count']/train.shape[0])*100),2)
        missing_values(train)
        plt.show()
    miss_df.sort_values(ascending=False,by='Count')

    # %% [markdown]
    # Очистка даних

    # %%
    def imputatiom_strategy(data):

        for i in data.select_dtypes(include=['int32', 'float64']):
            data[i].fillna(data[i].median(),inplace=True)

        data["crime_record"] = data["crime_record"].fillna('Missing')
              data["Primary    Contact    Gender"]    =    data["Primary    Contact
Gender"].fillna('Missing')

        import random
        for col in data.columns:
            if data[col].dtype == 'object':
                unique_values = data[col].dropna().unique()
                data[col].fillna(random.choice(unique_values), inplace=True)

        Q1 = data['customer Age'].quantile(0.25)
```

```
        Q3 = data['customer Age'].quantile(0.75)
        IQR = Q3. Q1

        lower_bound = Q1. 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        data = data[(data['customer Age'] >= lower_bound) & (data['customer Age'] <=
upper_bound)]

        return data
    train = imputatiom_strategy(train)

    # %%
    try:
        missing_values(train)
    except:
        print('No missing values')

    # %% [markdown]
    # **EDA**

    # %%
    class_counts=train['Outcome'].value_counts()
    sns.countplot(x='Outcome', data=train,color=custom_color)

    plt.title('Outcome')
    plt.xlabel('Classes')
    plt.ylabel('Count')
    plt.show()
    class_counts

    # %%
    imbalance                                                                    =
len(train.loc[train['Outcome']=='Pos'])//len(train.loc[train['Outcome']=='Neg'])
    imbalance

    # %%
    corr=train.corr(method='kendall')

    mask = np.triu(np.ones_like(corr, dtype=bool))

    plt.subplots(figsize=(11, 9))

    cmap = sns.diverging_palette(300, 20, as_cmap=True)
```

```
    sns.set(font_scale=0.6)

    sns.heatmap(corr,mask=mask, cmap=cmap, vmax=.3,center=0,
                linewidths=.5, cbar_kws={"shrink": .5},annot=True)

    plt.show()

    # %%
    sns.scatterplot(data=train,        x="Application:        Funded        Amount",
y="Amount",color=custom_color)
    plt.title("Scatterplot of Application: Funded Amount vs Amount")
    plt.xlabel("Application: Funded Amount")
    plt.ylabel("Amount")
    plt.show()

    # %%
    sns.scatterplot(data=train,   x="Average   Monthly   Sales",   y="Yearly   Total
Sales",color=custom_color)
    plt.title("Scatterplot of Yearly Total Sales vs Average Monthly Sales")
    plt.xlabel("Yearly Total Sales")
    plt.ylabel("Average Monthly Sales")
    plt.show()


    # %%
    sns.scatterplot(data=train, x="Months", y="Days",color=custom_color)
    plt.title("Scatterplot of Months vs Days")
    plt.xlabel("Months")
    plt.ylabel("Days")
    plt.show()

    # %%
    sns.scatterplot(data=train,     x="Satisfactory",     y="Number     of     Trade
Lines",color=custom_color)
    plt.title("Scatterplot of Satisfactory vs Number of Trade Lines")
    plt.xlabel("Satisfactory")
    plt.ylabel("Number of Trade Lines")
    plt.show()

    # %%
    def vifResults(df_numeric_values):
        vif = pd.DataFrame()
         vif["VIF Factor"] = [variance_inflation_factor(df_numeric_values.values, i)
for i in range(df_numeric_values.shape[1])]
```

```
        vif["features"] = df_numeric_values.columns
        return vif.sort_values(ascending=False,by='VIF Factor')

    vifResults(train.select_dtypes(include=['int32','float64'])).head(15)

    # %%
    plt.figure(figsize=(12, 6))
    sns.countplot(x='Office Space', hue='Outcome', data=train)
    plt.title('Office Space by Outcome')
    plt.xlabel('Office Space')
    plt.ylabel('Count')
    plt.legend(title='Outcome')
    plt.show()

    # %%
    pos            =            train.loc[train['Outcome']=='Pos',            'Public
Records'].value_counts().hvplot.bar()
    neg            =            train.loc[train['Outcome']=='Neg',            'Public
Records'].value_counts().hvplot.bar()

    (pos*neg).opts(
        title='Outcome by The Number of public records', xlabel='Number of public
records', ylabel='Count',
        width=400, height=400, legend_cols=2, legend_position='top_right'
    )

    # %%
    sales = train.hvplot.hist(
        y='Yearly Total Sales', by='Outcome', alpha=0.3, width=350, height=400,
          title="Outcome  by  Yearly  Total  Sales",  xlabel='Yearly  Total  Sales',
ylabel='Counts',
        legend='top'
    )
    sales

    # %%
    more30k = train[train['Yearly Total Sales'] <= 300000].hvplot.hist(
          y='Yearly  Total  Sales',  by='Outcome',  bins=50,  alpha=0.3,  width=500,
height=400,
        title="Outcome by Yearly Total Sales (<= 300000/Year)",
        xlabel='Yearly Total Sales', ylabel='Counts', legend='top'
    ).opts(xrotation=45)
    more30k
```

```
    # %%
    train.loc[train['Yearly Total Sales'] >= 2000000, 'Outcome'].value_counts()

    # %%
    satisfactory = train.hvplot.hist(
        y='Satisfactory', by='Outcome', alpha=0.3, width=350, height=400,
           title="Satisfactory by Yearly Total Sales", xlabel='Satisfactory',
ylabel='Outcome',
        legend='top'
    )
    satisfactory

    # %%
    positive_returns_by_age,negative_returns_by_age = train[train['Outcome'] ==
'Pos']['customer Age'].value_counts(),train[train['Outcome'] == 'Neg']['customer
Age'].value_counts()

    positiv = plt.bar(positive_returns_by_age.index, positive_returns_by_age.values)
    negativ = plt.bar(negative_returns_by_age.index, negative_returns_by_age.values)
    plt.title('Pos/Neg loan')
    plt.xlabel('Age')
    plt.ylabel('Count')
    plt.legend([positiv, negativ], ['Positive Loans', 'Negative Loans'])

    positiv + negativ

    plt.show()

    # %%
    crosstab = pd.crosstab(train["Outcome"],train["crime_record"])
    crosstab

    # %%
    crosstab = pd.crosstab(train["Outcome"],train["Has Website"])
    crosstab

    # %%
    train['Outcome'] = train.Outcome.map({'Pos':1, 'Neg':0})

    # %%
    train.corr()['Outcome'].drop('Outcome').sort_values().hvplot.barh(
        width=800, height=400,
        title="Correlation between loan status and numeric features",
        ylabel='Correlation', xlabel='Numerical Features',
```

