МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра компьютерных технологий и систем

АНТОНЧЕНКО Евгений Эдуардович

РАЗРАБОТКА МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ВЫЯВЛЕНИЯ МОШЕННИЧЕСКИХ ТРАНЗАКЦИЙ В БАНКОВСКИХ СИСТЕМАХ

Дипломная работа

Научный руководитель:
доцент кафедры компьютерных технологий и систем, кандидат физико-математических наук
Козловская И.С.

Допущена к защите
«___» _____ 2025 г.
Заведующий кафедрой компьютерных технологий и систем
кандидат физико-математических наук,
доктор педагогических наук, профессор В.В. Казаченок

Минск, 2025

РЕФЕРАТ

Дипломная работа, 59 стр., 15 источников, 12 рисунков, 1 приложение.

Ключевые слова: классификация транзакций, мошеннические операции, банковские системы, машинное обучение.

Объекты исследования – методы машинного обучения для распознавания мошеннических операций.

Цель исследования – разработка методов машинного обучения для задачи классификации транзакций и анализ эффективности.

Методы исследования – анализ исследований и материалов, постановка и решение задачи классификации, сравнительный анализ моделей.

В результате исследования – рассмотрены вариации мошенничества в банковских системах, проведены эксперименты с различными реализациями моделей, выполнен их сравнительный анализ.

Области применения – анализ данных, машинное обучение, безопасность банковских операций, компьютерные технологии.

РЭФЕРАТ

Дыпломная праца, 59 ст., 15 крыніц, 12 малюнкаў, 1 прыкладанне.

Ключавыя словы: класіфікацыя транзакцый, ашуканскія аперацыі, банкаўскія сістэмы, машыннае навучанне.

Аб'екты даследавання – метады машыннага навучання для распазнання ашуканскіх аперацый.

Мэта даследавання – распрацоўка метадаў машыннага навучання для задачы класіфікацыі транзакцый і аналіз эфектыўнасці.

Метады даследавання – аналіз даследаванняў і матэрыялаў, пастаноўка і рашэнне задачы класіфікацыі, параўнальны аналіз мадэляў.

У выніку даследавання – разгледжаны варыяцыі махлярства ў банкаўскіх сістэмах, праведзены эксперыменты з рознымі рэалізацыямі мадэляў, выкананы іх параўнальны аналіз.

Вобласці прымянення – аналіз дадзеных, машыннае навучанне, бяспека банкаўскіх аперацый, камп'ютэрныя тэхналогіі.

ABSTRACT

Thesis, 59 pages, 15 sources, 12 illustrations, 1 appendix.

Keywords: classification of transactions, fraudulent transactions, banking systems, machine learning.

The objects of research – machine learning methods for detecting fraudulent transactions.

The purpose of the research – developing machine learning methods for the task of classifying transactions and analyzing efficiency.

Research methods – analysis of research and materials, formulation and solution of the classification problem, comparative analysis of models.

As a result of the research – variations of fraud in banking systems were considered, experiments were conducted with various model implementations, and their comparative analysis was performed.

Areas of application – data analysis, machine learning, banking security, computer technology.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире цифровизация финансовых операций привела к значительному росту числа транзакций, совершаемых через банковские системы. Однако вместе с удобством и скоростью электронных платежей возросла и угроза мошеннических действий. Мошенничество в банковской сфере представляет собой одну из наиболее серьезных проблем для финансовых организаций, так как ежегодные потери от таких операций составляют миллиарды долларов. По данным Ассоциации сертифицированных специалистов по расследованию хищений (ACFE), компании теряют в среднем 5% своего годового дохода из-за мошеннических действий [2]. В условиях постоянного совершенствования методов атак со стороны злоумышленников, традиционные механизмы выявления мошенничества, такие как правила и сигнатуры, уже не справляются с новыми типами угроз, например, фишингом или использованием украденных данных.

Актуальность темы исследования обусловлена необходимостью разработки более эффективных методов обнаружения мошеннических транзакций. Современные технологии машинного обучения предлагают новые подходы к решению этой задачи, позволяя автоматически выявлять сложные паттерны в данных, которые трудно обнаружить вручную. Машинное обучение особенно эффективно в условиях работы с большими объемами данных и высокой скоростью их поступления, что характерно для банковских систем. Применяемые во многих финансовых организациях алгоритмы машинного обучения минимизируют риски путем анализа поведения пользователей и, соответсвенно, своевременного обнаружения подозрительных операций.

Целью данной работы является разработка методов машинного обучения для выявления мошеннических транзакций в банковских системах. Научная новизна работы заключается в разработке оптимизированного подхода к выявлению мошеннических операций, учитывающего особенности данных о транзакциях, такие как несбалансированность классов и высокая размерность. Практическая значимость исследования состоит в возможности применения разработанной модели в реальных банковских системах для минимизации потерь от мошеннических операций.

Таким образом, данная работа направлена на решение актуальной проблемы выявления мошеннических транзакций с использованием современных технологий машинного обучения. Результаты исследования могут быть полезны в области информационной безопасности и анализа данных.

ГЛАВА 1
ПРОБЛЕМА ОПРЕДЕЛЕНИЯ МОШЕННИЧЕСКИХ ОПЕРАЦИЙ

1.1 Обзор проблематики и анализ прикладных задач

Мошеннические операции в банковской сфере представляют собой действия, направленные на незаконное получение финансовых выгод за счет других лиц или организаций. Эти действия могут быть как простыми (использование украденных данных карт), так и сложными (социальная инженерия или манипуляции с корпоративными счетами). Мошеннические операции можно разделить на несколько категорий в зависимости от методов, используемых злоумышленниками, и объектов атаки.

Основные типы мошенничества включают:

1. Фишинг — это метод, при котором злоумышленники используют поддельные сайты, электронные письма или сообщения для получения конфиденциальных данных пользователей, таких как логины, пароли и данные банковских карт. Например, мошенники отправляют письма, якобы от имени банка, с просьбой обновить данные аккаунта через поддельную ссылку. По данным Антифишинговой рабочей группы (APWG) за 4 квартал 2024 года, количество случаев фишинговых атак составило более 980 000 [3].

2. Использование украденных данных кредитных карт заключается в получении доступа к данным кредитных карт или банковских счетов через взломы, физическое хищение или покупку на черном рынке. Одним из распространенных способов хищения данных кредитных карт является использование скиммеров на банкоматах - устройств, считывающих данные карт. Согласно исследованию компании Nilson Report, потери от мошенничества с кредитными картами достигли 34 миллиардов долларов США в 2023 году [12].

3. Транзакции, совершенные без согласия владельца счета, например, через поддельные устройства или программы. Согласно отчету ACFE, неавторизованные транзакции составляют около 30% всех случаев мошенничества в финансовой сфере [1].

4. Социальная инженерия – действия манипулятивного характера с доверием пользователей для получения доступа к их финансовым ресурсам. Исследование компании Verizon (2024) в рамках отчета Data Breach Investigations Report показывает, что социальная инженерия является причиной 82% успешных атак на финансовые организации [13].

1.2 Методы выявления мошеннических транзакций

Выявление мошеннических транзакций является сложной задачей, требующей сочетания различных подходов и технологий. Глобально методы можно разделить на две основные группы: детерминированные методы и современные методы, включая использование машинного обучения [1].

1.2.1 Детерминированные методы

Детерминированные (правило-ориентированные) методы основаны на ручном анализе данных и использовании заранее определенных правил. Эти методы просты в реализации, но имеют ограниченную эффективность при работе с большими объемами данных и новыми типами атак. Мошеннические операции классифицируют следующим образом:

1. Правила (Rule-based Systems) – это заранее определенные условия, используемые для фильтрации подозрительных транзакций. К примерам правил относятся блокировки операций выше определенной суммы, ограничения количества транзакций за единицу времени или запреты операций из необычных географических регионов. К преимуществам таких методов относится, главным образом, простота и низкая стоимость внедрения, однако такой подход неэффективен против новых типов атак и требуют постоянного обновления базы правил. Так же для этого типа характерен высокий уровень ложных срабатываний.

2. Сигнатуры (Signature-based Detection) – это установленные шаблоны известных мошеннических операций, которые используются для обнаружения подобных атак. В отличие от простых правил, сигнатуры представляют собой комплексные условия, учитывающие комбинации параметров транзакции, поведенческие паттерны пользователя и взаимосвязи между счетами. Сигнатурная детекция обладает высокой точностью определения тривиальных атак и, как и в случае с правилами, простотой внедрения. Тем не менее, существуют ограничения в обнаружении новых видов мошенничества и необходимость в постоянном обновлении баз данных сигнатур.

3. Пороговые значения (Threshold-based Detection) устанавливаются для параметров транзакций, таких как сумма, частота или время совершения операции, с целью выявить аномальные значения. Пороги разделяют на три типа: статические, адаптивные и групповые. Процессы, основанные на таком способе определения мошенничества, просты в реализации и поддержке, подходят для предварительной фильтрации данных, однако обладают высоким уровнем ложных срабатываний и не учитывает сложные зависимости между параметрами.

Современные системы противодействия мошенничеству, как правило, комбинируют все три метода в многоуровневую архитектуру, включающую первичный фильтр из правил и статических порогов, вторичный анализ комплексными сигнатурами и адаптивными порогами, и ручной верификацией.

1.2.2 Статистические методы

Статистические методы представляют собой фундаментальный подход к решению задачи обнаружения аномалий в банковских транзакциях. С помощью математического анализа данные методы позволяют оценить подозрительность транзакции, определяя вероятность того, является та или иная мошеннической. Однако, несмотря на меньшую эффективность в сравнении с современными алгоритмами машинного обучения, методы математической статистики по-прежнему остаются одним из ключевых инструментов борьбы с аномалиями в данных из-за простоты реализации, низких вычислительных затрат и хорошей интерпретируемости.

Методы статистического анализа строятся на предположении, что большинство транзакций соответствуют нормальному поведению клиентов, а мошеннические операции представляют собой отклонения от этого поведения. Для выявления таких аномалий могут быть использованы параметрические или непараметрические статистические модели. Статистические методы особенно ценны в условиях, когда требуется высокий уровень интерпретируемости результатов и прозрачности принятия решений, что является важным для финансового мониторинга.

Классические параметрические методы предполагают, что данные подчиняются тому или иному закону вероятностного распределения. В случае многомерных данных часто выдвигается предположение о нормальном распределении:

$$N(x|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} exp\left(\frac{-1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right),$$

где μ - вектор математических ожиданий, Σ - ковариационная матрица.

Аномалии в этом случае определяются как наблюдения, находящиеся в хвостах распределения. Мерой аномальности может служить расстояние Махаланобиса:

$$s(x) = (x-\mu)^T\Sigma^{-1}(x-\mu).$$

Подобные методы обладают существенными ограничениями:

– чувствительность к нарушениям предположения о нормальности распределения;

– неустойчивость к мультимодальным распределениям, характерным для реальных финансовых данных;

– высокая вычислительная сложность при работе с высокоразмерными данными.

Для преодоления ограничений параметрических методов применяются непараметрические статистические подходы. К ним относятся ядерные оценки плотности, общий вид которых представлен уравнением (1), и методы, основанные на квантилях, использующие эмпирические квантили распределения для определения порогов аномальности (например, правило трех сигм или его модификации).

$$f(x) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{x - x_i}{n}\right), \quad (1)$$

где K - ядерная функция (например, гауссова), h - параметр ширины окна.

Применение статистических методов к финансовым транзакциям требует учета специфических особенностей предметной области и природы данных. Главным аспектом является то, что распределение транзакций может изменяться во времени и требует адаптивных методов оценки параметров. Кроме того, зачастую для корректной оценки выбросов необходимы специальные подходы для обработки смешанных данных (например, кодирование категориальных переменных).

В целом, экспериментальные исследования показывают, что на стандартных наборах данных чистые статистические методы демонстрируют результаты, немногим уступающие более современным подходам. Однако их ценность сохраняется благодаря низкой вычислительной сложности при обработке больших объемов данных, возможности построения объяснимых правил обнаружения аномалий и устойчивости в условиях ограниченного количества помеченных данных.

Актуальным направлением является комбинация статистических подходов с методами машинного обучения, где статистические показатели используются как дополнительные признаки в более сложных моделях.

1.2.3 Машинное обучение

Со временем в системах обнаружения был осуществлен переход от правил на основе жесткой логики к адаптивным моделям машинного обучения, которые способны анализировать сложные нелинейные взаимосвязи в данных. Данная

трансформация обусловлена тремя взаимосвязанными факторами: экспоненциальным ростом объема транзакционных данных, усложнением схем мошенничества и необходимостью прогнозирования ранее неизвестных угроз.

Методы машинного обучения позволяют строить предиктивные модели, которые на основе исторических данных формируют решающие правила для классификации транзакций. Эти модели демонстрируют эффективность при работе с разнородными признаками, включая как количественные параметры (сумма операции, временные характеристики), так и категориальные переменные (тип платежного инструмента, географическая локация). Ключевая оптимизационная задача заключается в одновременной минимизации ложноположительных срабатываний и максимизации полноты обнаружения.

Принципиальное преимущество машинного обучения заключается в способности выявлять сложные многомерные зависимости, остающиеся за пределами возможностей традиционных статистических методов. Адаптивная природа данных алгоритмов обеспечивает возможность непрерывного обучения на новых данных, что критически важно для противодействия эволюционирующим схемам мошенничества. Кроме того, машинное обучение позволяет эффективно обрабатывать высокоразмерные данные, анализируя одновременно сотни взаимосвязанных параметров в условиях жестких временных ограничений, характерных для систем реального времени.

Однако применение машинного обучения сопряжено с рядом методологических и практических сложностей. Основной парадокс заключается в необходимости балансировки между предиктивной мощностью сложных моделей и требованием интерпретируемости решений, что особенно актуально для регулируемого банковского сектора. Данное противоречие частично разрешается через разработку гибридных систем, сочетающих преимущества машинного обучения с прозрачностью экспертных правил.

Следует отметить существенные ограничения рассматриваемого подхода. Во-первых, эффективное обучение моделей требует наличия репрезентативных выборок данных достаточного объема, что создает барьеры для небольших финансовых организаций. Во-вторых, многие современные алгоритмы (в частности, глубокие нейронные сети) страдают проблемой «черного ящика», затрудняющей интерпретацию принимаемых решений. В-третьих, существует риск переобучения и повышенные требования к вычислительным ресурсам, особенно при обработке потоковых данных в реальном времени.

Перспективы развития направления связаны с совершенствованием методов объяснимого искусственного интеллекта (Explainable AI, XAI), разработкой эффективных алгоритмов для работы с несбалансированными

данными и созданием распределенных систем обучения, позволяющих преодолеть проблему ограниченности тренировочных выборок.

1.2.4 Графовые модели

Графовые модели (Graph-based Models) представляют собой важный класс методов анализа данных, нашедший широкое применение в задачах финансового мониторинга благодаря способности выявлять сложные сетевые взаимосвязи. С формальной точки зрения, граф G определяется как упорядоченная пара (V, E), где V – множество вершин (узлов), а E – множество ребер (связей). В контексте обнаружения мошеннических транзакций вершины обычно соответствуют субъектам (физическим и юридическим лицам) и объектам (транзакциям, счетам), а ребра отражают различные типы взаимодействий между ними.

Финансовые графы обладают рядом специфических топологических свойств, требующих специализированных подходов к анализу. Они демонстрируют высокий коэффициент кластеризации и свойства «малого мира», где большинство вершин соединены короткими путями, при этом содержат немногочисленные узлы (хабы) с высокой степенью связности. Динамическая природа финансовых потоков обуславливает необходимость временного анализа, где граф структурно видоизменяется в дискретные моменты времени, отражая появление новых транзакций.

В настоящее время в анализе финансовых графов используются как классические алгоритмы, так и методы машинного обучения. Среди классических методов можно выделить обнаружение сообществ, вычисление метрик центральности, тогда как в машинном обучении применяются различные архитектуры графовых нейронных сетей (Graph Neural Networks, GNN), методы распределенного представления вершин (Graph Embedding) и временные графовые модели для анализа эволюции сетей.

Графовые модели, демонстрируя значительную эффективность с повышением детекции сложных схем по сравнению с традиционными подходами, тем не менее сталкиваются с рядом существенных методологических и практических вызовов. Вычислительная сложность обработки крупномасштабных графов требует применения распределенных алгоритмов типа Pregel и GraphX, использования эффективных методов выборки, включая случайные блуждания и метод лесных пожаров, а также задействования специализированного аппаратного обеспечения в виде графических и тензорных процессоров.

Проблема интерпретируемости решений, критически важная для соответствия регуляторным требованиям, обуславливает необходимость

разработки продвинутых методов визуализации сложных сетевых структур, внедрения техник объяснимого искусственного интеллекта таких как GNNExplainer и PGExplainer, а также создания гибридных систем, сочетающих правила и машинное обучение. Ресурсные требования к внедрению включают значительные вычислительные мощности, наличие узкоспециализированных компетенций и развертывание инфраструктуры для работы с графовыми базами данных.

Значительное внимание уделяется разработке инкрементальных алгоритмов потоковой обработки данных, технологий федерированного обучения для распределенных графовых структур, а также методов компрессии и аппроксимации графова также исследований потенциала квантовых алгоритмов для анализа графов. Экспериментальные исследования подтверждают, что несмотря на существенные ресурсные затраты, графовые подходы обеспечивают качественный прорыв в детекции сложных сетевых схем мошенничества, что делает их незаменимым компонентом современных систем финансового мониторинга.

1.3 Алгоритмы машинного обучения для задачи классификации

С помощью методов машинного обучения в задачах выявления аномальных транзакций строятся классификаторы, способные различать легальные и нелегальные операции на основе исторических данных. Специфика задачи обусловлена экстремальным дисбалансом классов (обычно менее 1% мошеннических случаев), концептуальным дрифтом и необходимостью обработки в режиме реального времени. Формально задача ставится как бинарная классификация $y = f(x)$, где $x \in R^d$ – вектор признаков транзакции, $y \in \{0,1\}$ – метка класса.

1.3.1 Логистическая регрессия

Логистическая регрессия (Logistic Regression) – это статистический метод классификации, в основе которого лежит сигмоидная функция для преобразования линейной комбинации входных признаков в значение вероятности принадлежности к классу. Для бинарной классификации модель задается уравнением:

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

где $P(y = 1|X)$ является вероятностью того, что транзакция является мошеннической ($y = 1$);

$X = (x_1, x_2, \dots, x_n)$ — вектор входных признаков (например, сумма, время, тип устройства);

$\beta_0, \beta_1, \dots \beta_n$ — коэффициенты модели, которые подбираются в процессе обучения.

После расчета вероятности модель принимает решение о классификации транзакции. Обычно используется пороговое значение, например, 0.5:

– Если $P(y = 1|X) > 0.5$, то транзакция классифицируется как мошенническая.

– $P(y = 1|X) \leq 0.5$, то транзакция считается легальной.

Однако в задачах с несбалансированными данными пороговое значение может быть скорректировано для минимизации ложных срабатываний (false positives) или ложных пропусков (false negatives).

Логистическая регрессия особенно полезна для анализа транзакционных данных благодаря своей способности работать с числовыми и категориальными признаками.

Ключевым преимуществом логистической регрессии является высокая интерпретируемость модели, обусловленная линейной природой алгоритма. Коэффициенты $\beta_0, \beta_1, \dots \beta_n$ количественно отражают влияние каждого признака на вероятность отнесения транзакции к мошеннической. В частности, положительное значение коэффициента при признаке "сумма транзакции" свидетельствует о возрастании вероятности мошенничества с увеличением суммы операции. Возможность обработки большого количества данных предоставляется путем снижения сложности алгоритма ($O(nd)$ для n наблюдений и d признаков), что критически важно для систем онлайн-мониторинга.

Несмотря на чувствительность к дисбалансу классов, характерному для задач детекции мошенничества, метод демонстрирует хорошую адаптивность за счет применения техник взвешивания классов или оптимизации порога классификации. Простота реализации и отсутствие необходимости сложной настройки гиперпараметров делают логистическую регрессию доступным инструментом для организаций с ограниченными ресурсами.

Однако метод имеет существенные ограничения, связанные с базовым предположением о линейной зависимости между логарифмом отношения шансов и признаками. В случаях сложных нелинейных взаимосвязей, характерных для многих сценариев мошенничества, предсказательная способность модели существенно снижается. Чувствительность к шуму в данных, проявляющаяся в искажении оценок коэффициентов при наличии аномальных значений признаков, требует тщательной предварительной

обработки данных, включая нормализацию числовых переменных и корректное кодирование категориальных признаков.

Важным ограничением является зависимость эффективности модели от репрезентативности обучающей выборки. Поскольку логистическая регрессия экстраполирует закономерности, выявленные в исторических данных, она может демонстрировать низкую эффективность против новых, ранее не встречавшихся схем мошенничества. Такое ограничение особенно заметно в условиях быстрой эволюции методов финансового мошенничества.

1.3.2 Случайный лес

Случайный лес (Random Forest) – метод машинного обучения, являющийся ансамблевым, то есть подразумевает объединение множества деревьев решений с последующим агрегированием их предсказаний. Данный алгоритм широко используется для задач классификации и регрессии благодаря своей высокой точности, устойчивости к шуму и способности работать с большими наборами данных.

Метод демонстрирует значительную эффективность в задачах выявления мошеннических операций. Важнейшим его преимуществом является устойчивость к дисбалансу классов, достигаемая за счет механизмов балансировки весов классов и построения деревьев на сбалансированных подвыборках. Способность обрабатывать разнородные данные, включая числовые и категориальные признаки, а также устойчивость к наличию пропущенных значений делают алгоритм особенно ценным для работы с реальными транзакционными данными.

Автоматически выявляемый алгоритмом нелинейный характер зависимостей позволяет обнаруживать сложные мошеннические схемы. Немаловажным свойством является встроенный механизм оценки значимости признаков, основанный на анализе уменьшения неопределенности при разбиениях по каждому признаку. Такой подход позволяет не только строить предсказательные модели, но и проводить содержательное исследование факторов мошенничества.

Однако метод имеет определенные ограничения. По сравнению с линейными моделями, такими как логистическая регрессия, случайный лес обладает меньшей интерпретируемостью. При работе с большими объемами данных алгоритм может требовать значительных вычислительных ресурсов, особенно при увеличении количества деревьев в ансамбле. Кроме того, существует риск переобучения при наличии зашумления данных.

По эффективности случайный лес занимает промежуточное положение между простыми линейными моделями и сложными алгоритмами градиентного

бустинга. По показателям точности метод, как правило, превосходит логистическую регрессию, но несколько уступает современным реализациям градиентного бустинга. В отличие от нейросетевых подходов, случайный лес сохраняет определенный уровень интерпретируемости признаков.

В практическом применении метод случайного лес особенно эффективен на начальных этапах разработки систем обнаружения мошенничества, когда требуется в короткий срок получить качественное решение с приемлемым уровнем точности. В промышленных системах модель может использоваться в сочетании с другими алгоритмами, например, в роли фильтра для предварительного отбора подозрительных транзакций.

1.3.3 Нейронные сети

Нейронные сети (Neural Network) занимают особое положение среди методов машинного обучения благодаря своей способности выявлять сложные нелинейные зависимости в данных. В области финансового мониторинга данное свойство имеет значимую роль, поскольку мошеннические схемы часто характеризуются неочевидными паттернами, которые трудно формализовать с помощью традиционных подходов. Математически процесс преобразования признаков в нейронной сети можно представить как композицию нелинейных функций, где каждый последовательный слой формирует все более абстрактные представления входных данных. Такая иерархическая структура позволяет автоматически извлекать значимые признаки без явного их задания.

Исследования в области применения нейросетевых детекторов мошенничества концентрируются на разработке специализированных архитектур, адаптированных к особенностям финансовых данных. Рекуррентные нейронные сети, в частности LSTM и GRU, демонстрируют высокую эффективность при анализе временных последовательностей транзакций, позволяя учитывать долгосрочные зависимости в поведении клиентов. Временные сверточные сети (TCN) предлагают альтернативный подход, сочетающий преимущества сверточных архитектур с возможностью обработки последовательностей переменной длины.

Автоэнкодерные архитектуры эффективны в задачах обнаружения аномалий, где модель обучается восстанавливать нормальные транзакции, а мошеннические операции идентифицируются по высокой ошибке реконструкции. Развитием данного направления стали вариационные автоэнкодеры, которые позволяют не только детектировать аномалии, но и моделировать распределение нормальных транзакций.

Обучение нейросетевых моделей для задач финансового мониторинга сопряжено с рядом методологических сложностей. Высокая

несбалансированность классов требует применения специализированных подходов к обучению, которые увеличивают вес сложных примеров в процессе оптимизации.

Актуальным направлением развития нейросетевых подходов является разработка методов, ориентированных на оптимизацию как статистических метрик, так и экономических показателей. Подходы, основанные на обучении с подкреплением, позволяют явно учитывать стоимость различных типов ошибок классификации, и поэтому активно используются при работе в условиях ограниченных ресурсов на расследование подозрительных операций.

Несмотря на сложность нейросетевых моделей, методы объяснимого искусственного интеллекта позволяют в определенной степени преодолеть проблему низкой трактуемости результатов работы. Локальные методы интерпретации, такие как LIME и SHAP, обеспечивают понимание отдельных предсказаний модели, в то время как анализ механизмов внимания в трансформерных архитектурах дает представление о значимости различных признаков. Развитие методов визуализации скрытых представлений, включая T-SNE проекции и активационные карты, способствует лучшему пониманию работы моделей.

Исследования на стандартных наборах данных демонстрируют превосходство нейросетевых подходов по сравнению с традиционными методами. Трансформерные архитектуры показывают точность, превосходящую результаты аналогичных показателей для более простых моделей. Однако следует учитывать, что достижение положительного итога требует значительных вычислительных ресурсов и тщательной настройки гиперпараметров.

1.3.4 Градиентный бустинг

Градиентный бустинг представляет собой семейство ансамблевых алгоритмов, основанных на последовательной компенсации ошибок предшествующих моделей. В отличие от параллельной архитектуры случайного леса, градиентный бустинг реализует итеративный подход, где каждая новая модель строится для коррекции остаточных ошибок предыдущей итерации. Математически процесс может быть описан следующим образом:

$$f(x) = f_0(x) + \eta \Sigma_i h_i(x)$$

где η – коэффициент обучения, h_i – слабые предсказатели, а f_0 – начальное приближение.

Градиентный бустинг строится на основе последовательного добавления слабых моделей (например, деревьев решений), каждая из которых фокусируется на исправлении ошибок предыдущих. Итоговое решение формируется как взвешенная сумма результатов работы всех моделей.

При детекции мошеннических транзакций современные реализации градиентного бустинга, как, например, XGBoost, LightGBM, CatBoost, демонстрируют особую эффективность. Способность алгоритмов автоматически обрабатывать разнородные признаки, включая временные метки и категориальные переменные, устраняет необходимость сложной предварительной обработки данных. Встроенные механизмы регуляризации (L-нормы, ограничение глубины деревьев) обеспечивают устойчивость к переобучению, что критически важно при работе с зашумленными финансовыми данными.

Особого внимания заслуживает способность алгоритма к адаптивной обработке несбалансированных выборок. В отличие от простых методов взвешивания классов, градиентный бустинг реализует динамическую коррекцию весов объектов через фокусную функцию потерь, что позволяет точнее выделять редкие случаи мошенничества. Экспериментальные исследования показывают, что данный подход обеспечивает увеличение метрики recall на 15-20% по сравнению с традиционными методами балансировки.

Несмотря на сложность ансамблевой структуры, метод предоставляет инструменты для анализа значимости признаков через показатели среднего улучшения точности при использовании самого признака, частоты использования в разбиениях и количество его вхождений в деревья. Это позволяет не только строить предиктивные модели, но и выявлять ключевые факторы риска, что важно для регуляторной отчетности и совершенствования правил мониторинга.

Основные вычислительные сложности механизмов градиентного бустинга связаны с необходимостью процессинга потоковых данных в режиме реального времени, что постепенно решается в рамках исследований в области инкрементального обучения и распределенных вычислений. Перспективным направлением является интеграция градиентного бустинга с методами глубокого обучения, где бустинг используется для финальной агрегации признаков, извлеченных нейросетевыми архитектурами.

ГЛАВА 2
РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Постановка задачи и выбранная реализация

Цель дипломной работы – обучить моделей машинного обучения с использованием разнообразных методов для определения нелегальных финансовых транзакций и сравнить их производительность и точность.

Среди видов мошенничества для решения поставленной задачи было выбрано определение мошеннических транзаций по кредитным картам, что обусловлено распространенностью проблемы относительно других видов мошенничества, а также наиболее высоким потенциалом для применения методов машинного обучения в связанной прикладной задаче.

В качестве языка программирования был выбран язык Python, а также вспомогательные библиотеки для работы с данными, методами машинного обучения и визуализации:

– scikit-learn предоставляет инструменты для предобработки данных, построения моделей классификации, регрессии, кластеризации и оценки их качества;

– matplotlib используется для построения графиков и визуализации данных в Python, предоставляет широкие возможности для создания различных типов диаграмм и настройки их внешнего вида;

– seaborn, основанная на matplotlib, используется для визуализации данных, а именно позволяет создавать более сложные статистические графики.

2.2 База данных

Набор данных содержит информацию о транзакциях, совершенных европейскими держателями кредитных карт в сентябре 2013 года [11]. В нем представлены операции за двухдневный период, включающие в общей сложности 284 807 записей, из которых лишь 492 относятся к мошенническим. Это делает набор данных крайне несбалансированным: мошеннические операции составляют всего 0,172% от общего числа транзакций.

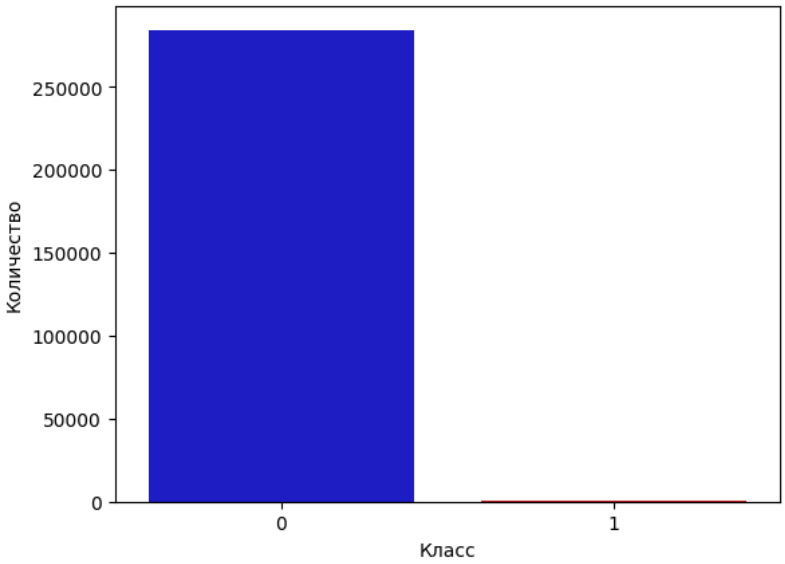

Рисунок 2.1 – Количество записей по каждой из меток класса

Все входные параметры, являющиеся числовыми в представленном датасете, были предварительно преобразованы с применением метода главных компонент для снижения размерности. Кроме того, используемые данные были анонимизированы и обфусцированы для обеспечения конфиденциальности. Исходные характеристики и дополнительная справочная информация о данных недоступны. Ниже представлены характеристики, которые не подвергались преобразованию:

– продолжительность транзакции в секундах Time;

– сумма операции Amount;

– метка класса, указывающая на принадлежность транзакции к легальной (Class 0) или мошеннической (Class 1).

Такая структура данных позволяет проводить анализ ключевых признаков, сохраняя при этом конфиденциальность исходной информации.

2.3 Анализ набора данных

Предварительный анализ данных является обязательным этапом создания моделей машинного обучения, обеспечивающим понимание их структуры, выявление проблем и их подготовку для эффективного моделирования. На данном этапе исследуются распределения признаков, их взаимосвязи с целевой переменной и наличие аномалий, пропусков и выбросов. Особое внимание уделяется оценке баланса классов, поскольку несбалансированные данные могут привести к смещению модели [9]. Анализ также позволяет определить необходимость преобразований (масштабирования или кодирование категориальных переменных) и выявить наиболее весомые признаки для

улучшения качества модели. Таким образом, предварительный анализ данных обеспечивает надежность построения точных и легко интерпретируемых моделей.

Во фрагменте кода ниже происходит визуализация плотностей распределений по сумме и времени продолжительности транзакций для двух классов: легальных транзакций (класс 0) и мошеннических транзакций (класс 1).

```
fig, ax = plt.subplots(1, 2, figsize=(18,4))
amount_val = df['Amount'].values
time_val = df['Time'].values
sns.distplot(amount_val, ax=ax[0], color='r',
axlabel='Сумма')
ax[0].set_xlim([min(amount_val), max(amount_val)])
ax[0].set_ylabel('Плотность')
sns.distplot(time_val, ax=ax[1], color='b',
axlabel='Время')
ax[1].set_xlim([min(time_val), max(time_val)])
ax[1].set_ylabel('Плотность')
plt.show()
```

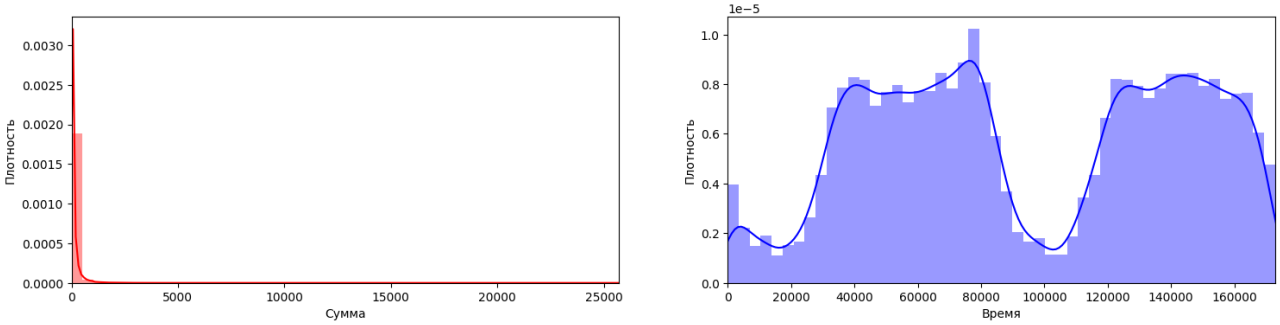

Рисунок 2.2 – Плотности распределения параметров

В данном фрагменте кода создаются два графика, демонстрирующих динамику сумм транзакций по часам для двух классов: легальных транзакций (класс 0) и мошеннических транзакций (класс 1).

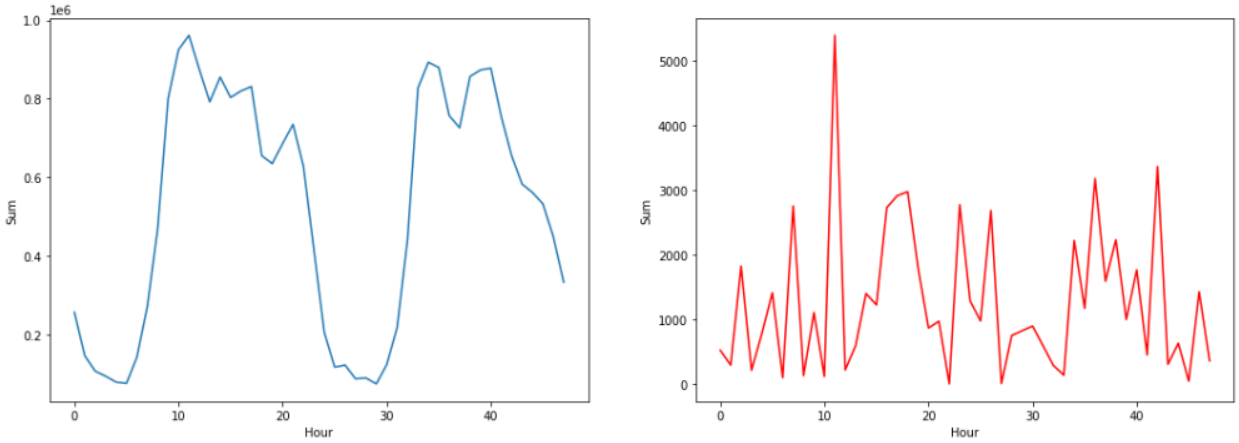

Рисунок 2.3 – Распределение транзакций по сумме переведенных средств

Правый график отображает мошеннические транзакции.

С помощью такого отображения транзакций можно сравнить, как меняются суммы транзакций в зависимости от времени суток для двух классов, и выявить потенциальные паттерны, например, временные интервалы, когда происходят аномально высокие мошеннические операции.

В следующем фрагменте происходит визуализация динамики количества транзакций по часам для двух классов.

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(18,6))
s = sns.lineplot(ax = ax1, x="Hour", y="Transactions",
data=df.loc[df.Class==0])
s = sns.lineplot(ax = ax2, x="Hour", y="Transactions",
data=df.loc[df.Class==1], color="red")
plt.suptitle("Total Number of Transactions")
plt.show();
```

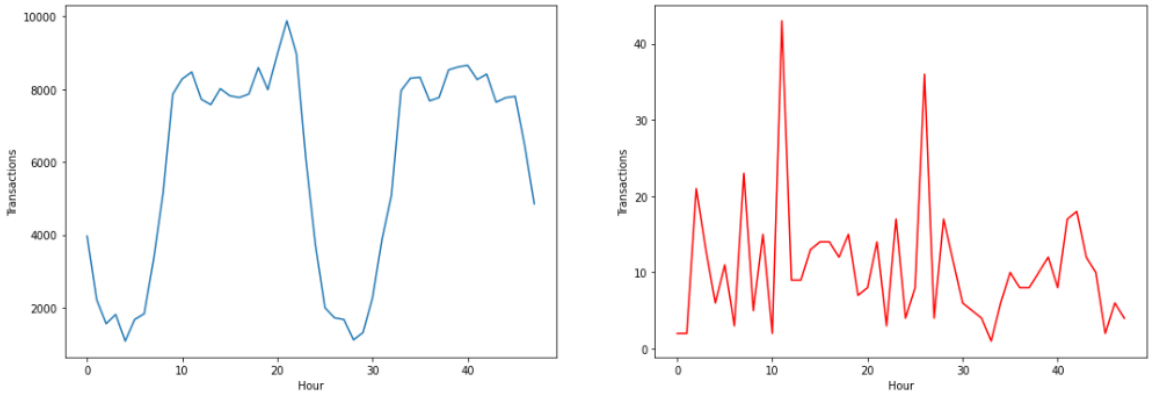

Рисунок 2.4 – Распределение транзакций по сумме переведенных средств

Левый график соответствует легальным транзакциям, правый – нелегальным.

Данный фрагмент кода создает интерактивный график для визуализации сумм мошеннических транзакций в зависимости от времени их совершения.

```
fraud = data_df.loc[data_df['Class'] == 1]
trace = go.Scatter(
    x = fraud['Time'],y = fraud['Amount'],
    name="Amount",
     marker=dict(
                color='rgb(238,23,11)',
                line=dict(
                    color='red',
                    width=1),
                opacity=0.5,
            ),
    text= fraud['Amount'],
    mode = "markers"
)
data = [trace]
layout = dict(title = 'Amount of fraudulent
transactions', xaxis = dict(title = 'Time [s]',
showticklabels=True), yaxis = dict(title = 'Amount'),
hovermode='closest')
fig = dict(data=data, layout=layout)
iplot(fig, filename='fraud-amount')
```

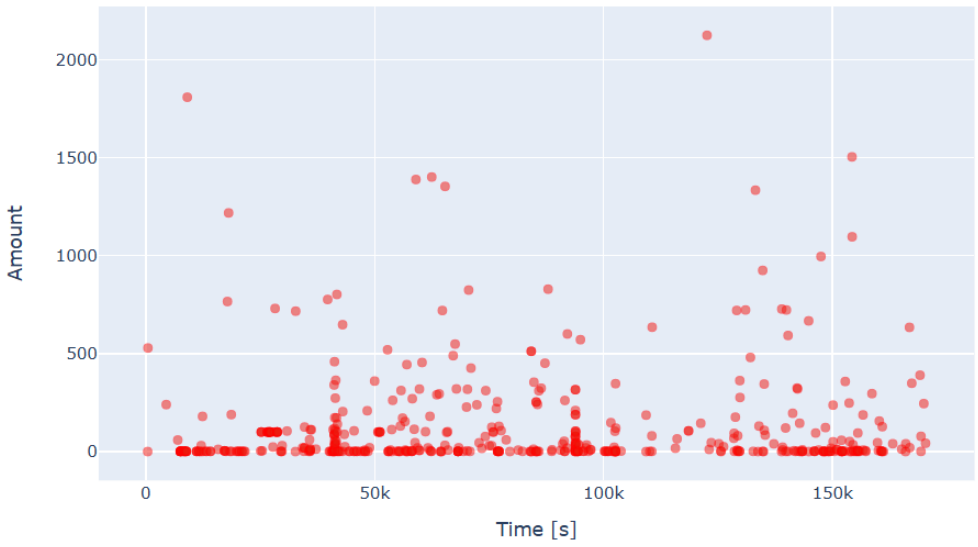

Рисунок 2.5 – Распределение транзакций по сумме от времени

Для корреляционного анализа параметров набора данных методом Пирсона используется следующий фрагмент, результатом которого является корреляционная матрица, приведенная на рисунке 2.6:

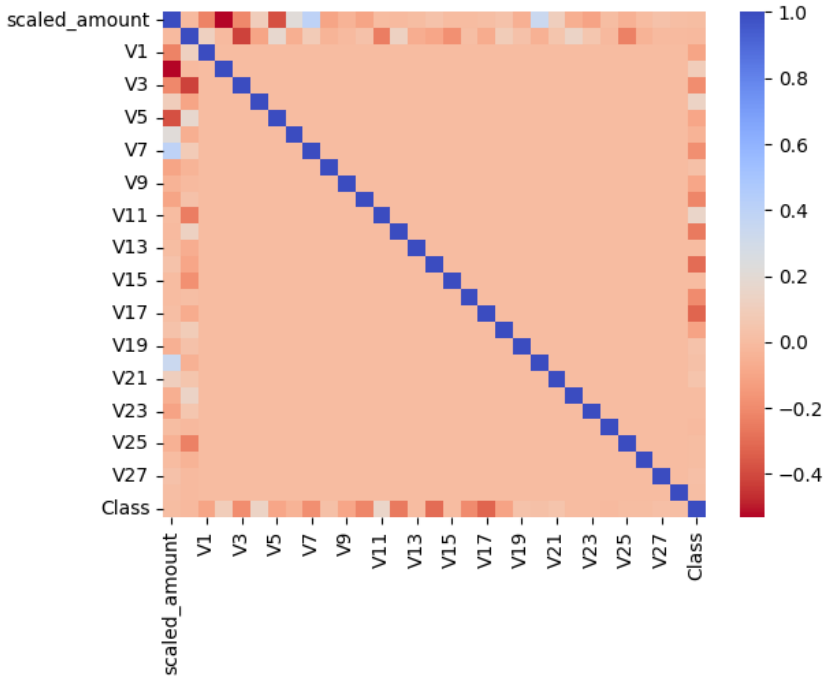

Рисунок 2.6 – Матрица корреляции параметров

```
plt.figure(figsize = (14,14))
plt.title('Credit Card Transactions features correlation
plot (Pearson)')
corr = data_df.corr()
sns.heatmap(corr,xticklabels=corr.columns,yticklabels=cor
r.columns,linewidths=.1,cmap="Reds")
plt.show()
```

Матрица отображает корреляцию между всеми парами признаков в датасете, что позволяет выявить как сильно коррелирующие признаки (как положительно, так и отрицательно), так и слабые или отсутствующие связи. Кроме того, с её помощью можно проанализировать взаимосвязь между признаками и целевой переменной для оценки их значимости в контексте моделирования.

2.4 Предобработка данных

Одним из ключевых этапов предварительной обработки данных является масштабирование признаков, таких как время совершения транзакции (Time) и сумма транзакции (Amount). Эти признаки имеют различные диапазоны значений по сравнению с другими переменными в наборе данных, что может негативно повлиять на работу алгоритмов машинного обучения. Например, методы, основанные на измерении расстояний в гиперплоскостях (например, k-ближайших соседей) или градиентный бустинг, чувствительны к различиям в масштабах признаков, что может привести к снижению их производительности.

Для устранения этой проблемы применяется стандартное масштабирование, при котором значения каждого признака преобразуются таким образом, чтобы они имели нулевое среднее значение и единичную дисперсию. Это обеспечивает равномерное влияние всех признаков на обучение модели. В результате масштабирования столбцы Time и Amount становятся сопоставимыми с другими признаками, которые уже были преобразованы с использованием метода главных компонент (PCA).

Вторым важным этапом предобработки данных является создание подвыборки для решения проблемы несбалансированности классов. Исходный набор данных характеризуется значительным дисбалансом: количество мошеннических транзакций составляет всего 0,172% от общего числа операций. Такая несбалансированность может привести к серьезным проблемам при обучении моделей.

В данном случае подвыборка представляет собой новый набор данных, в котором соотношение мошеннических и легальных транзакций составляет 50/50. Это достигается путем случайного выбора такого же количества легальных транзакций, как и мошеннических (492 примера каждого класса). Затем эти два набора объединяются в единый сбалансированный фрейм данных.

```
non_fraud_percent = round(df['Class'].value_counts()[0] /
len(df) * 100, 2)
fraud_percent = round(df['Class'].value_counts()[1] /
len(df) * 100, 2)
print(f'Без мошенничества: {non_fraud_percent}% данных')
print(f'Мошенничество: {fraud_percent}% данных')

features = df.drop(columns=['Class'])
target = df['Class']
```

```
stratified_splitter = StratifiedKFold(
    n_splits=5,
    random_state=None,
    shuffle=False
)

for tr_idx, te_idx in stratified_splitter.split(features,
target):
    print(f"Обучающая выборка: {tr_idx}, Тестовая
выборка: {te_idx}")
    X_train, X_test = features.iloc[tr_idx],
features.iloc[te_idx]
    y_train, y_test = target.iloc[tr_idx],
target.iloc[te_idx]

X_train = X_train.to_numpy()
X_test = X_test.to_numpy()
y_train = y_train.to_numpy()
y_test = y_test.to_numpy()

train_labels, train_counts = np.unique(y_train,
return_counts=True)
test_labels, test_counts = np.unique(y_test,
return_counts=True)

print('-' * 80)
print('Распределение меток:\n')
print(f"Обучающий набор: {train_counts / len(y_train)}")
print(f"Тестовый набор: {test_counts / len(y_test)}")
```

Сбалансированная подвыборка позволяет алгоритмам лучше выявлять паттерны, характерные для мошеннических транзакций, поскольку оба класса представлены равномерно. Модели, обученные на сбалансированных данных, демонстрируют более высокую чувствительность (recall) к редким событиям, что особенно важно для задач обнаружения мошенничества. Из этого следует, что использование сбалансированной подвыборки позволяет адекватно оценивать такие метрики, как F1-score, ROC AUC и полнота (recall), которые наиболее информативны для задач с несбалансированными классами.

Программный код для выполнения вышеописанной предобработки данных представлен ниже:

```
from sklearn.preprocessing import StandardScaler,
RobustScaler

standard_scaler = StandardScaler()
robust_scaler = RobustScaler()

df['normalized_amount'] = robust_scaler.fit_transform(
    df['Amount'].values.reshape(-1, 1)
)
df['normalized_time'] = robust_scaler.fit_transform(
    df['Time'].values.reshape(-1, 1)
)

df.drop(['Time', 'Amount'], axis=1, inplace=True)

temp_amount = df['normalized_amount']
temp_time = df['normalized_time']

df.drop(['normalized_amount', 'normalized_time'], axis=1,
inplace=True)
df.insert(0, 'normalized_amount', temp_amount)
df.insert(1, 'normalized_time', temp_time)

print(df.head())
```

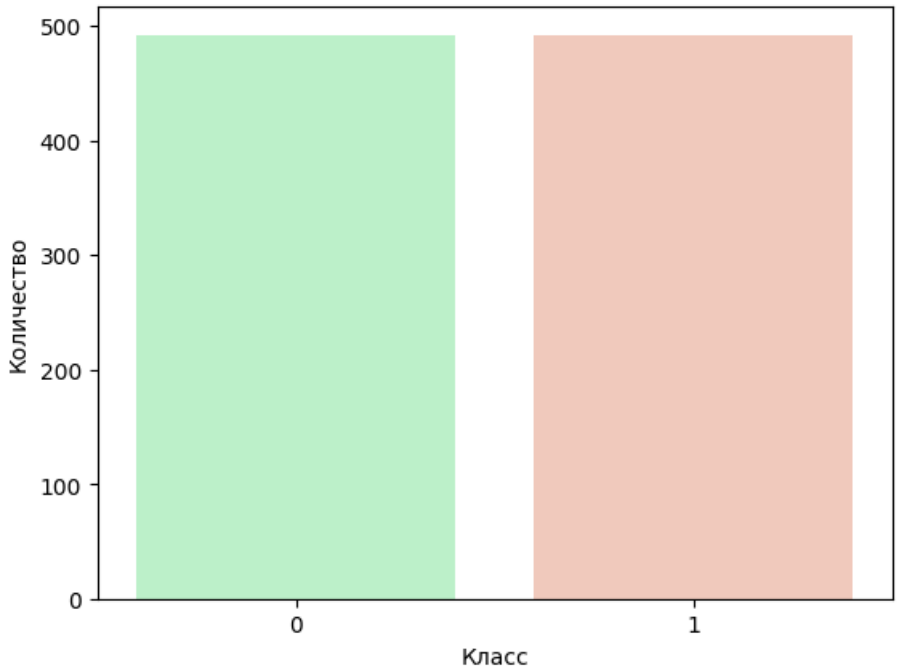

Рисунок 2.7 – Количество записей по каждой из меток класса в подвыборке

В результате выполнения этапов предобработки признаки Time и Amount были масштабированы до единого диапазона значений и создана сбалансированная подвыборка, содержащая 492 примера мошеннических транзакций и 492 примера легальных транзакций.

Корреляционная матрица для сбалансированной подвыборки представлена ниже. В сравнении с аналогичной матрицей для исходного набора данных на рисунке 6, приведенная корреляционная матрица свидетельствует наличие статистически значимых корреляций между определенными признаками и фактом мошенничества. Группа признаков, включающая V17, V12 и V10, демонстрирует отрицательную корреляцию с целевой переменной. Это означает, что уменьшение значений данных признаков статистически связано с повышением вероятности того, что транзакция является мошеннической. Напротив, признаки V17, V11 и V19 показывают положительную корреляционную зависимость – рост их значений соответствует увеличению вероятности мошеннической операции.

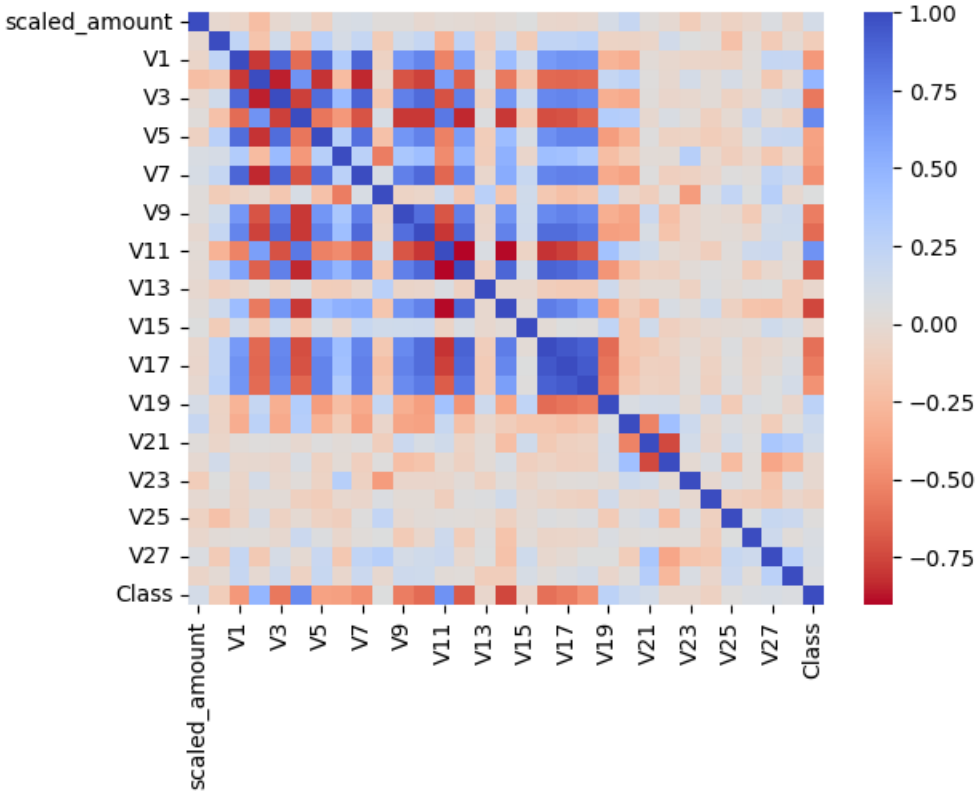

Рисунок 2.8 – Матрица корреляции для подвыборки

Таким образом, этапы масштабирования и создания сбалансированной подвыборки являются критически важными для успешного анализа данных и разработки эффективных систем обнаружения мошеннических транзакций. Эти шаги позволяют минимизировать влияние несбалансированности данных и повысить качество моделей машинного обучения.

2.5 Снижение размерности и кластеризация

Применение методов снижения размерности для визуализации и предварительного анализа данных транзакций представляет значительный интерес в задачах обнаружения мошеннических операций. Как показывают исследования, такие методы позволяют выявить скрытые структуры данных и оценить принципиальную возможность разделения классов с помощью машинного обучения. Среди наиболее распространенных подходов следует выделить t-SNE (t-distributed Stochastic Neighbor Embedding), PCA (Principal Component Analysis) и Truncated SVD (Singular Value Decomposition), каждый из которых обладает характерными особенностями.

Алгоритм t-SNE демонстрирует особую эффективность при визуализации высокоразмерных данных за счет сохранения локальных расстояний между точками. В основе метода лежит вероятностный подход, который минимизирует

расхождение между распределениями расстояний в исходном и редуцированном пространствах. Как показывают эксперименты на выборках банковских транзакций, t-SNE позволяет достичь четкого разделения мошеннических и легитимных операций даже на относительно небольших выборках. Это свидетельствует о наличии выраженных кластерных структур в данных, что является важным индикатором потенциальной эффективности последующих классификационных моделей. Однако следует учитывать, что вычислительная сложность алгоритма существенно выше по сравнению с линейными методами, что подтверждается экспериментальными данными: время обработки данных методом t-SNE может на порядки превышать аналогичные показатели для PCA и Truncated SVD.

Метод главных компонент (PCA), являясь линейным методом снижения размерности, обеспечивает более высокую производительность за счет преобразования данных в пространство ортогональных компонент с максимальной дисперсией. Практические результаты демонстрируют, что PCA сохраняет глобальную структуру данных, но может уступать t-SNE в способности визуализировать сложные нелинейные зависимости. Тем не менее, его вычислительная эффективность делает PCA ценным инструментом для предварительного анализа, особенно при работе с большими объемами данных.

Truncated SVD, как вариант сингулярного разложения матриц, занимает промежуточное положение между t-SNE и PCA по своим характеристикам. Метод особенно эффективен при работе с разреженными матрицами, характерными для текстовых данных, но также находит применение в задачах анализа транзакций. Эксперименты показывают, что по скорости обработки данных Truncated SVD может превосходить даже PCA, сохраняя при этом удовлетворительное качество визуализации.

Реализация трех вышеописанных подходов для набора данных транзаций представлена ниже:

```
features = new_df.drop(columns=['Class'])
target = new_df['Class']

# Визуализация с использованием T-SNE
start_time = time.time()
tsne_embedding = TSNE(
    n_components=2,
    random_state=42
).fit_transform(features.values)
tsne_time = time.time() - start_time
```

```
    print(f"Время выполнения T-SNE: {tsne_time:.2f}
секунд")

    # Анализ главных компонент (PCA)
    start_time = time.time()
    pca_projection = PCA(
        n_components=2,
        random_state=42
    ).fit_transform(features.values)
    pca_time = time.time() - start_time
    print(f"Время выполнения PCA: {pca_time:.2f} секунд")

    # Усечённое SVD
    start_time = time.time()
    svd_reduction = TruncatedSVD(
        n_components=2,
        algorithm='randomized',
        random_state=42
    ).fit_transform(features.values)
    svd_time = time.time() - start_time
    print(f"Время выполнения Truncated SVD:
{svd_time:.2f} секунд")
```

Для визуализации результатов работы алгоритмов кластеризации был использован следующий фрагмент программного кода:

```
    f, (ax1, ax2, ax3) = plt.subplots(1, 3,
figsize=(24,6))
    # labels = ['No Fraud', 'Fraud']
    f.suptitle('Clusters using Dimensionality Reduction',
fontsize=14)

    blue_patch = mpatches.Patch(color='#0A0AFF',
label='No Fraud')
    red_patch = mpatches.Patch(color='#AF0000',
label='Fraud')

    # t-SNE scatter plot
```

```
    ax1.scatter(X_reduced_tsne[:,0], X_reduced_tsne[:,1],
c=(y == 0), cmap='coolwarm', label='No Fraud',
linewidths=2)
    ax1.scatter(X_reduced_tsne[:,0], X_reduced_tsne[:,1],
c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
    ax1.set_title('t-SNE', fontsize=14)

    ax1.grid(True)
    ax1.legend(handles=[blue_patch, red_patch])

    # PCA scatter plot
    ax2.scatter(X_reduced_pca[:,0], X_reduced_pca[:,1],
c=(y == 0), cmap='coolwarm', label='No Fraud',
linewidths=2)
    ax2.scatter(X_reduced_pca[:,0], X_reduced_pca[:,1],
c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
    ax2.set_title('PCA', fontsize=14)

    ax2.grid(True)
    ax2.legend(handles=[blue_patch, red_patch])

    # TruncatedSVD scatter plot
    ax3.scatter(X_reduced_svd[:,0], X_reduced_svd[:,1],
c=(y == 0), cmap='coolwarm', label='No Fraud',
linewidths=2)
    ax3.scatter(X_reduced_svd[:,0], X_reduced_svd[:,1],
c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)
    ax3.set_title('Truncated SVD', fontsize=14)

    ax3.grid(True)
    ax3.legend(handles=[blue_patch, red_patch])
    plt.show()
```

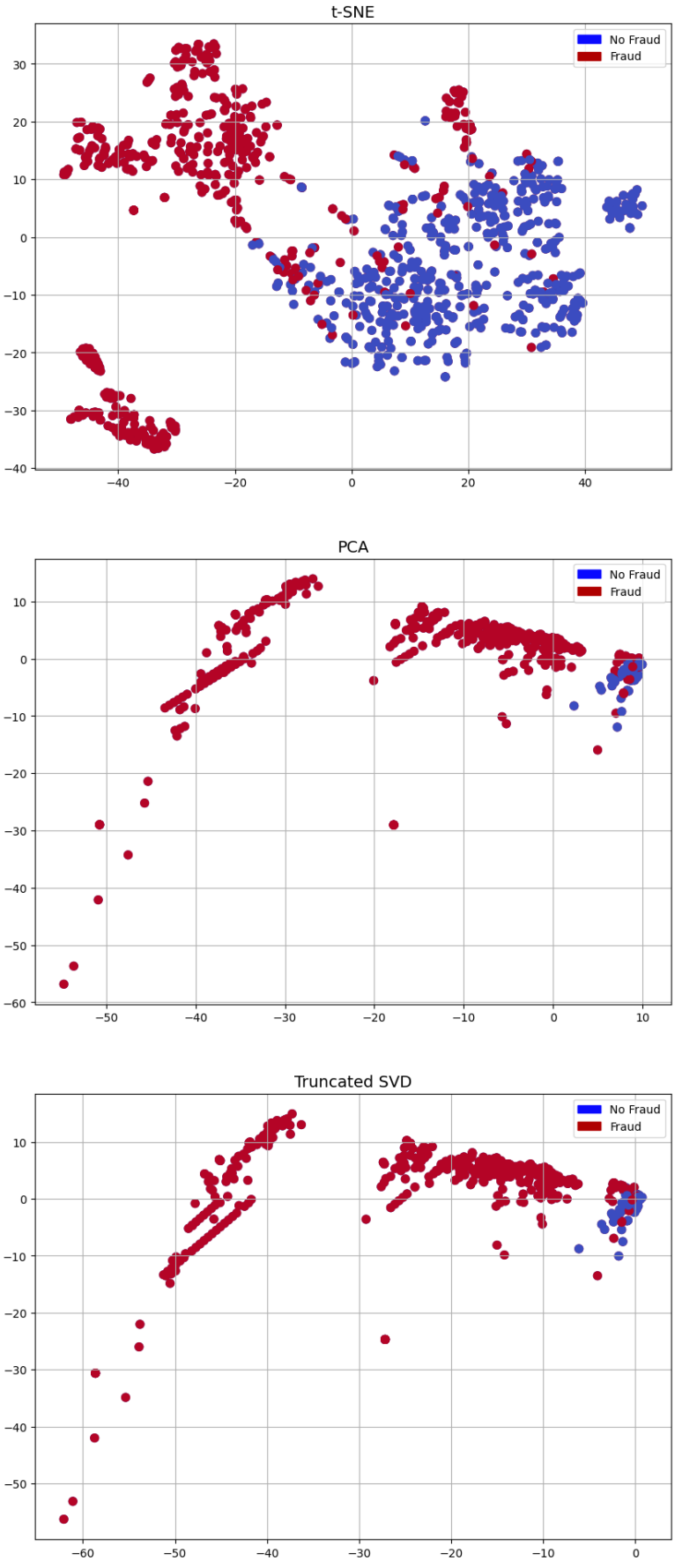

Рисунок 2.9 – Результаты кластеризации

Сравнительный анализ результатов применения этих методов к данным банковских транзакций подтверждает наличие устойчивых кластерных структур, соответствующих различным типам операций. Наблюдаемое разделение классов в редуцированном пространстве служит важным индикатором того, что признаки, используемые для описания транзакций, содержат достаточную информацию для последующего построения эффективных классификационных моделей. При этом следует отметить, что визуализация с помощью t-SNE, несмотря на свою ресурсоемкость, предоставляет наиболее четкую картину распределения данных, что может быть особенно ценным на этапе предварительного анализа и формирования гипотез.

2.6 Построение моделей

В рамках первичного исследования эффективности различных подходов машинного обучения для задачи классификации мошеннических транзакций был проведен сравнительный анализ шести классификационных алгоритмов различной сложности. Экспериментальная выборка включала как базовые методы, так и современные ансамблевые подходы:

– логистическая регрессия (LogisticRegression, LogReg) – параметрический линейный классификатор, основанный на максимизации функции правдоподобия;

– метод k-ближайших соседей (KNearsNeighborsClassifier, kNN) – непараметрический алгоритм, использующий локальные закономерности в пространстве признаков;

– метод опорных векторов (SupportVectorClassifier, SVC) – алгоритм максимального зазора с ядерными преобразованиями;

– дерево решений (DecisionTreeClassifier, DTC) – рекурсивный алгоритм разбиения пространства признаков;

– случайный лес (RandomForestClassifier, RFC) – бэггинговый ансамбль решающих деревьев;

– градиентный бустинг (XGBClassifier, XGB) – бустинговый ансамбль с поэтапной оптимизацией [6].

```
model_collection = {
    "LogReg": LogisticRegression(),
    "kNN": KNeighborsClassifier(),
    "SVC": SVC(),
    "DTC": DecisionTreeClassifier(),
    "RFC": RandomForestClassifier(),
```

```
    "XGBC": XGBClassifier()
}

for model_name, model in model_collection.items():
    model.fit(X_train, y_train)
    cv_scores = cross_val_score(
        estimator=model,
        X=X_train,
        y=y_train,
        cv=5
    )
```

Результаты эксперимента показали, что даже без тонкой настройки гиперпараметров, ансамблевые методы (RandomForest и XGBoost) демонстрируют статистически значимое превосходство по метрике точности над базовыми алгоритмами. Это объясняется их способностью автоматически учитывать нелинейные зависимости, устойчиво работать с зашумленными данными и эффективно обрабатывать категориальные признаки.

Таблица 2.1 – Результаты кросс-валидации

Классификатор	**Точность (accuracy score)**
LogisticRegression	94.0%
KNeighborsClassifier	94.0%
SVC	94.0%
DecisionTreeClassifier	91.0%
RandomForestClassifier	95.0%
XGBClassifier	95.0%

2.7 Оптимизация гиперпараметров моделей

После получения предварительных результатов определения мошеннических операций с помощью набора сконфигурированных по умолчанию классификаторов, была проведена оптимизация параметров с использованием метода GridSearchCV. Данный подход позволяет систематически исследовать пространство параметров для нахождения оптимальной конфигурации каждой модели.

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

model_hyperparams = {
    "LogisticRegression": {
        "penalty": ['l1', 'l2'],
        'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]
    },
    "KNeighbors": {
        "n_neighbors": [2, 3, 4, 5],
        'algorithm':  ['auto',  'ball_tree',  'kd_tree',
'brute']
    },
    "SVC": {
        'C': [0.5, 0.7, 0.9, 1],
        'kernel': ['sigmoid','rbf','linear','poly']
    },
    "DecisionTree": {
        "criterion": ["gini", "entropy"],
        "max_depth": [2, 3, 4],
        "min_samples_leaf": [5, 6, 7]
    },
    "RandomForest": {
        "criterion": ["gini", "entropy"],
        "max_depth": [2, 3, 4],
        "min_samples_leaf": [5, 6, 7]
    },
    "XGBoost": {
        "learning_rate": [0.01, 0.05, 0.1],
        "max_depth": [2, 3, 4],
        "gamma": [0, 0.1],
        "reg_alpha": [0, 0.25, 0.5],
        "reg_lambda": [1, 1.25, 1.5],
        "scale_pos_weight": list(range(1, 100, 10))
    }
}
```

```
best_models = {}

for model_name, params in model_hyperparams.items():

    if model_name == "LogisticRegression":
        model = LogisticRegression()
    elif model_name == "KNeighbors":
        model = KNeighborsClassifier()
    elif model_name == "SVC":
        model = SVC()
    elif model_name == "DecisionTree":
        model = DecisionTreeClassifier()
    elif model_name == "RandomForest":
        model = RandomForestClassifier()
    else:
        model = XGBClassifier(eval_metric="aucpr")

    grid_search = GridSearchCV(
        estimator=model,
        param_grid=params,
        cv=5,
    )

    grid_search.fit(X_train, y_train)
    best_models[model_name] = grid_search.best_estimator_
```

Для каждого классификатора был эмпирически определен набор параметров, которые в большей степени оказывают влияние на процесс принятия решения, с диапазонами допустимых значений. В результате применения оптимизационного метода GridSearchCV классификаторы, которые рассматриваются ниже.

Для логистической регрессии исследовалось влияние типа регуляризации (L1 и L2) и силы регуляризации C. Анализ показал, что оптимальная конфигурация достигается при L2 и C=0.1.

При оптимизации метода k-ближайших соседей рассматривались количество соседей, алгоритмы поиска соседей. Полученные результаты демонстрируют лучшие результаты при количестве соседей, равным 3, и с использованием алгоритма auto.

Классификатор метод опорных векторов показывает наиболее удачные результаты при параметре регуляризации C, равным 0.5, и линейном типе ядра.

Для классификаторов дерева решений и случайного леса рассматривались критерии разделения (энтропия и индекс Джини), максимальная глубина деревьев и минимальное количество образцов в листе. В первом случае оптимальная конфигурация включает в себя индекса Джини в качестве критерия, max_depth = 3 и min_samples_leaf = 5. Конфигурация исследуемых параметров для случайного леса аналогична, за исключением min_samples_leaf = 6.

Для алгоритма XGBoost исследовался широкий диапазон параметров, включающий уровень обучения, максимальную глубину, вес положительного класса и параметры регуляризации. По итогам оптимизации набор параметров состоит из learning_rate = 0.1, max_depth = 3, gamma = 0, reg_alpha = 0.5 и reg_lambda = 1.

По итогам кросс-валидацинного тестирования новых версий классификаторов результаты несколько изменились.

Таблица 2.2 – Результаты кросс-валидации после оптимизации

Классификатор	**Точность (accuracy score)**
LogisticRegression	94.79%
KNeighborsClassifier	94.03%
SVC	94.54%
DecisionTreeClassifier	93.01%
RandomForestClassifier	92.76%
XGBClassifier	94.79%

Стоит обратить внимание, что в данном случае имеет место переобучение моделей на валидационной выборке. Проблема переобучения решалась с помощью применения метода NearMiss в сочетании с кросс-валидацией. Этот реализует эвристический подход к андерсэмплингу, сохраняя те экземпляры мажоритарного класса, которые находятся ближе всего к представителям миноритарного класса.

```
from collections import Counter
from imblearn.pipeline import make_pipeline as
 imbalanced_make_pipeline
from imblearn.under_sampling import NearMiss
from sklearn.metrics import (precision_score,recall_score,
 f1_score, roc_auc_score)
```

```
features = df.drop('Class', axis=1)
target = df['Class']

for train_idx, test_idx in sss.split(features, target):
    print(f"Training indices: {train_idx}, Test indices:
 {test_idx}")

    X_train, X_test = features.iloc[train_idx],
 features.iloc[test_idx]
    y_train, y_test = target.iloc[train_idx],
 target.iloc[test_idx]

X_train = X_train.values
X_test = X_test.values
y_train = y_train.values
y_test = y_test.values

metrics = {
    'accuracy': [],
    'precision': [],
    'recall': [],
    'f1': [],
    'auc': []
}

X_resampled, y_resampled =
 NearMiss().fit_resample(features.values, target.values)
print(f'Resampled class distribution:
 {Counter(y_resampled)}')

for train_idx, test_idx in sss.split(X_train, y_train):
    pipeline = imbalanced_make_pipeline(
        NearMiss(sampling_strategy='majority'),
        log_reg
    )

    model = pipeline.fit(X_train[train_idx],
 y_train[train_idx])
```

```
    y_pred = model.predict(X_train[test_idx])

    metrics['accuracy'].append(
        pipeline.score(original_Xtrain[test_idx],
original_ytrain[test_idx])
    )
    metrics['precision'].append(
        precision_score(original_ytrain[test_idx], y_pred)
    )
    metrics['recall'].append(
        recall_score(original_ytrain[test_idx], y_pred)
    )
    metrics['f1'].append(
        f1_score(original_ytrain[test_idx], y_pred)
    )
    metrics['auc'].append(
        roc_auc_score(original_ytrain[test_idx], y_pred)
    )
```

Исходный датасет был разделен на матрицу признаков undersample_X и целевой вектор undersample_y, после чего применен стратифицированный алгоритм разбиения StratifiedShuffleSplit для сохранения исходного распределения классов в обучающей и тестовой выборках.

2.8 Сравнительный анализ построенных моделей

Для комплексной оценки динамики обучения различных алгоритмов машинного обучения был проведен анализ кривых обучения. Из тенденции расположения и сходимости кривых, представленных на рисунке X, можно сделать вывод, что поставленная задача классификации транзакций успешнее всего решается с помощью полученных моделей логистической регрессии, классификатора случайного леса и градиентного бустинга.

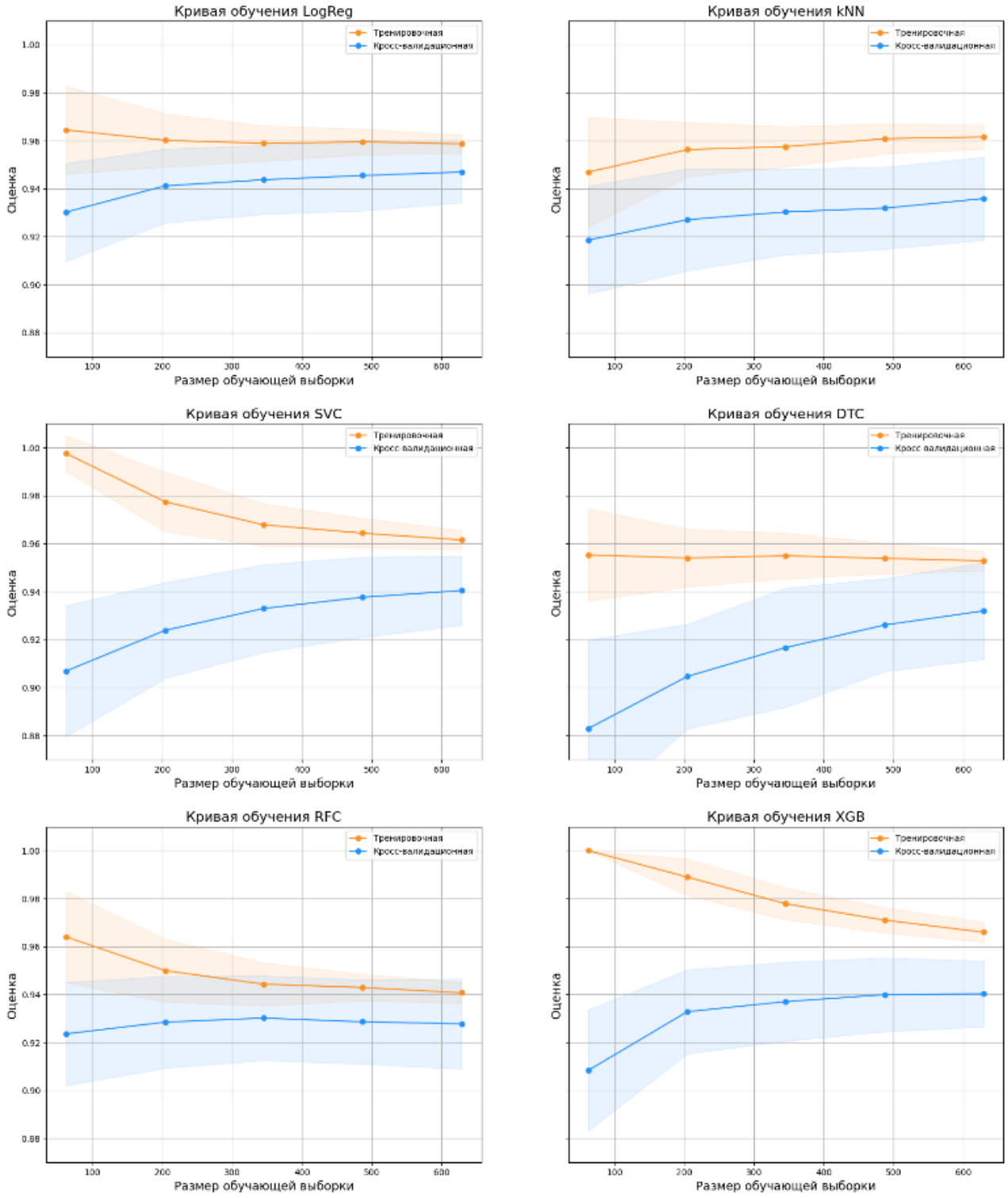

Рисунок 2.10 – Кривые обучения

График характеристических операционных кривых (Receiver Operating Characteristic curves), приведенный на рисунке X, отражает эффективность классификационных алгоритмов в задаче детекции мошеннических транзакций. По сравнительному положению ROC-кривых всех моделей относительно диагонали классификатора, их форме и значению площади под кривой AUC-

ROC, стоит выделить модель логистической регрессии, обладающей максимальным значением AUC, классификатор SVC и XGBoost.

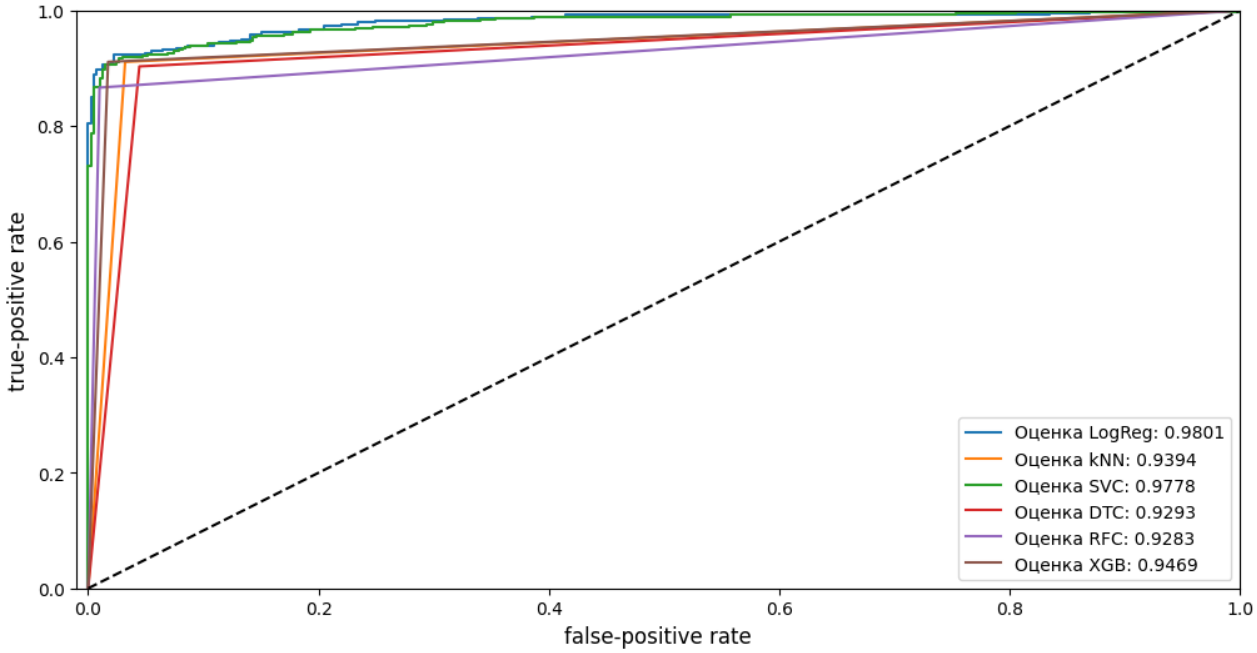

Рисунок 2.11 – Кривые ROC

Фрагмент кода ниже позволяет построить итоговую матрицу несоответствий, представленную на рисунке 2.12:

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

model_predictions = {
    "LogReg": log_reg_sm.predict(X_test),
    "kNN": knears_neighbors.predict(X_test),
    "SVC": svc.predict(X_test),
    "DTC": tree_clf.predict(X_test),
    "RFC": forest_clf.predict(X_test),
    "XGB": xgb_clf.predict(X_test)
}

confusion_matrices = {
    name: confusion_matrix(y_test, preds)
    for name, preds in model_predictions.items()
}
```

```
plt.style.use('seaborn')
fig, axes = plt.subplots(3, 2, figsize=(18, 22))
fig.suptitle('Confusion Matrix Comparison', y=1.02,
fontsize=20)
labels = ['no fraud', 'fraud']
for (model_name, cf_matrix), ax in
zip(confusion_matrices.items(), axes.flat):
    sns.heatmap(
        cf_matrix,
        ax=ax,
        annot=True,
        fmt='d',
        cmap='coolwarm_r',
        annot_kws={'size': 16},
        cbar=False
    )

    ax.set_title(model_name, fontsize=16)
    ax.set_xticklabels(labels, fontsize=14, rotation=0)
    ax.set_yticklabels(labels, fontsize=14, rotation=0)

plt.tight_layout()
plt.show()

plt.show()
```

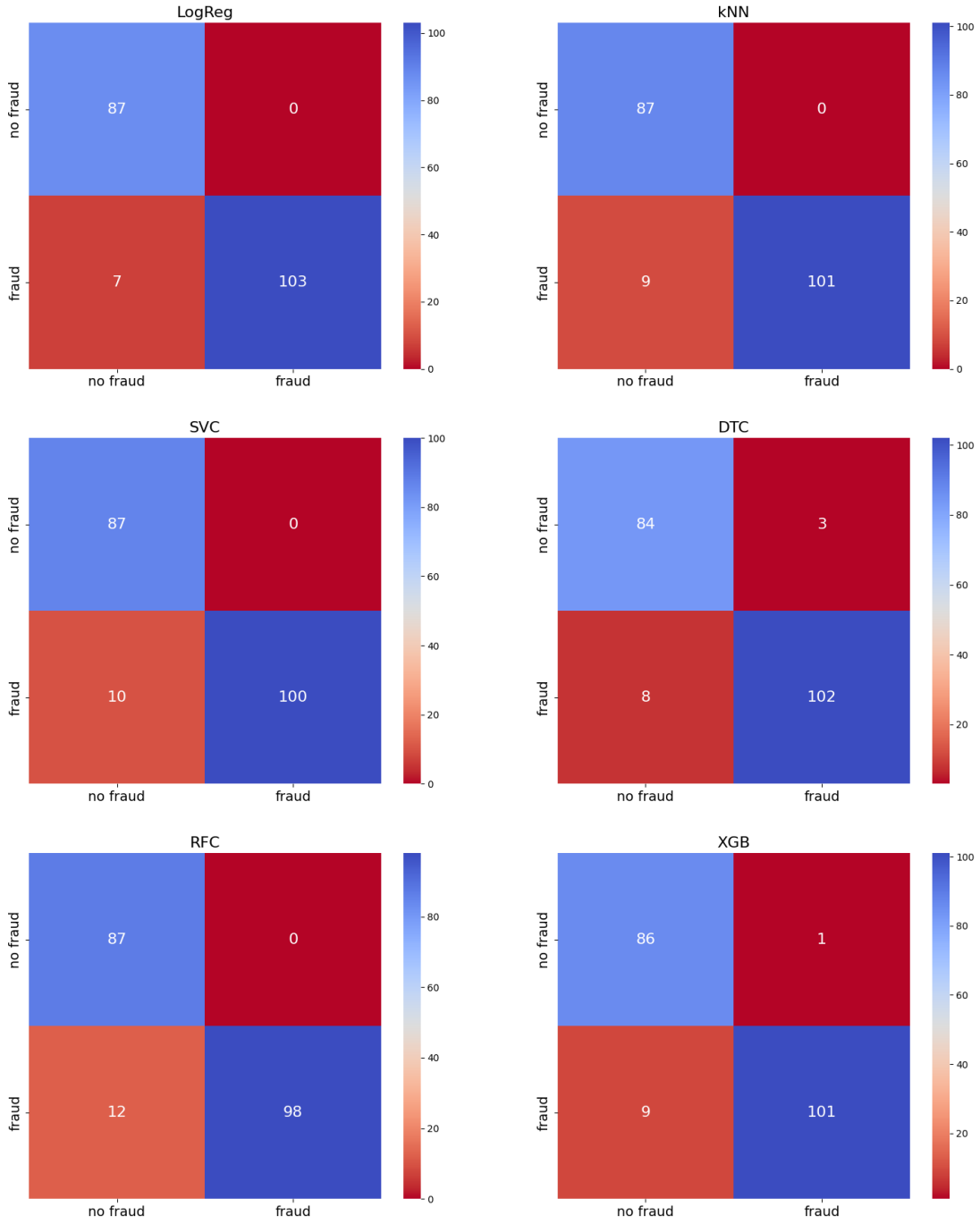

Рисунок 2.12 – Итоговая матрица несоответствий

Таблица 2.3 – Итоговая таблица результатов кросс-валидации

Классификатор	**precision, %**	**recall, %**	**F1-score, %**	**ROC-AUC, %**
Logistic Regression	96,23	97,04	96,47	98,01
k-Near Neighbor	95,09	96.0	95,11	93,94
Support Vector	95,35	95,12	95,09	97,78
Decision Tree	94,21	95,17	94,2	92,93
Random Forest	94,3	95,02	94,32	92,83
XGBoost	95,04	95,3	95,19	94,69

Представленные в таблице 2.3 результаты сравнительного анализа шести классификационных алгоритмов демонстрируют высокую эффективность всех рассмотренных моделей в задаче детекции мошеннических транзакций, что подтверждается значениями ключевых метрик, превышающими 92% по всем показателям. Логистическая регрессия показала наилучшие результаты по комплексной метрике F1-score и ROC-AUC, что свидетельствует о ее высокой сбалансированности между точностью и полнотой предсказаний. Данный факт указывает на достаточную линейную разделимость классов в пространстве признаков для данной задачи.

Метод опорных векторов продемонстрировал сравнимую с логистической регрессией эффективность, однако его несколько более низкие показатели полноты могут ограничивать применимость в сценариях, где критично минимизировать пропуск мошеннических операций. Ансамблевые методы, вопреки ожиданиям, показали несколько более скромные результаты: Random Forest и XGBoost уступили линейным моделям по всем метрикам, за исключением незначительного преимущества XGBoost в F1-score перед SVC.

Особого внимания заслуживает поведение дерева решений, которое, несмотря на простоту алгоритма, продемонстрировало достойные результаты (F1-score 94.2%), всего на 1-2% уступая более сложным ансамблевым методам. Это свидетельствует о том, что для данной задачи даже простые модели способны выявлять достаточно четкие паттерны мошеннической активности. Метод k-ближайших соседей показал наименьшую эффективность по комплексной метрике ROC-AUC, что может быть связано с чувствительностью алгоритма к дисбалансу классов и шуму в данных.

Полученные итоги позволяют сделать вывод о том, что для рассматриваемой задачи детекции мошеннических транзакций логистическая регрессия является оптимальным выбором, сочетая высокую точность, интерпретируемость и вычислительную эффективность.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была решена задача классификации для выявления мошеннических транзакций с использованием различных моделей машинного обучения. Основное внимание уделялось комплексный анализ данных с акцентом на обработку несбалансированных выборок, исследование взаимосвязей между признаками и оценку производительности различных алгоритмов.

Экспериментальные результаты продемонстрировали, что модели логистической регрессия и классификатор случайного леса показывают высокую производительность при работе со сложными зависимостями в данных. