Міністерство освіти та науки України
Львівський національний університет імені Івана Франка

КУРСОВА РОБОТА

на тему:

“Ідентифікація аномальних даних при заповненні бази даних”

Виконав:
студент групи ФЕП-24
Олексин В.М
Науковий керівник:
Ляшкевич В.Я

Оцінка «_____»

«_____» _____ 2021 р.

Львів - 2022

АНОТАЦІЇ

Ідентифікація аномалій - це виявлення неочікуваних подій, спостережень або елементів, які суттєво відрізняються від норми. Як приклад аномалії було обрано шахрайство з банківськими картами. В теперішні дні використання кредитних карт різко зросло. Оскільки кредитна карта найпопулярніший спосіб оплати, як для онлайн, так і для звичайних покупок, випадки шахрайства теж зростають. Існує проблема виявити хоч кілька шахрайських транзакцій із мільйонів простих повсякденних транзакцій щодня. Неправильно класифіковані дані можуть бути ще однією серйозною проблемою, оскільки не кожну шахрайську транзакцію виловлюють та повідомляють про неї. Потрібно мати просту та досить швидку модель виявлення шахрайства в реальному часі. Це мета цієї курсової, яка тут реалізується.

Abstract

Identification of anomalies is the detection of unexpected events, observations or elements that are significantly different from the norm. Bank card fraud was chosen as an example of an anomaly. Credit card usage has skyrocketed these days. With credit cards being the most popular form of payment for both online and everyday purchases, fraud is also on the rise. There is a problem to detect a few fraudulent transactions out of millions of geniune transactions everyday. Misclassified data can be another major problem, after all, not every fraudulent transaction is detected and reported. You need to have a simple and fairly fast real-time fraud detection model. This is the purpose of Term Paper, which is implemented here.

Зміст

Вступ

Виявлення аномалій – це процес пошуку моделей у даних, які не відповідають моделі нормальної поведінки. Метою виявлення аномалій є виявлення незвичайних випадків у даних, які, здавалося б, можна порівняти. Виявлення аномалій є важливим інструментом для виявлення шахрайства, вторгнення в мережу та інших рідкісних подій, які можуть мати велике значення, але їх важко знайти.Виявлення аномалій можна ефективно використовувати як інструмент для зменшення ризиків і виявлення шахрайства.

Сьогодні дані керують більшістю бізнес-рішень. Маючи доступ до більшої кількості даних, більшої кількості інформації, ніж будь-коли раніше, ще важливіше її аналізувати та правильно інтерпретувати. Коли йдеться про безпеку, пошук викидів — це лише перший крок. Визначення того, чи є викид загрозою безпеці, і розуміння першопричини аномалії є ключем до реального рішення.

Саме тут на допомогу приходить аналітика, оскільки вона може допомогти нам дізнатися про поведінку програми, системи, бази даних або поєднання цих речей із історичних даних, маючи можливість ідентифікувати аномалії чи тенденції до того, як вони стануть очевидними. Завчасне розуміння такої поведінки дає змогу завчасно знаходити тенденції та бачити речі, які відбуваються, перш ніж вони стануть проблемами.

В цій курсовій реалізується ідентифікація аномалій на прикладі шахрайства з банківськими карточками.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Що таке ідентифікація аномалій?

Ідентифікація аномалій — це етап інтелектуального аналізу даних, який визначає точки даних, події та/або спостереження, які відрізняються від нормальної поведінки набору даних. Аномальні дані можуть вказувати на критичні інциденти, такі як технічний збій. Машинне навчання поступово використовується для автоматизації виявлення аномалій.

Завдяки всім доступним аналітичним програмам і різноманітному програмному забезпеченню для управління компаніям тепер легше, ніж будь-коли, ефективно вимірювати кожен окремий аспект ділової активності. Це включає в себе операційну продуктивність додатків і компонентів інфраструктури, а також ключові показники ефективності (KPI), які оцінюють успіх організації. Завдяки мільйонам показників, які можна виміряти, компанії, як правило, отримують досить вражаючий набір даних для дослідження ефективності свого бізнесу.

У цьому наборі даних є шаблони даних, які представляють бізнес у стандартному вигляді. Несподівана зміна в цих шаблонах даних або подія, яка не відповідає очікуваному шаблону даних, вважається аномалією. Іншими словами, аномалія — це відхилення від стандартного вигляду бізнеса.

Але, що саме мається на увазі під "стандартним виглядом"? Немає нічого незвичайного в тому, що веб-сайт збирає велику суму доходу за один день – звичайно, якщо цей день Чорна П'ятниця. Це не є чимось незвичайним, тому що великий обсяг продажів у Чорну П'ятницю є усталеним піком природного бізнес-циклу будь-якої компанії.

Дійсно, було б аномалією, якби така компанія не мала високий обсяг продажів у Чорну П'ятницю, особливо якщо обсяги продажів у Чорну П'ятницю за попередні роки були дуже високими. Відсутність змін може бути аномалією, якщо вона порушує шаблон, який є нормальним для даних цього конкретного показника. Аномалії не є категорично хорошими чи поганими, це лише відхилення від очікуваного значення метрики в певний момент часу.

1.2 Методи ідентифікацій аномалій

Під час пошуку даних щодо аномалій, які є відносно рідкісними, користувач неминуче зіткнеться з відносно високими рівнями "шуму", які можуть бути схожі на аномальну поведінку. Це пояснюється тим, що межа між ненормальною та нормальною поведінкою зазвичай нечітка і може часто змінюватися, коли зловмисники адаптують свої стратегії.

Крім того, оскільки багато шаблонів даних базуються на часі та сезонності, існує додаткова вбудована складність методів виявлення аномалій. Наприклад, необхідність розбити численні тенденції з часом вимагає більш складних методів визначення фактичних змін у сезонності порівняно з шумом або аномальними даними.

З усіх цих причин існують різні методи виявлення аномалій. Залежно від обставин один може бути кращим за інших для певного користувача чи набору даних. Генеративний підхід створює модель, засновану виключно на прикладах звичайних даних із навчання, а потім оцінює кожен тестовий приклад, щоб побачити, наскільки добре він відповідає моделі. Навпаки, дискримінаційний підхід намагається розрізнити нормальні та ненормальні класи даних. Обидва типи даних використовуються для навчання систем дискримінаційним підходам.

Ідентифікація аномалій на основі кластеризації:

Виявлення аномалій на основі кластеризації залишається популярним у неконтрольованому навчанні. Він базується на припущенні, що подібні точки даних мають тенденцію групуватися разом у групи, що визначається їхньою близькістю до локальних центроїдів.

K-середні, широко використовуваний алгоритм кластеризації, створює «k» подібних кластерів точок даних. Потім користувачі можуть налаштувати системи на позначення екземплярів даних, які не входять у ці групи, як аномалії даних. Будучи технікою без нагляду, кластеризація не потребує маркування даних.

Алгоритми кластеризації можуть бути розгорнуті для захоплення аномального класу даних. Алгоритм уже створив багато кластерів даних у

навчальному наборі, щоб обчислити поріг для аномальної події. Потім він може використовувати це правило для створення нових кластерів, імовірно, збираючи нові аномальні дані.

Однак кластеризація не завжди працює для даних часових рядів. Це пояснюється тим, що дані відображають еволюцію з часом, але метод створює фіксований набір кластерів.

Ідентифікація аномалій на основі щільності:

Методи виявлення аномалій на основі щільності вимагають маркованих даних. Ці методи виявлення аномалій ґрунтуються на припущенні, що нормальні точки даних, як правило, виникають у щільному сусідстві, тоді як аномалії з'являються далеко й рідко.

Існує два типи алгоритмів для цього типу оцінки аномалії даних:

- Коефіцієнт локального викиду (LOF), який також називають відносною щільністю даних, базується на відстані досяжності;
- Підтримка векторного машинного виявлення аномалій;

Машина опорних векторів (SVM) зазвичай використовується в контрольованих налаштуваннях, але розширення SVM також можна використовувати для виявлення аномалій для деяких немаркованих даних. SVM — це нейронна мережа, яка добре підходить для класифікації лінійно роздільних бінарних шаблонів — очевидно, що краще розділення, то чіткіші результати.

Такі алгоритми виявлення аномалій можуть вивчати більш м'які межі залежно від цілей кластеризації екземплярів даних і належного визначення аномалій. Залежно від ситуації подібний детектор аномалій може виводити числові скалярні значення для різних цілей.

1.3 Машинне навчання

Є декілька типів машинного навчання:

- Контрольоване (Supervised)
- Неконтрольоване (Unsupervised)
- Часткове (Semi-supervised)

Контрольоване (Supervised) машинне навчання для ідентифікації аномалій:

Контрольоване навчання будує прогностичну модель, використовуючи позначений навчальний набір із нормальними та

аномальними зразками. До найпоширеніших контрольованих методів належать k-сусіди, дерева рішень, контрольовані нейронні мережі та SVM.

Перевага контрольованих моделей полягає в тому, що вони можуть запропонувати більш високий рівень виявлення, ніж неконтрольовані методи. Це пояснюється тим, що вони можуть повертати оцінку достовірності з виходом моделі, включати дані та попередні знання та кодувати взаємозалежності між змінними.

Неконтрольоване (Unsupervised**) машинне навчання для ідентифікації аномалій:**

Неконтрольовані методи не вимагають ручного маркування даних навчання. Натомість вони діють на основі припущення, що лише невеликий, статистично відмінний відсоток мережевого трафіку є зловмисним і ненормальним. Таким чином, ці методи припускають, що колекції частих, подібних випадків є нормальними, і позначають нечасті групи даних як шкідливі.

Найпопулярніші алгоритми неконтрольованого виявлення аномалій включають автокодери, K-середні, GMM, аналіз на основі перевірки гіпотез і PCA.

Часткове (Semi-supervised) машинне навчання для ідентифікації аномалій:

Термін часткове виявлення аномалій може мати різні значення. Часткове виявлення аномалій може стосуватися підходу до створення моделі для нормальних даних на основі набору даних, який містить як нормальні, так і аномальні дані, але не позначені. Цей метод тренування в ході можна назвати частковим.

Частковий алгоритм виявлення аномалій також може працювати з частково позначеним набором даних. Потім він створить алгоритм класифікації лише на цій позначеній підмножині даних і використає цю модель для прогнозування статусу решти даних.

1.4 Випадки використання ідентифікацій аномалій

Деякі з основних випадків використання виявлення аномалій включають виявлення вторгнень на основі аномалій, виявлення шахрайства, запобігання втраті даних (DLP), виявлення зловмисного програмного забезпечення на основі аномалій, виявлення медичних аномалій, виявлення аномалій на соціальних платформах, виявлення

аномалій журналу, Інтернет речей (IoT), виявлення аномалій у Big Data, аномалій промисловості/моніторингу та аномалій у відеоспостереженні.

Система виявлення вторгнень на основі аномалій (IDS) — це будь-яка система, призначена для ідентифікації та запобігання зловмисній діяльності в комп'ютерній мережі. Один комп'ютер може мати власний IDS, який називається системою виявлення вторгнень на хост (HIDS), і таку систему також можна розширити для покриття великих мереж. У цьому масштабі це називається виявлення вторгнення в мережу (NIDS).

Це також іноді називають виявленням аномалій за у мережевій поведінці, і це той вид постійного моніторингу, який розроблено для забезпечення інструментів виявлення аномалій у поведінці мережі. Більшість IDS залежать від методів виявлення на основі сигнатур або аномалій, але оскільки IDS на основі сигнатур погано обладнані для виявлення унікальних атак, методи виявлення на основі аномалій залишаються більш популярними.

Шахрайство в банківській сфері (операції з кредитними картками, заявки на повернення податків тощо), страхові претензії (автомобільні, медичні тощо), телекомунікації та інших сферах є серйозною проблемою як для приватного бізнесу, так і для уряду. Виявлення шахрайства потребує адаптації, виявлення та запобігання з даними в режимі реального часу.

Запобігання втраті даних (DLP) схоже на запобігання шахрайству, але зосереджується виключно на втраті конфіденційної інформації на ранній стадії. На практиці це означає реєстрацію та аналіз доступу до файлових серверів, баз даних та інших джерел інформації майже в реальному часі для виявлення незвичайних моделей доступу.

Виявлення зловмисного програмного забезпечення є ще однією важливою областю, яка зазвичай поділяється на етапи вилучення функцій і кластеризації/класифікації. Величезним викликом тут є сам масштаб даних, а також адаптивний характер зловмисної поведінки.

Виявлення аномалій у медичних зображеннях і записах дозволяє експертам ефективніше діагностувати та лікувати пацієнтів. Великі обсяги незбалансованих даних означають обмежену здатність виявляти та інтерпретувати шаблони без цих методів. Це область, яка ідеально підходить для штучного інтелекту, враховуючи величезний обсяг обробки даних.

Виявлення аномалій у соціальній мережі дозволяє адміністраторам ідентифікувати фальшивих користувачів, онлайн-шахраїв, розповсюджувачів чуток і спамерів, які можуть мати серйозні наслідки для бізнесу та суспільства.

Моніторинг даних, створених у сфері Інтернету речей (IoT), гарантує, що дані, створені компонентами IT-інфраструктури, мітками радіочастотної ідентифікації (RFID), метеостанціями та іншими датчиками, є точними, і виявляє помилкову та шахрайську поведінку до того, як станеться лихо. Те ж саме стосується моніторингу промислових систем, таких як високотемпературні енергетичні системи, електростанції, вітряні турбіни та накопичувачі, які піддаються величезному щоденному навантаженню.

2. Застосовані засоби та методи

2.1 Мова програмування

Python — це потужна мова програмування, яка проста у вивченні. Вона має ефективні високорівневі структури даних і простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний і динамічний синтаксис Python набір тексту разом із його інтерпретованою природою робить його ідеальною мовою для написання сценаріїв і швидкого застосування розвиток у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вільному доступі у вихідному або двійковому вигляді для усіх основних платформ з веб-сайту Python, https://www.python.org/, і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програми та інструменти, а також додаткову документацію. Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інші мови, що викликаються з C). Python також підходить як мова розширення для настроюваних програми.

Python — це високорівнева, інтерпретована, інтерактивна та об’єктно-орієнтована мова сценаріїв. Розроблено бути високочитабельним. У ньому часто використовуються англійські ключові слова, де, як інші мови, використовують розділові знаки, і вона має менше синтаксичних конструкцій, ніж інші мови.

Python інтерпретується, обробляється під час виконання інтерпретатором. Вам не потрібно компілювати вашу програму перед її виконанням. Це схоже на PERL і PHP.

Python є об’єктно-орієнтованим, підтримує об’єктно-орієнтований стиль або техніку програмування, яка інкапсулює код всередині об'єктів.

Python — це мова для початківців, чудова мова для програмістів початківців і підтримує розробку широкого спектру додатків від простої обробки тексту до браузерних ігор.

2.2 Особливості машинного навчання

Машинне навчання – це метод статистичного навчання, коли кожен екземпляр у наборі даних описується набором функцій або атрибутів. Навпаки, термін «Глибинне навчання» — це метод статистичного навчання, який витягує функції або атрибути з вихідних даних. Глибинне навчання робить це за допомогою нейронних мереж із багатьма прихованими шарами, великими даними та потужними обчислювальними ресурсами. Терміни здаються дещо взаємозамінними, однак за допомогою методу глибокого навчання алгоритм створює представлення дані автоматично. На відміну від цього, подання даних жорстко закодовано, як набір функцій в алгоритмах машинного навчання, що вимагає додаткових процесів, таких як вибір і вилучення функцій (наприклад, PCA). Обидва ці терміни різко контрастують з іншим класом класичних алгоритмів штучного інтелекту, відомих як системи на основі правил, де кожне рішення програмується вручну таким чином, що він нагадує статистичну модель.

У машинному та глибокому навчанні існує багато різних моделей, які поділяються на дві різні категорії: контрольовані та неконтрольовані. У неконтрольованому навчанні такі алгоритми, як k-середні, ієрархічна кластеризація та моделі суміші Гаусса, намагаються вивчити значущі структури в даних. Контрольоване навчання включає вихідну мітку, пов’язану з кожним екземпляром у наборі даних. Цей вихід може бути дискретним/категоріальним або дійсним. Регресійні моделі оцінюють результати з реальним значенням, тоді як моделі класифікації оцінюють результати з дискретними значеннями. Прості двійкові моделі класифікації мають лише дві моделі вихідні мітки, 1 (позитивний) і 0 (негативний). Деякі популярні алгоритми контрольованого навчання, які

вважаються машинним навчанням: це лінійна регресія, логістична регресія, дерева рішень, опорні векторні машини та нейронні мережі, а також непараметричні моделі, такі як k-найближчі сусіди.

2.3 Бібліотеки використанні для реалізації проекту

NumPy:

NumPy - це бібліотека Python, яку застосовують для математичних обчислень: починаючи з базових функцій і закінчуючи лінійною алгеброю. Повна назва бібліотеки — Numerical Python extensions, або Числові розширення Python.

Ця бібліотека має кілька важливих особливостей, які зробили її популярним інструментом. По-перше, вихідний код у вільному доступі зберігається на GitHub, тому NumPy називають open-source модулем для Python.

По-друге, бібліотека написана мовами C та Fortran. Це компілювані мови (мови програмування, текст яких перетворюється на машинний код — набір інструкцій для конкретного типу процесора. Перетворення відбувається за допомогою спеціальної програми-компілятора, завдяки якому обчислення компілюваними мовами відбуваються швидше), на яких обчислення виробляються набагато швидше та ефективніше, ніж мовами, що інтерпретуються (мови програмування, які не заточені під конкретний тип процесора і можуть бути запущені на різних типах пристроїв). До цих мов належить і сам Python.

Де використовується NumPy:

- Наукові обчислення. NumPy користуються вчені для вирішення багатовимірних завдань у математиці та фізиці, біоінформатиці, обчислювальної хімії та навіть когнітивної психології.
- Створення нових потужних бібліотек. На основі NumPy з'являються нові типи масивів, можливості яких виходять за межі того, що пропонує бібліотека. Наприклад, бібліотеки Dask, CuPy чи XND.
- Data Science. В основі екосистеми для аналізу даних лежить NumPy. Бібліотека використовується на всіх етапах роботи з даними: вилучення та перетворення, аналіз, моделювання та оцінка, репрезентація.
- Machine Learning. Бібліотеки для машинного навчання scikit-learn та SciPy також працюють завдяки обчислювальним потужностям NumPy.

- Візуалізація даних. У порівнянні безпосередньо з Python можливості NumPy дозволяють дослідникам візуалізувати набори даних, які набагато більші за розміром. Наприклад, бібліотека лежить в основі системи PyViz, яка включає десятки програм для візуалізації.

Як працює NumPy:

Спочатку розберемося у пристрої масивів, які обробляє NumPy. Розглянемо однорідний двовимірний масив. Він виглядає як проста таблиця – дві осі значень та осередки всередині (елементи масиву). Якщо з'явиться третя вісь, масив стане тривимірним. Важлива умова - всі елементи повинні мати єдиний тип даних, наприклад, лише цілі числа.

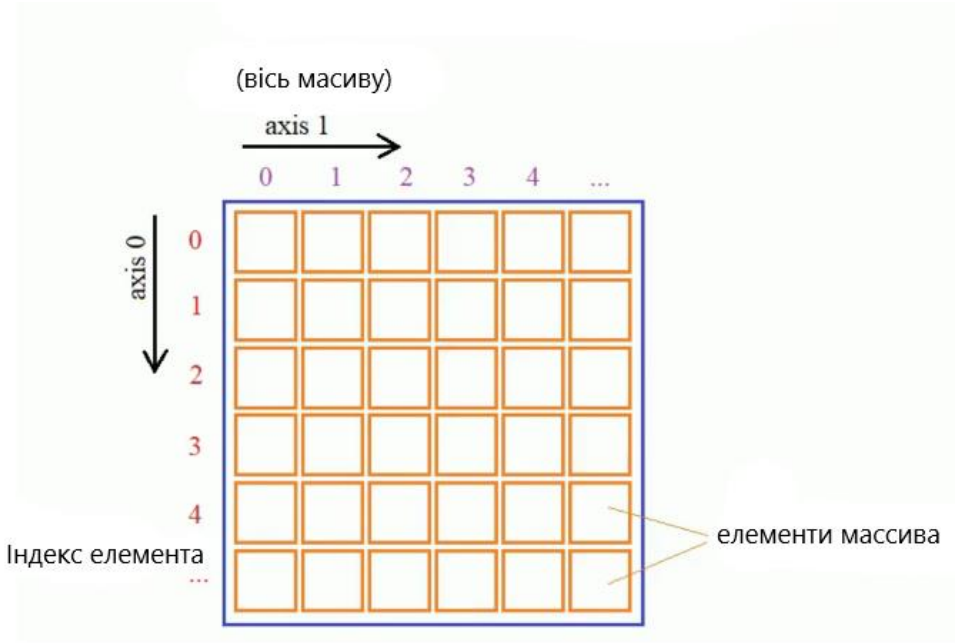

Рис.1 - Приклад візуалізації двовимірного масиву

Звичайно, крім двовимірних масивів, бібліотека NumPy обробляє й інші з різною кількістю осей. Цю варіативність позначають числом N, як будь-яку змінну математичної задачі. Тому зазвичай кажуть, що NumPy працює з N-вимірними масивами даних.

З цими даними NumPy здійснює обчислення, використовуючи математичні функції, генератори випадкових чисел, лінійні рівняння або перетворення Фур'є. Наприклад, можна вирішити систему рівнянь методом linalg.solve:

```
import numpy as np
left = np.array( [ [1, 3], [2, -4] ] )
right = np.array( [9, 8] )
```

np.linalg.solve(left, right)
Відповідь: array([6., 1.])

Як і сам Python, бібліотека NumPy відрізняється простотою у вивченні та використанні. Для початку роботи достатньо освоїти концепцію масивів. Наприклад, у базових арифметичних обчисленнях є спосіб обробки масивів, який називають трансляцією або broadcasting.

Якщо в масиві величини вказані в милях, а результат потрібно одержати в кілометрах, можна помножити його на звичайне число 1,6 (скалярну величину). NumPy приймає самостійне рішення помножити на задане число кожен елемент у масиві, і користувачеві не доводиться прописувати окрему команду.

Pandas:

Pandas - це бібліотека Python для обробки та аналізу структурованих даних, її назва походить від "panel data" ("панельні дані"). Панельними даними називають інформацію, отриману в результаті досліджень та структуровану у вигляді таблиць. Для роботи з такими масивами даних створено Pandas.

Pandas – це opensource-бібліотека, тобто її вихідний код у відкритому доступі розміщено на GitHub. Користувачі можуть додавати туди свій код: вносити пояснення, доповнювати методи роботи та оновлювати розділи. Для роботи буде потрібний компілятор (програма, яка перекладає текст з мови програмування в машинний код) C/C++ та середовище розробки Python. Детальний процес встановлення компілятора для різних операційних систем можна знайти в документації Pandas.

Навичка роботи з цією бібліотекою стане в нагоді дата-саентистам або аналітикам даних. За допомогою Pandas ці фахівці можуть групувати та візуалізувати дані, створювати зведені таблиці та робити вибірку за певними ознаками.

Pandas працюють з форматами csv, excel, sql, html, hdf та іншими. Повний список можна переглянути за допомогою методу .read, де через нижнє підкреслення _ будуть вказані всі доступні формати.

Matplotlib:

Matplotlib — це бібліотека Python для візуалізації даних. У ній можна побудувати двовимірні (плоскі) та тривимірні графіки. Альтернатива модуля візуалізації програми для технічних обчислень MatLab. Matplotlib має об'єктно-орієнтований інтерфейс, тобто користувач безпосередньо взаємодіє з кожним об'єктом. За допомогою коду можна задавати будь-який елемент діаграми, у тому числі ярлики та позначки на осях.

Matplotlib використовують для відтворення різноманітних видів графіків. Це незамінна бібліотека для будь-якого аналітика даних. Крім цього, Matplotlib є основою інших бібліотек, наприклад Seaborn, яка представляє високорівневий інтерфейс над Matplotlib. У деяких випадках ми використовуємо Seaborn, наприклад, коли хочемо зробити швидко і красиво, але коли хочеться більшої деталізації та опрацювання, то сміливо користуємось Matplotlib.

Matplotlib.pyplot - найвищий високрівневий інтерфейс з набором команд і функцій. У високорівневому інтерфейсі все автоматизовано, тому його найпростіше освоювати новачкам.

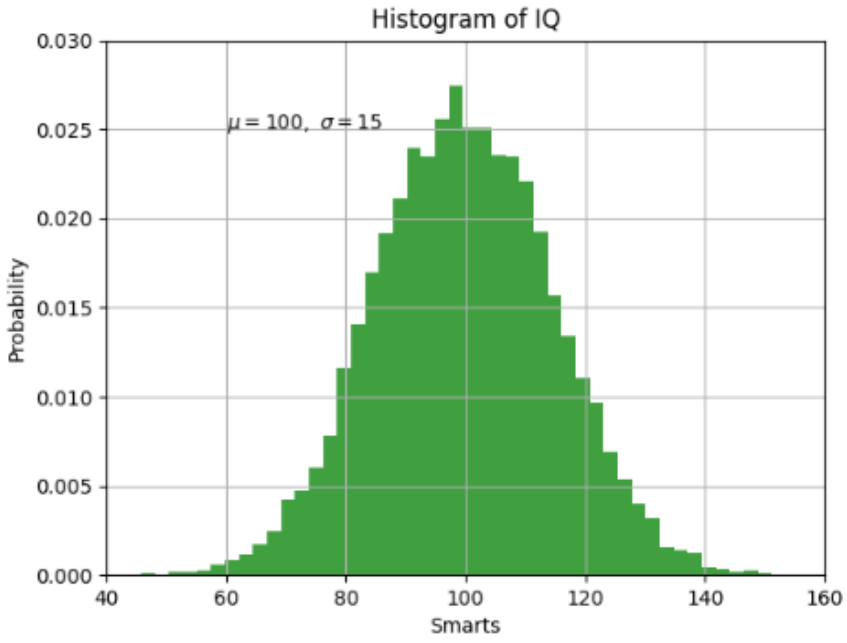

Рис.2 - Гістограма побудована в Matplotlib

Seaborn:

Seaborn – це бібліотека для візуалізації даних та виділення їх статистичних особливостей. Seaborn написана поверх бібліотеки Matplotlib, але пропонує інтерфейс вищого рівня, а це означає, що "тони" шаблонного коду, що формує зовнішній вигляд графіків, заховані "під капотом". Щоб створити графік більш ніж придатний для швидкої "розвідки" даних, потрібно всього один-два рядки коду, а якщо потрібно підготувати графік для презентації або публікації, Seaborn також надає доступ до низькорівневих налаштувань.

Ще одна важлива особливість бібліотеки Seaborn - це закладений у ній механізм попередньої обробки даних, можливий завдяки тісній інтеграції з бібліотекою Pandas. Дані можна передавати прямо в об'єктах DataFrame, а Seaborn сама виконає всю необхідну роботу: агрегацію (виділення підмножин), статистичну обробку (наприклад, обчислення довірчих інтервалів) та візуальне виділення отриманих результатів.

Так само як Python дозволяє зосередитись на задачі, а не коді за допомогою якого вона вирішується, Seaborn дозволяє сконцентруватися на важливих нюансах візуального представлення ваших даних, а не коді за допомогою якого ці нюанси потрібно малювати та виділяти.

Мабуть, єдиний мінус Seaborn полягає в тому, що одержувані графіки не є інтерактивними - не можна навести покажчик миші на якусь точку і побачити відповідне їй значення, не можна виділити якусь область графіка і збільшити її.

sklearn:

Scikit-learn – один з найбільш широко використовуваних пакетів Python для Data Science та Machine Learning. Він дозволяє виконувати безліч операцій та надає безліч алгоритмів. Scikit-learn також пропонує чудову документацію про свої класи, методи та функції, а також опис використовуваних алгоритмів.

Scikit-Learn підтримує:

- попередню обробку даних;
- зменшення розмірності;
- вибір моделі;
- регресії;
- класифікації;
- кластерний аналіз

Він також надає кілька наборів даних, які можна використовувати для тестування ваших моделей.

Scikit-learn не реалізує все, що пов'язане із машинним навчанням. Наприклад, він не має комплексної підтримки для:

- нейронних мереж;
- самоорганізуються карт (мереж Кохонена);
- навчання асоціативних правил;
- навчання з підкріпленням (reinforcement learning)

Scikit-learn заснований на NumPy та SciPy, тому необхідно зрозуміти хоча б ази цих двох бібліотек, щоб ефективно застосовувати Scikit-learn.

Scikit-learn – це пакет з відкритим вихідним кодом. Як і більшість матеріалів з екосистеми Python, вона безкоштовна навіть для комерційного використання. Його ліцензовано під ліцензією BSD.

Xgboost:

Xgboost означає eXtreme Gradient Boosting і розроблено на основі градієнтного посилення. Xgboost підвищує продуктивність звичайної моделі підвищення градієнта.

Методи послідовного ансамблю, також відомі як «Boosting», створюють послідовність моделей, які намагаються виправити помилки моделей перед ними в послідовності. Перша модель побудована на даних навчання, друга покращує першу модель, третя покращує другу і так далі.

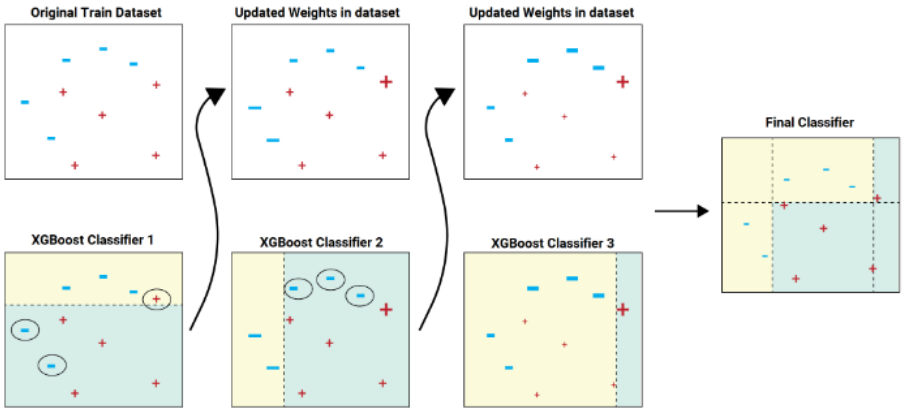

Рис.3 - Передавання даних

На зображені вище, навчений набір даних передається до класифікатора 1. Жовтий фон означає, що класифікатор передбачив дефіс, а синій фон означає, що він передбачив плюс. Модель класифікатора 1

неправильно передбачає два дефіси та один плюс. Вони виділені кружечком. Ваги цих неправильно передбачених точок даних збільшуються та надсилаються до наступного класифікатора. Тобто до класифікатора 2. Класифікатор 2 правильно передбачає два дефіси, які не зміг зробити класифікатор 1. Але класифікатор 2 також допускає деякі інші помилки. Цей процес триває, і ми маємо комбінований остаточний класифікатор, який правильно прогнозує всі точки даних.

Моделі класифікаторів можна додавати, поки всі елементи в навчальному наборі даних не будуть правильно передбачені або додано максимальну кількість моделей класифікаторів. Оптимальну максимальну кількість моделей класифікаторів для навчання можна визначити за допомогою налаштування гіперпараметрів.

imblearn:

imbalanced-learn — це набір інструментів Python з відкритим вихідним кодом, спрямований на надання широкого спектру методів для вирішення проблеми незбалансованого набору даних, яка часто зустрічається в машинному навчанні та розпізнаванні образів. Впроваджені сучасні методи можна розділити на 4 групи: (I) недостатня вибірка, (II) надмірна вибірка, (III) комбінація надмірної та недостатньої вибірки та (IV) методи ансамблевого навчання . Пропонований набір інструментів залежить лише від numpy, scipy та scikit-learn і поширюється за ліцензією MIT. Крім того, він повністю сумісний із scikit-learn і є частиною проекту, який підтримується scikit-learn-contrib. Документація, модульні тести, а також інтеграційні тести надаються для полегшення використання та внеску. Вихідний код, двійкові файли та документацію можна завантажити з https://github.com/scikit-learn-contrib/imbalanced-learn.

2.4 База даних

База даних містить транзакції, здійснені за допомогою кредитних карток у вересні 2013 року європейськими власниками карток.

Ця база даних представляє транзакції, які відбулися за два дні, де ми маємо 492 шахрайства з 284 807 транзакцій.

Вона містить лише числові вхідні змінні, які є результатом перетворення PCA. Функції V1, V2, … V28 є основними компонентами, отриманими за допомогою PCA, єдиними характеристиками, які не були перетворені за допомогою PCA, є «Час» і «Сума». Функція «Час» містить секунди, що минули між кожною транзакцією та першою транзакцією в наборі даних. Функція «Сума» — це сума транзакції, цю функцію можна

використовувати для навчання, залежного від вартості, залежного від прикладу. Функція «Клас» — це змінна відповіді, яка приймає значення 1 у разі шахрайства та 0 в іншому випадку.

3. ОГЛЯД КОДУ ТА РЕЗУЛЬТАТИ

In [1]:
```
# Імпорт необхідних бібліотек для проекту
import numpy as np
import pandas as pd
import time
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from termcolor import colored as cl # Для налаштування тексту
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier
from sklearn.multiclass import OneVsRestClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn import metrics, model_selection
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    classification_report,
    roc_auc_score,
    f1_score,
)

ATTRS = ["bold"]
COLOR = "blue"
```

Рис.4 - Імпортування бібліотек

In [3]:
```
df = pd.read_csv("creditcard.csv")

print(cl("Розмір бази {}".format(df.shape), attrs=ATTRS, color=COLOR))
print(
    cl("Дубльовані занчення {}".format(df.duplicated().sum()), attrs=ATTRS, color=COLOR)
)
print(cl("Нульові значення {}".format(df.isnull().sum().sum()), attrs=ATTRS, color=COLOR))


df.drop_duplicates(inplace=True)
```

```
Розмір бази (284807, 31)
Дубльовані занчення 1081
Нульові значення 0
```

Рис.5 - Читання даних

Вибір особливостей:

Ми видалили повторювані записи, і в наборі даних немає нульових значень. Оскільки існує 31 стовпець, нам потрібно знайти найкращі стовпці, які підуть для навчання даних. Під час навчання набору даних слід уникати будь-яких сильно корельованих функцій. Найкращий спосіб перевірити кореляцію — це графічно візуалізувати кореляційну матрицю, яка дає нам уявлення про те, як функції корелюють одна з одною, і може допомогти нам передбачити, які характеристики є найбільш релевантними для прогнозу.

```
In [6]:
correlation = df.corr()
correlation["Class"].sort_values(ascending=False)
plt.figure(figsize=(20, 9))
plt.title("Кореляційний графік особливостей операцій з кредитною карткою", fontsize=16)
sns.heatmap(
    correlation,
    xticklabels=correlation.columns,
    yticklabels=correlation.columns,
    linewidth=2,
    cmap="Reds",
    cbar=True,
)
plt.show()
```

Рис.6 - Особливості співвідношення з цілю

Рис.7 - HeatMap

У HeatMap ми бачимо, що більшість функцій не корелюють з іншими функціями, але є деякі функції, які або позитивно, або негативно корелюють одна з одною. Наприклад, V2 і V5 сильно негативно корелюють з функцією під назвою Сума. Ми також бачимо певну кореляцію з V20 і Amount. Це дає нам глибше розуміння наявних у нас Даних.

Нижче я зменшую розмірність набору даних на основі коефіцієнта кореляції між функціями та вибираю найкращі функції на основі значення хі-квадрат.

In [7]:
```
print(
    cl(
        "Розмір набору даних до зменшення розмірності: {}".format(
            df.shape
        ),
        attrs=ATTRS,
        color=COLOR,
    )
)
columns = list(df.columns)
for column in columns:
    if column == "Class":
        continue

    filtered_columns = [column]
    for col in df.columns:
        if (column == col) | (column == "Class"):
            continue
        cor_val = df[column].corr(df[col])
        if cor_val > 0.7:
            columns.remove(col)
            continue
        else:
            filtered_columns.append(col)
    df = df[filtered_columns]

features = df.drop(["Class"], axis=1)

scaler = MinMaxScaler((0, 1))
X = scaler.fit_transform(features)
selector = SelectKBest(chi2, k=21)
selector.fit(X, df["Class"])
filtered_columns = selector.get_support()

filtered_data = features.loc[:, filtered_columns]
df = filtered_data.join(df["Class"])

print(
    cl(
        "Розмір набору даних зараз становить: {}".format(df.shape),
        attrs=ATTRS,
        color=COLOR,
    )
)
```

```
Розмір набору даних до зменшення розмірності: (283726, 31)
Розмір набору даних зараз становить: (283726, 22)
```

Рис.7 - Зменшення розмірності набору даних

Ми вибрали найкращі функції для навчання даних. Тепер давайте перевіримо випадки шахрайства та не шахрайства в наборі даних. Це дає уявлення про те, чи є набір даних незбалансованим.

```
fraudulent_transactions = df.query("Class==1")
genuine_transactions = df.query("Class==0")
total_cases = len(df)
fraudulent_count = len(fraudulent_transactions)
genuine_count = len(genuine_transactions)
fraudulent_percent = round(fraudulent_count / total_cases * 100, 4)

print(cl("ІНФОРМАЦІЯ ПРО ТРАНЗАКЦІЇ В БАЗІ ДАНИХ", attrs=["bold"], color="green"))
print(
    cl(
        "-------------------------------------------------",
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "Загальна кількість транзакцій становить{}".format(total_cases),
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "Кількість справжніх транзакцій становить {}".format(genuine_count),
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "Кількість шахрайських транзакцій {}".format(fraudulent_count),
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "Відсоток шахрайських транзакцій становить {}%".format(fraudulent_percent),
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "-------------------------------------------------",
        attrs=["bold"],
        color="green",
    )
)

cases = ["Справжні випадки", "Шахрайські випадки"]
colors = ["lawngreen", "deeppink"]

data = [genuine_count, fraudulent_count]
explode = (0, 0.1)
LEFT = 0.47
BOTTOM = 0.4
WIDTH = 0.45
HEIGHT = 0.45


fig, axes = plt.subplots(figsize=(20, 6))
ax = fig.add_axes([LEFT, BOTTOM, WIDTH, HEIGHT])

axes.bar(
    cases,
    data,
    color=colors,
)
axes.set_ylabel("Кількість випадків", fontsize=13)
axes.set_ylim(0, 310000)
axes.tick_params(axis="x", labelsize=14)
axes.bar_label(axes.containers[0], padding=4, fontsize=14)
ax.pie(
    data,
    labels=cases,
    explode=explode,
    shadow=True,
    colors=colors,
    autopct="%.2f%%",
    textprops={"fontsize": 14},
)
fig.suptitle(
    "Справжні та шахрайські випадки транзакцій ", fontsize=16
)
plt.show()
```

Рис.8, 9 - Перевірка

```
ІНФОРМАЦІЯ ПРО ТРАНЗАКЦІЇ В БАЗІ ДАНИХ
--------------------------------------------------
Загальна кількість транзакцій становить283726
Кількість справжніх транзакцій становить 283253
Кількість шахрайських транзакцій 473
Відсоток шахрайських транзакцій становить 0.1667%
--------------------------------------------------
```

Рис.10 - Вивід

Рис.11 - Справжні та шахрайські випадки

Ми бачимо, що з 283726 зразків лише 473 випадки шахрайства, що становить лише 0,166% від загальної кількості зразків. Це дуже незбалансовані дані щодо цілі. Існують різні методи лікування незбалансованих даних.

Більшість методів машинного навчання найкраще працюють для збалансованого набору даних і погано працюють для незбалансованого набору даних. У наборі даних із дуже незбалансованими класами класифікатор завжди «прогнозує» найпоширеніші класи без будь-якої розробки функцій, і він завжди матиме високу точність. Таким чином, точність є оманливим показником для обчислення дуже незбалансованих даних. Метрикою, яка може забезпечити кращу обробку другорядних класів, є:

- **Confusion Matrix**
- **Precision**
- **Recall**
- **F1-score**
- **AOC**

Інша техніка полягає в тому, щоб покарати алгоритми, що збільшує вартість помилок класифікації для класу меншості. Penalized-SVM — один

із популярних алгоритмів. Під час навчання ми можемо використовувати аргумент class_weight ="balanced", щоб штрафувати за помилки класу меншості на суму, пропорційну тому, наскільки він недостатньо представлений. Ми також хочемо включити аргумент probability =True, якщо ми хочемо ввімкнути оцінки ймовірності для алгоритмів SVM.

Ми можемо спеціально змінити алгоритми, коли набір даних незбалансований. Алгоритм на основі дерева краще працює в незбалансованому наборі даних, вивчаючи ієрархію питань if/else. Використовуючи «Класифікатор випадкового лісу», «SVM» і «XGBoost» і порівняйте їхні бали, щоб вибрати найкращу модель. Нарешті, я спрогнозую новий набір даних. Я використовую техніку зменшення розмірності.

```
print(cl("Деталі кількості шахрайських операцій:", attrs=ATTRS, color=COLOR))
print(cl(fraudulent_transactions.Amount.describe(), attrs=ATTRS, color=COLOR))
print(
    cl("-------------------------------------------------------", attrs=ATTRS, color=COLOR)
)
print(
    cl("-------------------------------------------------------", attrs=ATTRS, color=COLOR)
)
print(cl("Деталі кількості справжніх транзакцій::", attrs=ATTRS, color=COLOR))
print(cl(genuine_transactions.Amount.describe(), attrs=ATTRS, color=COLOR))

print(
    cl("-------------------------------------------------------", attrs=ATTRS, color=COLOR)
)
```

```
Деталі кількості шахрайських операцій:
count     473.000000
mean      123.871860
std       260.211041
min         0.000000
25%         1.000000
50%         9.820000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
-------------------------------------------------------
-------------------------------------------------------
Деталі кількості справжніх транзакцій::
count    283253.000000
mean         88.413575
std         250.379023
min           0.000000
25%           5.670000
50%          22.000000
75%          77.460000
max       25691.160000
Name: Amount, dtype: float64
-------------------------------------------------------
```

Рис.13 - Вивід

Як ми можемо чітко помітити з цього, середня грошова операція для шахраїв є більшою. Це робить цю проблему надзвичайно важливою для вирішення. Давайте перевіримо транзакцію з часом.

```
def plot_line(x, y, title="", xlabel="", ylabel="", color=""):
    plt.plot(x, y, color)
    plt.title(title, fontsize=14)
    plt.xlabel(xlabel, fontsize=13)
    plt.ylabel(ylabel, fontsize=13)
    plt.ticklabel_format(axis="x", style="sci", scilimits=(0, 0))
    plt.ticklabel_format(axis="y", style="sci", scilimits=(0, 0))


a = plt.hist(fraudulent_transactions["Time"], 15, density=True)
b = plt.hist(genuine_transactions["Time"], 100, density=True)
plt.close()

plt.figure(figsize=(24, 7))
plt.subplot(1, 3, 1)
plot_line(
    x=fraudulent_transactions["Time"],
    y=fraudulent_transactions["Amount"],
    title="Час шахрайських транзакцій",
    xlabel="Час",
    ylabel="Кількість транзакцій",
    color="g-o",
)
plt.subplot(1, 3, 2)
plot_line(
    x=genuine_transactions["Time"],
    y=genuine_transactions["Amount"],
    title="Час справжніх транзакцій",
    xlabel="Час",
    ylabel="Кількість транзакцій",
    color="b-o",
)
plt.subplot(1, 3, 3)

plt.plot(a[1][1:], a[0], label="Шахрайські")
plt.plot(b[1][1:], b[0], label="Справжні")
plt.xlabel("Час", fontsize=13)
plt.title("Графік щільності", fontsize=14)
plt.ticklabel_format(axis="x", style="sci", scilimits=(0, 0))
plt.legend()
```

Рис.13 - Перевірка

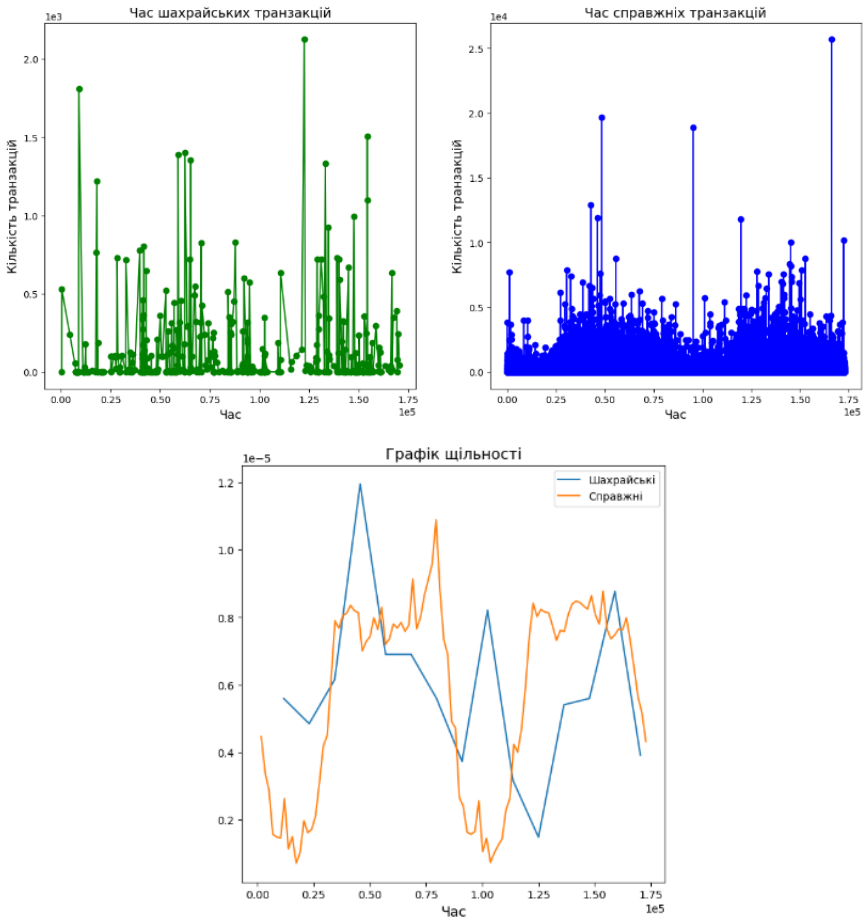

Рис.14, 15 - Графіки

Моделювання даних

Перш ніж адаптувати модель до даних, нам потрібно підготувати дані. Наведені дані не містять нульових і повторюваних значень, і всі дані масштабуються. Давайте розділимо наші дані на тренування та тестування, щоб ми могли змоделювати треновані дані та перевірити нашу модель за допомогою невидимих тестових даних і обчислити точність моделі. Оскільки набір даних дуже незбалансований, я використовую метод випадкової передискретизації для вибірки класу меншості. Я вибрав три класифікатори, а саме XGBoost, SMV і Random Forest. Для XGBoost я використовую зважений XGBoost(scale_pos_weight), щоб покарати меншість. Для моделі SMV я використовую class_weight='balanced', probability=True, а для Random Forest я використовую class_weight = "balanced"

```
amount_range = max(df["Amount"]) - min(df["Amount"])
print(
    cl(
        "Діапазон суми в даних становить {}. Ось чому нам потрібно масштабувати цільову змінну, оскільки функції є трансформова
            amount_range
        ),
        attrs=ATTRS,
        color=COLOR,
    )
)

scaler = StandardScaler()
amount = df["Amount"].values
df["Amount"] = scaler.fit_transform(amount.reshape(-1, 1))

y = df["Class"]
X = df.drop("Class", axis=1)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=51, shuffle = True, stratify = y
)
ros = RandomOverSampler(sampling_strategy="minority", random_state=1)
rus = RandomUnderSampler(random_state=1)
X_train, y_train = rus.fit_resample(X_train, y_train)

n_estimators = 10

models = []

models.append(("RFC", RandomForestClassifier(class_weight="balanced", n_jobs=40)))
models.append(("XGB", XGBClassifier(scale_pos_weight=99)))
models.append(("SMV", SVC(class_weight="balanced", probability=True)))
scoring11 = ["accuracy", "precision", "recall", "f1", "roc_auc"]
scoring = "accuracy"
cross_val_results = {}
cm_results = {}
print(cl("Оцінки перехресної перевірки різних алгоритмів", attrs=ATTRS, color=COLOR))
print(
    cl(
        "-----------------------------------------------------------",
        attrs=ATTRS,
        color=COLOR,
    )
)
for name, model in models:
    start = time.time()

    md = model.fit(X_train, y_train)
    end = time.time()
    y_pred = md.predict(X_test)
    cm = pd.DataFrame(
        confusion_matrix(y_test, y_pred),
        columns=["Predicted Healthy", "Predicted Parkinson's"],
        index=["True Healthy", "True Parkinson's"],
    )
    cm_results[name] = cm

    print(
        cl(
            "ЗАГАЛЬНИЙ ЧАС НА ВИКОНАННЯ {} це {}".format(name, end - start),
            attrs=ATTRS,
```

Рис.16 - Розділення даних

```
Діапазон суми в даних становить 102.60089068346085. Ось чому нам потрібно масштабувати цільову змінну, оскільки функції є тра
нсформованою версією PCA.
Оцінки перехресної перевірки різних алгоритмів
-----------------------------------------------------------
ЗАГАЛЬНИЙ ЧАС НА ВИКОНАННЯ RFC це 0.21392416954040527
ROCAUC оцінка RFC це 0.9476389602450461
Оцінка точності RFC це 0.9793818066471646
F1 оцінка RFC це 0.12946428571428573
-----------------------------------------------------------
ЗАГАЛЬНИЙ ЧАС НА ВИКОНАННЯ XGB це 0.11967945098876953
ROCAUC оцінка XGB це 0.9281746315622245
Оцінка точності XGB це 0.9195361787614986
F1 оцінка XGB це 0.03752107925801011
-----------------------------------------------------------
ЗАГАЛЬНИЙ ЧАС НА ВИКОНАННЯ SMV це 0.2802886962890625
ROCAUC оцінка SMV це 0.5644174070416372
Оцінка точності SMV це 0.5604271666725408
F1 оцінка SMV це 0.004311033051253394
-----------------------------------------------------------
```

Рис.17 - Вивід

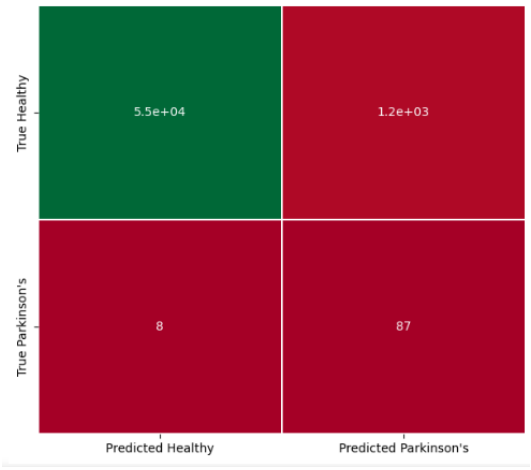

Рис.18 - Матриця для RFC

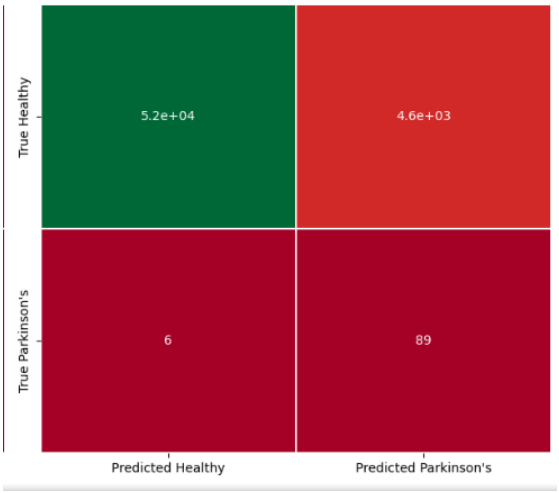

Рис.19 - Матриця для XGB

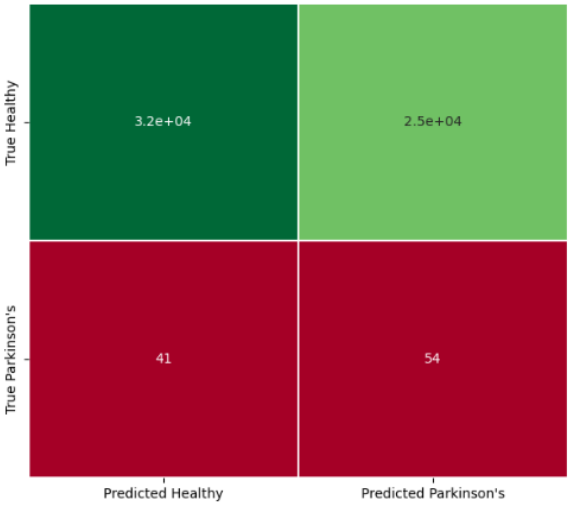

Рис.20 - Матриця для SMV

Прогноз нового набору даних:

Прогнозування нового набору даних Оскільки Класифікатор "випадкових лісів" має найкращі показники ROC AUC і F1, я вибрав Класифікатор випадкових лісів для прогнозування нового набору даних.

```
In [24]: col = [
             "Amount",
             "V26",
             "V24",
             "V19",
             "V18",
             "V17",
             "V16",
             "V14",
             "V12",
             "V11",
             "V10",
             "V9",
             "V8",
             "V7",
             "V6",
             "V5",
             "V4",
             "V3",
             "V2",
             "V1",
             "Time",
         ]
         val = [
             [
                 349.62,
                 -0.129115,
                 0.045928,
                 0.803443,
                 0.045791,
                 0.327894,
                 -0.320401,
                 -0.311169,
                 -0.4875,
                 -4.36123,
                 -2.34223,
                 0.12345,
                 0.348756,
                 0.462388,
                 -0.231234,
                 2.3456,
                 3.45223,
                 2.3221,
                 -0.072781,
                 -2.34567,
```

```
df_new = pd.DataFrame(data=val, columns=col)
amount = df_new["Amount"].values
df_new["Amount"] = scaler.fit_transform(amount.reshape(-1, 1))

rfc = RandomForestClassifier(class_weight="balanced", n_jobs=40).fit(X_test, y_test)
y_pred = rfc.predict(df_new)

lbl = []
for i in y_pred:
    if i == 1:
        lbl.append("Шахрайські")
    else:
        lbl.append("Не шахрайські")
plt.figure(figsize=(20, 4))
my_x = [i for i in range(len(y_pred))]
plt.scatter(my_x, y_pred, c="red", s=400)
for i, txt in enumerate(lbl):
    plt.annotate(txt, (my_x[i] + 0.08, y_pred[i]), fontsize=13)
plt.title(
    "Прогнозування шахрайства з кредитними картками за допомогою класифікатора XGBoost для нового набору даних",
    fontsize=14,
)
plt.xlabel("Кількість нових записів", fontsize=13)
plt.ylabel("Шахрайські/Не шахрайські", fontsize=13)
plt.yticks(ticks=())
plt.ylim(-0.2, 1.5)
plt.show()
```

Рис.21, 22 - Прогнозування нового набору даних

Рис.23 - Прогнозування шахрайства з кредитними картками за допомогою класифікатора XGBoost для нового набору даних

Висновок:

У цій курсовій реалізовано ідентифікацію шахрайських дій з банківськими картами. Шахрайство з кредитними картками є серйозною проблемою, яку потрібно вирішити. Дані є дуже незбалансованими, і дані дисбалансу обробляються за допомогою методу випадкової надлишкової/недобірної вибірки та використання відповідних параметрів у класифікаторах. Виявлення шахрайства є складною проблемою, яка потребує значного планування перш ніж кинути на нього алгоритми машинного навчання. Через це було випробувано декілька варіантів ідентифікації. Результати були порівнянні та обрано найкращі варіанти. Двома найкращими класифікаторами виявилися "Random Forest" і XGB на основі їх результатів F1. Передбачено новий набір даних за допомогою моделі "Random Forest".

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. PythonCode [Електронний ресурс].- режим доступу: URL: https://www.thepythoncode.com/article/credit-card-fraud-detection-using-sklearn-in-python
2. anodot [Електронний ресурс].- режим доступу: URL: https://www.anodot.com/blog/what-is-anomaly-detection/
3. kaggle [Електронний ресурс].- режим доступу: URL: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
4. AviNetworks [Електронний ресурс].- режим доступу: URL: https://avinetworks.com/glossary/anomaly-detection/
5. Medium [Електронний ресурс].- режим доступу: URL: https://medium.datadriveninvestor.com/credit-card-fraud-detection-using-local-outlier-factor-and-isolation-forest-in-python-56edd0a44af5
6. geekforgeeks [Електронний ресурс].- режим доступу: URL: https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/
7. pythonru [Електронний ресурс].- режим доступу: URL: https://pythonru.com/biblioteki/seaborn-plot
8. skillfactory [Електронний ресурс].- режим доступу: URL: https://blog.skillfactory.ru/glossary/matplotlib/
9. tutorialspoint [Електронний ресурс].- режим доступу: URL: https://www.tutorialspoint.com/fraud-detection-in-python
10. trenton3983.github.io [Електронний ресурс].- режим доступу: URL: https://trenton3983.github.io/files/projects/2019-07-19_fraud_detection_python/2019-07-19_fraud_detection_python.html#Explore-the-features-available-in-your-dataframe

Додаток:

```python
# Імпорт необхідних бібліотек для проекту
import numpy as np
import pandas as pd
import time
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from termcolor import colored as cl # Для налаштування тексту
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier
from sklearn.multiclass import OneVsRestClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn import metrics, model_selection
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    classification_report,
    roc_auc_score,
    f1_score,
)

ATTRS = ["bold"]
COLOR = "blue"
```

```python
df = pd.read_csv("creditcard.csv")

print(cl("Розмір бази {}".format(df.shape), attrs=ATTRS, color=COLOR))
print(
    cl("Дубльовані занчення {}".format(df.duplicated().sum()), attrs=ATTRS, color=COLOR)
)
print(cl("Нульові значення {}".format(df.isnull().sum().sum()), attrs=ATTRS, color=COLOR))


df.drop_duplicates(inplace=True)
```

```
Розмір бази (284807, 31)
Дубльовані занчення 1081
Нульові значення 0
```

```python
correlation = df.corr()
correlation["Class"].sort_values(ascending=False)
plt.figure(figsize=(20, 9))
plt.title("Кореляційний графік особливостей операцій з кредитною карткою", fontsize=16)
sns.heatmap(
    correlation,
    xticklabels=correlation.columns,
    yticklabels=correlation.columns,
    linewidth=2,
    cmap="Reds",
    cbar=True,
)
plt.show()
```

```
print(
    cl(
        "Розмір набору даних до зменшення розмірності: {}".format(
            df.shape
        ),
        attrs=ATTRS,
        color=COLOR,
    )
)
columns = list(df.columns)
for column in columns:
    if column == "Class":
        continue

    filtered_columns = [column]
    for col in df.columns:
        if (column == col) | (column == "Class"):
            continue
        cor_val = df[column].corr(df[col])
        if cor_val > 0.7:
            columns.remove(col)
            continue
        else:
            filtered_columns.append(col)
    df = df[filtered_columns]

features = df.drop(["Class"], axis=1)

scaler = MinMaxScaler((0, 1))
X = scaler.fit_transform(features)
selector = SelectKBest(chi2, k=21)
selector.fit(X, df["Class"])
filtered_columns = selector.get_support()

filtered_data = features.loc[:, filtered_columns]
df = filtered_data.join(df["Class"])

print(
    cl(
        "Розмір набору даних зараз становить: {}".format(df.shape),
        attrs=ATTRS,
        color=COLOR,
    )
)
```

```
fraudulent_transactions = df.query("Class==1")
genuine_transactions = df.query("Class==0")
total_cases = len(df)
fraudulent_count = len(fraudulent_transactions)
genuine_count = len(genuine_transactions)
fraudulent_percent = round(fraudulent_count / total_cases * 100, 4)

print(cl("ІНФОРМАЦІЯ ПРО ТРАНЗАКЦІЇ В БАЗІ ДАНИХ", attrs=["bold"], color="green"))
print(
    cl(
        "-------------------------------------------------",
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "Загальна кількість транзакцій становить{}".format(total_cases),
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "Кількість справжніх транзакцій становить {}".format(genuine_count),
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "Кількість шахрайських транзакцій {}".format(fraudulent_count),
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "Відсоток шахрайських транзакцій становить {}%".format(fraudulent_percent),
        attrs=["bold"],
        color="green",
    )
)
print(
    cl(
        "-------------------------------------------------",
```

```
        attrs=["bold"],
        color="green",
    )
)

cases = ["Справжні випадки", "Шахрайські випадки"]
colors = ["lawngreen", "deeppink"]

data = [genuine_count, fraudulent_count]
explode = (0, 0.1)
LEFT = 0.47
BOTTOM = 0.4
WIDTH = 0.45
HEIGHT = 0.45


fig, axes = plt.subplots(figsize=(20, 6))
ax = fig.add_axes([LEFT, BOTTOM, WIDTH, HEIGHT])

axes.bar(
    cases,
    data,
    color=colors,
)
axes.set_ylabel("Кількість випадків", fontsize=13)
axes.set_ylim(0, 310000)
axes.tick_params(axis="x", labelsize=14)
axes.bar_label(axes.containers[0], padding=4, fontsize=14)
ax.pie(
    data,
    labels=cases,
    explode=explode,
    shadow=True,
    colors=colors,
    autopct="%.2f%%",
    textprops={"fontsize": 14},
)
fig.suptitle(
    "Справжні та шахрайські випадки транзакцій ", fontsize=16
)
plt.show()
```

```
print(cl("Деталі кількості шахрайських операцій:", attrs=ATTRS, color=COLOR))
print(cl(fraudulent_transactions.Amount.describe(), attrs=ATTRS, color=COLOR))
print(
    cl("------------------------------------------------------", attrs=ATTRS, color=COLOR)
)
print(
    cl("------------------------------------------------------", attrs=ATTRS, color=COLOR)
)
print(cl("Деталі кількості справжніх транзакцій::", attrs=ATTRS, color=COLOR))
print(cl(genuine_transactions.Amount.describe(), attrs=ATTRS, color=COLOR))

print(
    cl("------------------------------------------------------", attrs=ATTRS, color=COLOR)
)
```

```
def plot_line(x, y, title="", xlabel="", ylabel="", color=""):
    plt.plot(x, y, color)
    plt.title(title, fontsize=14)
    plt.xlabel(xlabel, fontsize=13)
    plt.ylabel(ylabel, fontsize=13)
    plt.ticklabel_format(axis="x", style="sci", scilimits=(0, 0))
    plt.ticklabel_format(axis="y", style="sci", scilimits=(0, 0))


a = plt.hist(fraudulent_transactions["Time"], 15, density=True)
b = plt.hist(genuine_transactions["Time"], 100, density=True)
plt.close()

plt.figure(figsize=(24, 7))
plt.subplot(1, 3, 1)
plot_line(
    x=fraudulent_transactions["Time"],
    y=fraudulent_transactions["Amount"],
    title="Час шахрайських транзакцій",
    xlabel="Час",
    ylabel="Кількість транзакцій",
    color="g-o",
)
plt.subplot(1, 3, 2)
plot_line(
    x=genuine_transactions["Time"],
    y=genuine_transactions["Amount"],
    title="Час справжніх транзакцій",
    xlabel="Час",
    ylabel="Кількість транзакцій",
    color="b-o",
)
plt.subplot(1, 3, 3)

plt.plot(a[1][1:], a[0], label="Шахрайські")
plt.plot(b[1][1:], b[0], label="Справжні")
plt.xlabel("Час", fontsize=13)
plt.title("Графік щільності", fontsize=14)
plt.ticklabel_format(axis="x", style="sci", scilimits=(0, 0))
plt.legend()
```

```
amount_range = max(df["Amount"]) - min(df["Amount"])
print(
    cl(
        "Діапазон суми в даних становить {}. Ось чому нам потрібно масштабувати цільову змінну, оскільки функції є трансформован
            amount_range
        ),
        attrs=ATTRS,
        color=COLOR,
    )
)

scaler = StandardScaler()
amount = df["Amount"].values
df["Amount"] = scaler.fit_transform(amount.reshape(-1, 1))

y = df["Class"]
X = df.drop("Class", axis=1)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=51, shuffle = True, stratify = y
)
ros = RandomOverSampler(sampling_strategy="minority", random_state=1)
rus = RandomUnderSampler(random_state=1)
X_train, y_train = rus.fit_resample(X_train, y_train)

n_estimators = 10

models = []

models.append(("RFC", RandomForestClassifier(class_weight="balanced", n_jobs=40)))
models.append(("XGB", XGBClassifier(scale_pos_weight=99)))
models.append(("SMV", SVC(class_weight="balanced", probability=True)))
scoring11 = ["accuracy", "precision", "recall", "f1", "roc_auc"]
scoring = "accuracy"
cross_val_results = {}
cm_results = {}
print(cl("Оцінки перехресної перевірки різних алгоритмів", attrs=ATTRS, color=COLOR))
print(
    cl(
        "------------------------------------------------------------",
        attrs=ATTRS,
        color=COLOR,
```

```
        color=COLOR,
    )
)
for name, model in models:
    start = time.time()

    md = model.fit(X_train, y_train)
    end = time.time()
    y_pred = md.predict(X_test)
    cm = pd.DataFrame(
        confusion_matrix(y_test, y_pred),
        columns=["Predicted Healthy", "Predicted Parkinson's"],
        index=["True Healthy", "True Parkinson's"],
    )
    cm_results[name] = cm

    print(
        cl(
            "ЗАГАЛЬНИЙ ЧАС НА ВИКОНАННЯ {} це {}".format(name, end - start),
            attrs=ATTRS,
            color=COLOR,
        )
    )
    print(
        cl(
            "ROCAUC оцінка {} це {}".format(name, roc_auc_score(y_test, y_pred)),
            attrs=ATTRS,
            color=COLOR,
        )
    )
    print(
        cl(
            "Оцінка точності {} це {}".format(name, accuracy_score(y_test, y_pred)),
            attrs=ATTRS,
            color=COLOR,
        )
    )
    print(
        cl(
            "F1 оцінка {} це {}".format(name, f1_score(y_test, y_pred)),
            attrs=ATTRS,
            color=COLOR,
        )
    )
            color=COLOR,
        )
    )
    print(
        cl(
            "------------------------------------------------------------",
            attrs=ATTRS,
            color=COLOR,
        )
    )
print(
    cl(
        "------------------------------------------------------------",
        attrs=ATTRS,
        color=COLOR,
    )
)

c = 1
plt.figure(figsize=(18, 6))
for i in cm_results:
    plt.subplot(1, 3, c)
    plt.suptitle("Матриця плутанини різних моделей", fontsize=16)
    sns.heatmap(
        cm_results[i],
        cmap="RdYlGn",
        linewidth=0.2,
        annot=True,
        cbar=False,
    )
    plt.title("Матриця для {}".format(i))
    c = c + 1
plt.tight_layout()
plt.show()
```

```
col = [
    "Amount",
    "V26",
    "V24",
    "V19",
    "V18",
    "V17",
    "V16",
    "V14",
    "V12",
    "V11",
    "V10",
    "V9",
    "V8",
    "V7",
    "V6",
    "V5",
    "V4",
    "V3",
    "V2",
    "V1",
    "Time",
]
val = [
    [
        349.62,
        -0.129115,
        0.045928,
        0.803443,
        0.045791,
        0.327894,
        -0.320401,
        -0.311169,
        -0.4875,
        -4.36123,
        -2.34223,
        0.12345,
        0.348756,
        0.462388,
        -0.231234,
        2.3456,
        3.45223,
        2.3221,
        -0.072781,
        -2.34567,
        0,
    ],
    [
        549.62,
        -0.429135,
        0.0239258,
        0.40343,
        0.025791,
        0.627894,
        -0.120401,
        -0.467169,
        -0.9874,
        -2.36143,
        -1.31253,
        0.22444,
        0.146776,
        0.241288,
        -0.13224,
        1.345622,
        1.45223,
        4.23122,
        -0.073112,
        -4.34569,
        1,
    ],
    [
        49.62,
        -0.729135,
        0.0539258,
        0.70343,
        0.085791,
        0.227894,
        -0.420401,
        -0.767169,
        -0.2874,
        -4.36143,
        -4.31253,
        0.82444,
        0.246776,
        0.541288,
        -0.53224,
        4.345622,
        3.45223,
        5.23122,
        -0.093112,
        -2.34569,
        2,
    ],
]

df_new = pd.DataFrame(data=val, columns=col)
amount = df_new["Amount"].values
df_new["Amount"] = scaler.fit_transform(amount.reshape(-1, 1))

rfc = RandomForestClassifier(class_weight="balanced", n_jobs=40).fit(X_test, y_test)
y_pred = rfc.predict(df_new)

lbl = []
for i in y_pred:
    if i == 1:
        lbl.append("Шахрайські")
    else:
        lbl.append("Не шахрайські")
plt.figure(figsize=(20, 4))
my_x = [i for i in range(len(y_pred))]
plt.scatter(my_x, y_pred, c="red", s=400)
for i, txt in enumerate(lbl):
    plt.annotate(txt, (my_x[i] + 0.08, y_pred[i]), fontsize=13)
plt.title(
    "Прогнозування шахрайства з кредитними картками за допомогою класифікатора XGBoost для нового набору даних",
    fontsize=14,
)
plt.xlabel("Кількість нових записів", fontsize=13)
plt.ylabel("Шахрайські/Не шахрайські", fontsize=13)
plt.yticks(ticks=())
plt.ylim(-0.2, 1.5)
plt.show()
```