```
)

# %%
unrepaid_loans = train[train['Outcome'] == 0].groupby('Industry').size()

unrepaid_loans_sorted = unrepaid_loans.sort_values(ascending=False)
top30_unrepaid = unrepaid_loans_sorted.head(10)
top30_unrepaid.plot.barh(figsize=(10, 5))

plt.title('Top 10 Industries with Most Unrepaid Loans')
plt.ylabel('Industry')
plt.xlabel('Number of Unrepaid Loans')

plt.gca().invert_yaxis()

plt.show()

# %%
grouped_data = train.groupby(['Industry', 'Outcome']).size().reset_index(name='count')

pivot_data = grouped_data.pivot_table(index='Industry', columns='Outcome', values='count', fill_value=0)
pivot_data.sort_values(by=0,ascending=True).head(10)

# %%
average_income = train.groupby('Industry')['Yearly Total Sales'].mean()
top30_income = average_income[top30_unrepaid.index]
top30_income

# %%
viz_data = pd.DataFrame({'Industry': top30_income.index, 'Average Yearly Total Sales': top30_income.values})

plt.figure(figsize=(10,8))
sns.barplot(y='Industry', x='Average Yearly Total Sales', data=viz_data)
plt.xlabel('Average Yearly Total Sales')
plt.ylabel('Industry')
plt.title('Average Yearly Total Sales by Industry')
plt.show()

# %%
grouped_data = train.groupby(['Industry', 'Outcome']).size().reset_index(name='count')
```

```
    pivot_data = grouped_data.pivot_table(index='Industry', columns='Outcome',
values='count', fill_value=0)

    pivot_data['repaid_percentage'] = pivot_data[1] / (pivot_data[0] +
pivot_data[1]) * 100

    sorted_data = pivot_data.sort_values(by='repaid_percentage', ascending=True)
    sorted_data.head(10)

    # %%
    import klib

    # %%
    klib.dist_plot(train.select_dtypes(include=['int32','float64']),showall=True)

    # %%
    ax = sns.countplot(x='Type', data=train)
    plt.xticks(rotation=45)

    for p in ax.patches:
        ax.annotate(format(p.get_height(), '.0f'),
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha = 'center',
                    va = 'center',
                    xytext = (0, 10),
                    textcoords = 'offset points')
    plt.show()

    # %%
    print('\nDataset shape:',train.shape)
    for i in train.select_dtypes(include=['int32','float64']):

         print('\nMin and Max Value range of {}: '.format(i),train[i].min(),' to
',train[i].max())
        print('Mean value of {}: '.format(i),train[i].mean())
    print('\n')
    for i in train.select_dtypes(include=['object']):
        print('\nUnique values in {}: '.format(i),train[i].nunique())
              if train[i].nunique()<10: print('Unique values of {}:
'.format(i),train[i].unique())

    # %%
    print('\nTrain')
```

```
    application_vars = [var for var in train.columns if var.startswith("Application") and train[var].nunique() < 15]

    for var in application_vars:
        unique_values = train[var].nunique()
        print(f"{var}, unique values: {unique_values}")


    application_vars = [var for var in test.columns if var.startswith("Application") and test[var].nunique() < 15]
    print('\nTest')
    for var in application_vars:
        unique_values = test[var].nunique()
        print(f"{var}, unique values: {unique_values}")

    # %%
    train['Application: Origination Fee'] = np.where(train['Application: Origination Fee'] == 3.00, 0, np.where(train['Application: Origination Fee'] == 2.50, 1, 2))
    test['Application: Origination Fee'] = np.where(test['Application: Origination Fee'] == 3.00, 0, np.where(test['Application: Origination Fee'] == 2.50, 1, 2))

    # %%
    us_states = ['AL', 'AK', 'AZ', 'AR', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA',
                 'HI', 'ID', 'IL', 'IN', 'IA', 'KS', 'KY', 'LA', 'ME', 'MD',
                 'MA', 'MI', 'MN', 'MS', 'MO', 'MT', 'NE', 'NV', 'NH', 'NJ',
                 'NM', 'NY', 'NC', 'ND', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC',
                 'SD', 'TN', 'TX', 'UT', 'VT', 'VA', 'WA', 'WV', 'WI', 'WY']
    train = train[train['Shipping State'].isin(us_states)]
    test = test[test['Shipping State'].isin(us_states)]

    # %% [markdown]
    # Violinplot

    # %%
    threshold = 24
    without_cat = [col for col in train.columns if train[col].nunique() > threshold]
    categorical_data = train[without_cat].select_dtypes(exclude=['object', 'category'])

    n = len(categorical_data.columns)
    half_n = n // 2
    columns_1 = categorical_data.columns[:half_n]
    columns_2 = categorical_data.columns[half_n:]

    fig, axs = plt.subplots(nrows=1, ncols=half_n, figsize=(15, 5))
```

```
for i, column in enumerate(columns_1):
    if i < len(axs):
        sns.violinplot(train[column], ax=axs[i])
        axs[i].set_title(column)


fig, axs = plt.subplots(nrows=1, ncols=half_n, figsize=(15, 5))

for i, column in enumerate(columns_2):
    if i < len(axs):
        sns.violinplot(train[column], ax=axs[i])
        axs[i].set_title(column)


plt.show()

# %% [markdown]
# KDE

# %%
columns=train.select_dtypes(include=['int32','float64']).columns.to_list()
n = len(columns)
half_n = n // 2
columns_1 = columns[:half_n]
columns_2 = columns[half_n:]

fig, axs = plt.subplots(nrows=1, ncols=half_n, figsize=(15, 5))

for i, column in enumerate(columns_1):
    if i < len(axs):
        sns.kdeplot(train[column], ax=axs[i])
        axs[i].set_title(column)
        plt.xticks(rotation=45)

fig, axs = plt.subplots(nrows=1, ncols=half_n, figsize=(15, 5))
for i, column in enumerate(columns_2):
    if i < len(axs):
        sns.kdeplot(train[column], ax=axs[i])
        axs[i].set_title(column)
        plt.xticks(rotation=45)

plt.show()
```

```
# %%
vifResults(train.select_dtypes(include=['int32','float64'])).head(10)

# %%
train = train.drop(['Yearly Total Sales','Months','Factor Rate','Amount','Brokers submitted last 1 month'],axis=1)

# %%
test = test[train.columns]

# %%
vifResults(train.select_dtypes(include=['int32','float64'])).head(10)

# %%
industry_return_rate = train.groupby('Industry')['Outcome'].mean()

threshold = 0.5

risk_df = pd.DataFrame(industry_return_rate)
risk_df['Risk'] = risk_df['Outcome'].apply(lambda x: 1 if x < threshold else 0)

train['Risk'] = train['Industry'].map(risk_df['Risk'])
test['Risk'] = test['Industry'].map(risk_df['Risk']).fillna(0)

# %%
train = train.drop(['Industry'],axis=1)
test = test.drop(['Industry'],axis=1)

# %% [markdown]
# **Кодування категоріальних змінних**

# %%
bins = [0, 18, 25, 35, 45, 55, 65, 100]
labels = ['<18', '18-24', '25-34', '35-44', '45-54', '55-64', '65+']
train['age_group'] = pd.cut(train['customer Age'], bins=bins, labels=labels, right=False)
test['age_group'] = pd.cut(test['customer Age'], bins=bins, labels=labels, right=False)

# %%
def labelEncoding(data,columns):
    le = LabelEncoder()
    data[columns] = data[columns].apply(le.fit_transform)
    return data
```

```
train['age_group'] = train['age_group'].astype(str)
test['age_group'] = test['age_group'].astype(str)

train['age_group'].replace('nan', 'n', inplace=True)
test['age_group'].replace('nan', 'n', inplace=True)


le = LabelEncoder()
le.fit(['<18', '18-24', '25-34', '35-44', '45-54', '55-64', '65+', 'n'])
train['age_group_enc'] = le.transform(train['age_group'])
test['age_group_enc'] = le.transform(test['age_group'])


train = labelEncoding(train,train.select_dtypes(include='object').columns)
test = labelEncoding(test,test.select_dtypes(include='object').columns)
train = train.drop(['customer Age'],axis=1)
test = test.drop(['customer Age'],axis=1)

# %%
X_train,y_train = train.drop('Outcome',axis=1),train['Outcome']
X_test,y_test = test.drop('Outcome',axis=1),test['Outcome']

# %% [markdown]
# **Методи балансування**

# %% [markdown]
#         Undersampling
#     Random Under Samplimng

# %%
from imblearn.under_sampling import RandomUnderSampler
def random_under_sampling(X_train,y_train):
    rus = RandomUnderSampler(random_state=42)
    X_resampled, y_resampled = rus.fit_resample(X_train,y_train)
    print(y_resampled.value_counts())
    return X_resampled, y_resampled

# %% [markdown]
#
#         Oversampling
#     ROS
#     SMOTE
#     ADASYN
```

```
#

# %%
from imblearn.over_sampling  import RandomOverSampler
def random_over_sampling(X_train,y_train):
    ros = RandomOverSampler(random_state=42)
    X_resampled, y_resampled = ros.fit_resample(X_train,y_train)
    print(y_resampled.value_counts())
    return X_resampled, y_resampled

from imblearn.over_sampling import SMOTE
def smote(X, y):
    smote = SMOTE(random_state=13)
    X_resampled, y_resampled = smote.fit_resample(X, y)
    print(y_resampled.value_counts())
    return X_resampled, y_resampled


from imblearn.over_sampling import ADASYN
def adasyn(X, y):
    ada = ADASYN(random_state=13)
    X_resampled, y_resampled = ada.fit_resample(X, y)
    print(y_resampled.value_counts())
    return X_resampled,y_resampled

# %% [markdown]
#         Hybrid
#     SMOTETOMEK
#     SMOTEENN

# %%
from imblearn.combine import SMOTETomek
def smotetomek(X,y):
    smote_tomek = SMOTETomek(random_state=13)
    X_resampled, y_resampled = smote_tomek.fit_resample(X, y)
    print(y_resampled.value_counts())
    return X_resampled, y_resampled

from imblearn.combine import SMOTEENN
def smoteenn(X,y):
    smote_enn = SMOTEENN(random_state=13)
    X_resampled, y_resampled = smote_enn.fit_resample(X, y)
    print(y_resampled.value_counts())
    return X_resampled, y_resampled
```

```
# %% [markdown]
#     Custom loss functions

# %%
def focal_loss(label: np.ndarray, pred: np.ndarray):
    gamma = 2
    alpha = 0.5

    y_true = label
    y_pred = pred
    y_pred = 1.0 / (1.0 + np.exp(-y_pred))

    grad = -(y_true * np.power((1. y_pred), gamma. 1) * gamma * np.log(y_pred) +
               (1. y_true) * np.power(y_pred, gamma) * ((gamma. 1) * alpha *
np.log(1. y_pred). gamma * np.log(y_pred)))
     hess = ((y_true * np.power((1. y_pred), gamma. 2) * np.power((gamma *
np.log(y_pred). 1), 2)) +
              ((1. y_true) * np.power(y_pred, gamma. 1) * np.power(((gamma. 1) *
alpha * np.log(1. y_pred). gamma * np.log(y_pred)), 2)))
    return grad, hess

def weighted_binary_cross_entropy(labels: np.ndarray, predt: np.ndarray, alpha =
1, ):
            imbalance_alpha     =     len(labels[np.where((labels     ==     1))]
)/len(labels[np.where((labels == 0))] )
    sigmoid_pred = 1.0 / (1.0 + np.exp(-predt))
    grad = -(imbalance_alpha ** labels) * (labels. sigmoid_pred)
    hess = (imbalance_alpha ** labels) * sigmoid_pred * (1.0. sigmoid_pred)

    return grad, hess

# %%
from sklearn.metrics import roc_curve,auc

# %%
def bild(model,y_pred):
   disp1 = ConfusionMatrixDisplay.from_estimator(
      model, X_test, y_test,
      values_format='d',
      display_labels=['Neg', 'Poss']
   )

   for text in disp1.text_:
```

```
        for t in text:
            t.set_fontsize(18)


    y_score = model.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_score)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(10, 8))
    plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate', fontsize=18)
    plt.ylabel('True Positive Rate', fontsize=18)
    plt.title('Receiver Operating Characteristic', fontsize=18)
    plt.legend(loc="lower right", fontsize=18)
    plt.show()


# %% [markdown]
# **Балансування**

# %%
X_balanced_rus,y_balanced_rus = random_under_sampling(X_train, y_train)

X_balanced_ros,y_balanced_ros = random_over_sampling(X_train,y_train)

X_balanced_smote, y_balanced_smote = smote(X_train, y_train)

X_balanced_adasyn, y_balanced_adasyn = adasyn(X_train, y_train)

X_balancedt,y_balancedt = smotetomek(X_train,y_train)

X_balancedn,y_balancedn = smoteenn(X_train,y_train)


# %% [markdown]
# **Підбір гіперпараметрів**

# %%
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 300),
```

```
        'max_depth': trial.suggest_int('max_depth', 3, 10),
        'learning_rate': trial.suggest_loguniform('learning_rate', 1e-4, 1),
        'subsample': trial.suggest_float('subsample', 0.5, 1),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 1),
        'gamma': trial.suggest_float('gamma', 0, 1),
        'min_child_weight': trial.suggest_int('min_child_weight', 1, 10),
    }

    model = xgb.XGBClassifier(**params, random_state=42)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    return accuracy


# %%
# study = optuna.create_study(direction='maximize')
# study.optimize(objective, n_trials=60)
# study.best_params

# %% [markdown]
# 1. Найкращі параметри для несбалансованих / фокал лос функції
# {'n_estimators': 102,
#  'max_depth': 10,
#  'learning_rate': 0.0011966131925523543,
#  'subsample': 0.8006227968027733,
#  'colsample_bytree': 0.9803398728188094,
#  'gamma': 0.8386194241845784,
#  'min_child_weight': 3}
# 2. Найкращі для ROS
# {'n_estimators': 258,
#  'max_depth': 10,
#  'learning_rate': 0.07331865190064507,
#  'subsample': 0.7678677499159607,
#  'colsample_bytree': 0.596916820568302,
#  'gamma': 0.3378097005569924,
#  'min_child_weight': 3}
# 2. Найкращі для RUS
# {'n_estimators': 221,
#  'max_depth': 10,
#  'learning_rate': 0.00041821413923095774,
#  'subsample': 0.8742097017145511,
```

```
    #  'colsample_bytree': 0.5936492805139413,
    #  'gamma': 0.9463373617864398,
    #  'min_child_weight': 7}
    # 2. Найкращі для SMOTE
    # {'n_estimators': 143,
    #  'max_depth': 10,
    #  'learning_rate': 0.0012381122229706693,
    #  'subsample': 0.8764884885259474,
    #  'colsample_bytree': 0.5122076493704834,
    #  'gamma': 0.42260708044796647,
    #  'min_child_weight': 6}
    # 3. Найкращі для ADASYN
    # {'n_estimators': 190,
    #  'max_depth': 7,
    #  'learning_rate': 0.13836053931252873,
    #  'subsample': 0.6016105058143335,
    #  'colsample_bytree': 0.8645099541402509,
    #  'gamma': 0.7162834664094448,
    #  'min_child_weight': 3}
    # 4. Найкращі для SMOTETOMEK
    # {'n_estimators': 255,
    #  'max_depth': 8,
    #  'learning_rate': 0.04101703529289752,
    #  'subsample': 0.6873034967286848,
    #  'colsample_bytree': 0.5498898753584462,
    #  'gamma': 0.9616072598594146,
    #  'min_child_weight': 1}
    # 5. Найкращі для SMOTEENN
    # {'n_estimators': 168,
    #  'max_depth': 6,
    #  'learning_rate': 0.08509848665588302,
    #  'subsample': 0.9239135350691549,
    #  'colsample_bytree': 0.9043052520550187,
    #  'gamma': 0.1889174325488193,
    #  'min_child_weight': 1}

    # %%
    train['flag_Volume.  Three  Months  Ago'] = np.where(train['Volume.  Three  Months
Ago'].isna() == True, 1,0)
    train['flag_Time_Business'] = np.where(train['Time  In  Business  Actual'].isna()
== True, 1,0)

    test['flag_Volume.  Three  Months  Ago'] = np.where(test['Volume.  Three  Months
Ago'].isna() == True, 1,0)
```

```
test['flag_Time_Business'] = np.where(test['Time In Business Actual'].isna() == True, 1,0)


# %% [markdown]
# **Кластеризація**

# %%
from sklearn.metrics import silhouette_score
import skfuzzy as fuzz

max_clusters = 10

silhouette_scores = []

for n_clusters in range(2, max_clusters+1):
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(data=train.T, c=n_clusters, m=2, error=0.005, maxiter=1000)
    labels = np.argmax(u, axis=0)
    silhouette_avg = silhouette_score(train, labels)
    silhouette_scores.append(silhouette_avg)

plt.figure(figsize=(10,5))
plt.plot(range(2, max_clusters+1), silhouette_scores, marker = '.')
plt.title('Silhouette Scores vs Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.grid(True)
plt.show()

# %%
app_data = train[['Application: Buy Rate',
'Application: Funded Amount','Application: Origination Fee', 'Application: Remittance Frequency']]

scaler = StandardScaler()
scaled_features = scaler.fit_transform(app_data)

cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(scaled_features.T, 2, 2, error=0.005, maxiter=1000, init=None)
labels = np.argmax(u, axis=0)

train['Fuzzy_Cluster'] = labels
```

```
tsne = TSNE(n_components=2, random_state=42)
data_2d = tsne.fit_transform(train)

plt.figure(figsize=(10, 8))
plt.scatter(data_2d[:, 0], data_2d[:, 1], c=labels, cmap='viridis', alpha=0.5)

plt.colorbar(label='Cluster Label')
plt.title('t-SNE visualization of clusters')
plt.xlabel('First t-SNE feature')
plt.ylabel('Second t-SNE feature')
plt.grid(True)
plt.show()

# %%
app_data = test[['Application: Buy Rate',
'Application: Funded Amount','Application: Origination Fee', 'Application: Remittance Frequency']]

scaler = StandardScaler()
scaled_data = scaler.fit_transform(app_data)

cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(scaled_data.T, 2, 2, error=0.005, maxiter=1000, init=None)

labels = np.argmax(u, axis=0)

test['Fuzzy_Cluster'] = labels

# %%
train = train.drop(['Application: Buy Rate',
'Application: Funded Amount', 'Application: Origination Fee','Application: Remittance Frequency'], axis = 1)
test= test.drop(['Application: Buy Rate',
'Application: Funded Amount', 'Application: Origination Fee','Application: Remittance Frequency'], axis = 1)

# %% [markdown]
# **Побудова моделей**

# %% [markdown]
# Random Under Sampling

# %%
params = {
```

```
    'n_estimators': 221,
    'max_depth': 10,
    'learning_rate': 0.00041821413923095774,
    'subsample': 0.8742097017145511,
    'colsample_bytree': 0.5936492805139413,
    'gamma': 0.9463373617864398,
    'min_child_weight': 7}
params_focal = {
    'objective': focal_loss,
    'n_estimators': 221,
    'max_depth': 10,
    'learning_rate': 0.00041821413923095774,
    'subsample': 0.8742097017145511,
    'colsample_bytree': 0.5936492805139413,
    'gamma': 0.9463373617864398,
    'min_child_weight': 7,
    'reg_alpha':0.8,
    'reg_lambda':0.9}
params_weighted = {
    'objective': weighted_binary_cross_entropy,
    'n_estimators': 221,
    'max_depth': 10,
    'learning_rate': 0.00041821413923095774,
    'subsample': 0.8742097017145511,
    'colsample_bytree': 0.5936492805139413,
    'gamma': 0.9463373617864398,
    'min_child_weight': 7,
    'reg_alpha':0.8,
    'reg_lambda':0.9}

model = xgb.XGBClassifier(**params)
model.fit(X_balanced_rus,y_balanced_rus)
y_pred = model.predict(X_test)
threshold = 0.3
y_prob = model.predict_proba(X_test)
y_prob = (y_prob[:,1] > threshold).astype(int)

modelrusf = xgb.XGBClassifier(**params_focal)

modelrusf.fit(X_balanced_rus,y_balanced_rus)

y_pred_rusf = modelrusf.predict(X_test)
threshold = 0.3
y_prob_rusf = modelrusf.predict_proba(X_test)
```

```
y_prob_rusf = (y_prob_rusf[:,1] > threshold).astype(int)

modelrusw = xgb.XGBClassifier(**params_weighted)

modelrusw.fit(X_balanced_rus,y_balanced_rus)

y_pred_rusw = modelrusw.predict(X_test)
threshold = 0.3
y_prob_rusw = modelrusw.predict_proba(X_test)
y_prob_rusw = (y_prob_rusw[:,1] > threshold).astype(int)

# %%
result = {'Accuracy Score':accuracy_score(y_test, y_prob),
                'Precision Score':precision_score(y_test,y_pred),
                'Recall Score' :recall_score(y_test,y_pred),
                'F1 Score':f1_score(y_test,y_pred),
                'G-mean' : geometric_mean_score(y_test,y_pred)}

resultf = {'Accuracy Score':accuracy_score(y_test, y_prob_rusf),
                'Precision Score':precision_score(y_test,y_pred_rusf),
                'Recall Score' :recall_score(y_test,y_pred_rusf),
                'F1 Score':f1_score(y_test,y_pred_rusf),
                'G-mean' :  geometric_mean_score(y_test,y_pred_rusf)}

resultw = {'Accuracy Score':accuracy_score(y_test, y_prob_rusw),
                'Precision Score':precision_score(y_test,y_pred_rusw),
                'Recall Score' :recall_score(y_test,y_pred_rusw),
                'F1 Score':f1_score(y_test,y_pred_rusw),
                'G-mean' :  geometric_mean_score(y_test,y_pred_rusw)}

results = pd.DataFrame.from_dict(result,orient='index',columns= ['RUS'])
results =
pd.concat([results,pd.DataFrame.from_dict(resultf,orient='index',columns=  ['RUS
FL'])],axis=1)
results =
pd.concat([results,pd.DataFrame.from_dict(resultw,orient='index',columns=  ['RUS
WCE'])],axis=1)
result


# %%
bild(model,y_pred)

# %%
```

```
bild(modelrusf,y_pred_rusf)

# %%
bild(modelrusw,y_pred_rusw)

# %% [markdown]
# Random Over Sampling

# %%
params = {
  'n_estimators': 258,
  'max_depth': 10,
  'learning_rate': 0.07331865190064507,
  'subsample': 0.7678677499159607,
  'colsample_bytree': 0.596916820568302,
  'gamma': 0.3378097005569924,
  'min_child_weight': 3}
params_focal = {
  'objective': focal_loss,
  'n_estimators': 258,
  'max_depth': 10,
  'learning_rate': 0.07331865190064507,
  'subsample': 0.7678677499159607,
  'colsample_bytree': 0.596916820568302,
  'gamma': 0.3378097005569924,
  'min_child_weight': 3,
  'reg_alpha':0.8,
  'reg_lambda':0.9}
params_weighted = {
  'objective': weighted_binary_cross_entropy,
  'n_estimators': 258,
  'max_depth': 10,
  'learning_rate': 0.07331865190064507,
  'subsample': 0.7678677499159607,
  'colsample_bytree': 0.596916820568302,
  'gamma': 0.3378097005569924,
  'min_child_weight': 3,
  'reg_alpha':0.8,
  'reg_lambda':0.9}

model = xgb.XGBClassifier(**params)
model.fit(X_balanced_ros,y_balanced_ros)
y_pred = model.predict(X_test)
threshold = 0.3
```

```
    y_prob = model.predict_proba(X_test)
    y_prob = (y_prob[:,1] > threshold).astype(int)

    modelrosf = xgb.XGBClassifier(**params_focal)

    modelrosf.fit(X_balanced_ros,y_balanced_ros)
    y_pred_rosf = modelrosf.predict(X_test)
    threshold = 0.3
    y_prob_rosf = modelrosf.predict_proba(X_test)
    y_prob_rosf = (y_prob_rosf[:,1] > threshold).astype(int)

    modelrosw = xgb.XGBClassifier(**params_weighted)

    modelrosw.fit(X_balanced_ros,y_balanced_ros)
    y_pred_rosw = modelrosw.predict(X_test)
    threshold = 0.3
    y_prob_rosw = modelrosw.predict_proba(X_test)
    y_prob_rosw = (y_prob_rosw[:,1] > threshold).astype(int)

    # %%
    result = {'Accuracy Score':accuracy_score(y_test, y_prob),
                    'Precision Score':precision_score(y_test,y_pred),
                    'Recall Score' :recall_score(y_test,y_pred),
                    'F1 Score':f1_score(y_test,y_pred),
                    'G-mean' : geometric_mean_score(y_test,y_pred)}

    resultf = {'Accuracy Score':accuracy_score(y_test, y_prob_rosf),
                    'Precision Score':precision_score(y_test,y_pred_rosf),
                    'Recall Score' :recall_score(y_test,y_pred_rosf),
                    'F1 Score':f1_score(y_test,y_pred_rosf),
                    'G-mean' : geometric_mean_score(y_test,y_pred_rosf)}
    resultw = {'Accuracy Score':accuracy_score(y_test, y_prob_rosw),
                    'Precision Score':precision_score(y_test,y_pred_rosw),
                    'Recall Score' :recall_score(y_test,y_pred_rosw),
                    'F1 Score':f1_score(y_test,y_pred_rosw),
                    'G-mean' : geometric_mean_score(y_test,y_pred_rosw)}

    results =
pd.concat([results,pd.DataFrame.from_dict(result,orient='index',columns=
['ROS'])],axis=1)
    results =
pd.concat([results,pd.DataFrame.from_dict(resultf,orient='index',columns=['ROS
FL'])],axis=1)
```

```
    results = pd.concat([results,pd.DataFrame.from_dict(resultw,orient='index',columns=['ROS WCE'])],axis=1)
    result

    # %%
    bild(model,y_pred)

    # %%
    bild(modelrosf,y_pred_rosf)

    # %%
    bild(modelrosw,y_pred_rosw)

    # %% [markdown]
    # SMOTE

    # %%
    params = {
        'n_estimators': 143,
        'max_depth': 10,
        'learning_rate': 0.0012381122229706693,
        'subsample': 0.8764884885259474,
        'colsample_bytree': 0.5122076493704834,
        'gamma': 0.42260708044796647,
        'min_child_weight': 6,
        'reg_alpha':0.8,
        'reg_lambda':0.9}
    params_focal = {
        'objective': focal_loss,
        'n_estimators': 143,
        'max_depth': 10,
        'learning_rate': 0.0012381122229706693,
        'subsample': 0.8764884885259474,
        'colsample_bytree': 0.5122076493704834,
        'gamma': 0.42260708044796647,
        'min_child_weight': 6,
        'reg_alpha':0.8,
        'reg_lambda':0.9}
    params_weighted = {
        'objective': weighted_binary_cross_entropy,
        'n_estimators': 143,
        'max_depth': 10,
        'learning_rate': 0.0012381122229706693,
```

```
    'subsample': 0.8764884885259474,
    'colsample_bytree': 0.5122076493704834,
    'gamma': 0.42260708044796647,
    'min_child_weight': 6,
    'reg_alpha':0.8,
    'reg_lambda':0.9}

model = xgb.XGBClassifier(**params)
model.fit(X_balanced_smote,y_balanced_smote)
y_pred = model.predict(X_test)
threshold = 0.3
y_prob = model.predict_proba(X_test)
y_prob = (y_prob[:,1] > threshold).astype(int)

modelsmotef = xgb.XGBClassifier(**params_focal)

modelsmotef.fit(X_balanced_smote,y_balanced_smote)
y_pred_smotef = modelsmotef.predict(X_test)
threshold = 0.3
y_prob_smotef = modelsmotef.predict_proba(X_test)
y_prob_smotef = (y_prob_smotef[:,1] > threshold).astype(int)

modelsmotew = xgb.XGBClassifier(**params_weighted)

modelsmotew.fit(X_balanced_smote,y_balanced_smote)
y_pred_smotew = modelsmotew.predict(X_test)
threshold = 0.3
y_prob_smotew = modelsmotew.predict_proba(X_test)
y_prob_smotew = (y_prob_smotew[:,1] > threshold).astype(int)

# %%
result = {'Accuracy Score':accuracy_score(y_test, y_prob),
                'Precision Score':precision_score(y_test,y_pred),
                'Recall Score' :recall_score(y_test,y_pred),
                'F1 Score':f1_score(y_test,y_pred),
                'G-mean' : geometric_mean_score(y_test,y_pred)}

resultf = {'Accuracy Score':accuracy_score(y_test, y_prob_smotef),
                'Precision Score':precision_score(y_test,y_pred_smotef),
                'Recall Score' :recall_score(y_test,y_pred_smotef),
                'F1 Score':f1_score(y_test,y_pred_smotef),
                'G-mean' : geometric_mean_score(y_test,y_pred_smotef)}
resultw = {'Accuracy Score':accuracy_score(y_test, y_prob_smotew),
                'Precision Score':precision_score(y_test,y_pred_smotew),
```

```
                    'Recall Score' :recall_score(y_test,y_pred_smotew),
                    'F1 Score':f1_score(y_test,y_pred_smotew),
                    'G-mean' : geometric_mean_score(y_test,y_pred_smotew)}

    results = pd.concat([results,pd.DataFrame.from_dict(result,orient='index',columns=['SMOTE'])],axis=1)
    results = pd.concat([results,pd.DataFrame.from_dict(resultf,orient='index',columns=['SMOTE FL'])],axis=1)
    results = pd.concat([results,pd.DataFrame.from_dict(resultw,orient='index',columns=['SMOTE WCE'])],axis=1)
    result

    # %%
    bild(model,y_pred)

    # %%
    bild(modelsmotef,y_pred_smotef)

    # %%
    bild(modelsmotew,y_pred_smotew)

    # %% [markdown]
    # ADASYN

    # %%
    params = {
        'n_estimators': 190,
        'max_depth': 7,
        'learning_rate': 0.13836053931252873,
        'subsample': 0.6016105058143335,
        'colsample_bytree': 0.8645099541402509,
        'gamma': 0.7162834664094448,
        'min_child_weight': 3,
        'reg_alpha':0.8,
        'reg_lambda':0.9}
    params_focal = {
        'objective': focal_loss,
        'n_estimators': 190,
        'max_depth': 7,
        'learning_rate': 0.13836053931252873,
        'subsample': 0.6016105058143335,
```

```
    'colsample_bytree': 0.8645099541402509,
    'gamma': 0.7162834664094448,
    'min_child_weight': 3,
    'reg_alpha':0.8,
    'reg_lambda':0.9}
params_weighted = {
    'objective': weighted_binary_cross_entropy,
    'n_estimators': 190,
    'max_depth': 7,
    'learning_rate': 0.13836053931252873,
    'subsample': 0.6016105058143335,
    'colsample_bytree': 0.8645099541402509,
    'gamma': 0.7162834664094448,
    'min_child_weight': 3,
    'reg_alpha':0.8,
    'reg_lambda':0.9}

model = xgb.XGBClassifier(**params)
model.fit(X_balanced_adasyn,y_balanced_adasyn)
y_pred = model.predict(X_test)
threshold = 0.3
y_prob = model.predict_proba(X_test)
y_prob = (y_prob[:,1] > threshold).astype(int)

modeladasynf = xgb.XGBClassifier(**params_focal)

modeladasynf.fit(X_balanced_adasyn,y_balanced_adasyn)
y_pred_adasynf = modeladasynf.predict(X_test)
threshold = 0.3
y_prob_adasynf = modeladasynf.predict_proba(X_test)
y_prob_adasynf = (y_prob_adasynf[:,1] > threshold).astype(int)

modeladasynw = xgb.XGBClassifier(**params_weighted)

modeladasynw.fit(X_balanced_adasyn,y_balanced_adasyn)
y_pred_adasynw = modeladasynw.predict(X_test)
threshold = 0.3
y_prob_adasynw = modeladasynw.predict_proba(X_test)
y_prob_adasynw = (y_prob_adasynw[:,1] > threshold).astype(int)

# %%
result = {'Accuracy Score':accuracy_score(y_test, y_prob),
                'Precision Score':precision_score(y_test,y_pred),
                'Recall Score' :recall_score(y_test,y_pred),
```

```
                    'F1 Score':f1_score(y_test,y_pred),
                    'G-mean' : geometric_mean_score(y_test,y_pred)}

    resultf = {'Accuracy Score':accuracy_score(y_test, y_prob_adasynf),
                    'Precision Score':precision_score(y_test,y_pred_adasynf),
                    'Recall Score' :recall_score(y_test,y_pred_adasynf),
                    'F1 Score':f1_score(y_test,y_pred_adasynf),
                    'G-mean' : geometric_mean_score(y_test,y_pred_adasynf)}

    resultw = {'Accuracy Score':accuracy_score(y_test, y_prob_adasynw),
                    'Precision Score':precision_score(y_test,y_pred_adasynw),
                    'Recall Score' :recall_score(y_test,y_pred_adasynw),
                    'F1 Score':f1_score(y_test,y_pred_adasynw),
                    'G-mean' : geometric_mean_score(y_test,y_pred_adasynw)}

    results                                                                 =
pd.concat([results,pd.DataFrame.from_dict(result,orient='index',columns=
['ADASYN'])],axis=1)
    results                                                                 =
pd.concat([results,pd.DataFrame.from_dict(resultf,orient='index',columns=    ['ADASYN
FL'])],axis=1)
    results                                                                 =
pd.concat([results,pd.DataFrame.from_dict(resultw,orient='index',columns=    ['ADASYN
WCE'])],axis=1)
    result

    # %%
    bild(model,y_pred)

    # %%
    bild(modeladasynf,y_pred_adasynf)

    # %%
    bild(modeladasynw,y_pred_adasynf)

    # %% [markdown]
    # SMOTETOMEK

    # %%
    params = {
        'max_depth': 10,
        'n_estimators': 150,
        'learning_rate': 0.5,
        'subsample': 0.8,
```

```
    'colsample_bytree': 0.9,
    'seed': 12,
    'reg_alpha':0.8,
    'reg_lambda':0.9}
params_focal = {
    'objective': focal_loss,
    'max_depth': 10,
    'n_estimators': 150,
    'learning_rate': 0.5,
    'subsample': 0.8,
    'colsample_bytree': 0.9,
    'seed': 12,
    'reg_alpha':0.8,
    'reg_lambda':0.9}
params_weighted = {
    'objective': weighted_binary_cross_entropy,
    'max_depth': 10,
    'n_estimators': 150,
    'learning_rate': 0.5,
    'subsample': 0.8,
    'colsample_bytree': 0.9,
    'seed': 12,
    'reg_alpha':0.8,
    'reg_lambda':0.9}

model = xgb.XGBClassifier(**params)
model.fit(X_balancedt,y_balancedt)
y_pred = model.predict(X_test)
threshold = 0.3
y_prob = model.predict_proba(X_test)
y_prob = (y_prob[:,1] > threshold).astype(int)

modeltomekf = xgb.XGBClassifier(**params_focal)

modeltomekf.fit(X_balancedt,y_balancedt)
y_pred_tomekf = modeltomekf.predict(X_test)
threshold = 0.3
y_prob_tomekf = modeltomekf.predict_proba(X_test)
y_prob_tomekf = (y_prob_tomekf[:,1] > threshold).astype(int)

modeltomekw = xgb.XGBClassifier(**params_weighted)

modeltomekw.fit(X_balancedt,y_balancedt)
y_pred_tomekw = modeltomekw.predict(X_test)
```

```
threshold = 0.3
y_prob_tomekw = modeltomekw.predict_proba(X_test)
y_prob_tomekw = (y_prob_tomekw[:,1] > threshold).astype(int)

# %%
result = {'Accuracy Score':accuracy_score(y_test, y_prob),
                'Precision Score':precision_score(y_test,y_pred),
                'Recall Score' :recall_score(y_test,y_pred),
                'F1 Score':f1_score(y_test,y_pred),
                'G-mean' : geometric_mean_score(y_test,y_pred)}

resultf = {'Accuracy Score':accuracy_score(y_test, y_prob_tomekf),
                'Precision Score':precision_score(y_test,y_pred_tomekf),
                'Recall Score' :recall_score(y_test,y_pred_tomekf),
                'F1 Score':f1_score(y_test,y_pred_tomekf),
                'G-mean' : geometric_mean_score(y_test,y_pred_tomekf)}
resultw = {'Accuracy Score':accuracy_score(y_test, y_prob_tomekw),
                'Precision Score':precision_score(y_test,y_pred_tomekw),
                'Recall Score' :recall_score(y_test,y_pred_tomekw),
                'F1 Score':f1_score(y_test,y_pred_tomekw),
                'G-mean' : geometric_mean_score(y_test,y_pred_tomekw)}

results                                                                    =
pd.concat([results,pd.DataFrame.from_dict(result,orient='index',columns=
['SMOTETOMEK'])],axis=1)
results                                                                    =
pd.concat([results,pd.DataFrame.from_dict(resultf,orient='index',columns=
['SMOTETOMEK FL'])],axis=1)
results                                                                    =
pd.concat([results,pd.DataFrame.from_dict(resultw,orient='index',columns=
['SMOTETOMEK WCE'])],axis=1)
result

# %%
bild(model,y_pred)

# %%
bild(modeltomekf,y_pred_tomekf)

# %%
bild(modeltomekw,y_pred_tomekw)

# %% [markdown]
# SMOTEENN
```

```
# %%
params = {
    'n_estimators': 168,
    'max_depth': 6,
    'learning_rate': 0.08509848665588302,
    'subsample': 0.9239135350691549,
    'colsample_bytree': 0.9043052520550187,
    'gamma': 0.1889174325488193,
    'min_child_weight': 1}
params_focal = {
    'objective': focal_loss,
    'n_estimators': 168,
    'max_depth': 6,
    'learning_rate': 0.08509848665588302,
    'subsample': 0.9239135350691549,
    'colsample_bytree': 0.9043052520550187,
    'gamma': 0.1889174325488193,
    'min_child_weight': 1,
    'reg_alpha':0.8,
    'reg_lambda':0.9}
params_weighted = {
    'objective': weighted_binary_cross_entropy,
    'n_estimators': 168,
    'max_depth': 6,
    'learning_rate': 0.08509848665588302,
    'subsample': 0.9239135350691549,
    'colsample_bytree': 0.9043052520550187,
    'gamma': 0.1889174325488193,
    'min_child_weight': 1,
    'reg_alpha':0.8,
    'reg_lambda':0.9}

model = xgb.XGBClassifier(**params)
model.fit(X_balancedn,y_balancedn)
y_pred = model.predict(X_test)
threshold = 0.3
y_prob = model.predict_proba(X_test)
y_prob = (y_prob[:,1] > threshold).astype(int)

modelsmoteenf = xgb.XGBClassifier(**params_focal)

modelsmoteenf.fit(X_balancedn,y_balancedn)
y_pred_smoteeenf = modelsmoteenf.predict(X_test)
```

```
threshold = 0.3
y_prob_smoteenf = modelsmoteenf.predict_proba(X_test)
y_prob_smoteenf = (y_prob_smoteenf[:,1] > threshold).astype(int)

modelsmoteenw = xgb.XGBClassifier(**params_weighted)

modelsmoteenw.fit(X_balancedn,y_balancedn)
y_pred_smoteeenw = modelsmoteenw.predict(X_test)
threshold = 0.3
y_prob_smoteenw = modelsmoteenw.predict_proba(X_test)
y_prob_smoteenw = (y_prob_smoteenw[:,1] > threshold).astype(int)

# %%
result = {'Accuracy Score':accuracy_score(y_test, y_prob),
                'Precision Score':precision_score(y_test,y_pred),
                'Recall Score' :recall_score(y_test,y_pred),
                'F1 Score':f1_score(y_test,y_pred),
                'G-mean' : geometric_mean_score(y_test,y_pred)}

resultf = {'Accuracy Score':accuracy_score(y_test, y_prob_smoteenf),
                'Precision Score':precision_score(y_test,y_pred_smoteeenf),
                'Recall Score' :recall_score(y_test,y_pred_smoteeenf),
                'F1 Score':f1_score(y_test,y_pred_smoteeenf),
                'G-mean' : geometric_mean_score(y_test,y_pred_smoteeenf)}

resultw = {'Accuracy Score':accuracy_score(y_test, y_prob_smoteenw),
                'Precision Score':precision_score(y_test,y_pred_smoteeenw),
                'Recall Score' :recall_score(y_test,y_pred_smoteeenw),
                'F1 Score':f1_score(y_test,y_pred_smoteeenw),
                'G-mean' : geometric_mean_score(y_test,y_pred_smoteeenw)}

results                                                                    =
pd.concat([results,pd.DataFrame.from_dict(result,orient='index',columns=
['SMOTEENN'])],axis=1)
results                                                                    =
pd.concat([results,pd.DataFrame.from_dict(resultf,orient='index',columns=   ['SMOTEENN
FL'])],axis=1)
results                                                                    =
pd.concat([results,pd.DataFrame.from_dict(resultw,orient='index',columns=   ['SMOTEENN
WCE'])],axis=1)
result

# %%
bild(model,y_pred)
```

```
# %%
bild(modelsmoteenf,y_pred_smoteeenf)

# %%
bild(modelsmoteenw,y_pred_smoteeenw)

# %% [markdown]
# Focal Loss

# %%
params = {
    'objective': focal_loss,
    'n_estimators': 102,
    'max_depth': 10,
    'learning_rate': 0.0011966131925523543,
    'subsample': 0.8006227968027733,
    'colsample_bytree': 0.9803398728188094,
    'gamma': 0.8386194241845784,
    'min_child_weight': 3,
    'reg_alpha':0.8,
    'reg_lambda':0.9}

modelloss = xgb.XGBClassifier(**params)

modelloss.fit(X_train,y_train)
y_pred_loss = modelloss.predict(X_test)
threshold = 0.3
y_prob_loss = modelloss.predict_proba(X_test)
y_prob_loss = (y_prob_loss[:,1] > threshold).astype(int)

# %%
result = {'Accuracy Score':accuracy_score(y_test, y_prob_loss),
                'Precision Score':precision_score(y_test,y_pred_loss),
                'Recall Score' :recall_score(y_test,y_pred_loss),
                'F1 Score':f1_score(y_test,y_pred_loss),
                'G-mean' : geometric_mean_score(y_test,y_pred_loss)}
results =
pd.concat([results,pd.DataFrame.from_dict(result,orient='index',columns=['FOCAL
LOSS'])],axis=1)

# %%
result
```

```
# %%
bild(modelloss,y_pred_loss)

# %% [markdown]
# Weighted Binary Cross Entropy

# %%
params = {
    'objective': weighted_binary_cross_entropy,
    'n_estimators': 102,
    'max_depth': 10,
    'learning_rate': 0.0011966131925523543,
    'subsample': 0.8006227968027733,
    'colsample_bytree': 0.9803398728188094,
    'gamma': 0.8386194241845784,
    'min_child_weight': 3,
    'reg_alpha':0.8,
    'reg_lambda':0.9}

modelent = xgb.XGBClassifier(**params)

modelent.fit(X_train,y_train)
y_pred_ent = modelent.predict(X_test)
threshold = 0.3
y_prob_ent = modelent.predict_proba(X_test)
y_prob_ent = (y_prob_ent[:,1] > threshold).astype(int)


# %%
result = {'Accuracy Score':accuracy_score(y_test, y_prob_ent),
                'Precision Score':precision_score(y_test,y_pred_ent),
                'Recall Score' :recall_score(y_test,y_pred_ent),
                'F1 Score':f1_score(y_test,y_pred_ent),
                'G-mean' : geometric_mean_score(y_test,y_pred_ent)}

results =
pd.concat([results,pd.DataFrame.from_dict(result,orient='index',columns = ['WEIGHTED
BINARY CROSS ENTROPY'])],axis=1)

# %%
result

# %%
bild(modelent,y_pred_ent)
```

```
# %%
results.to_csv('res.csv')
```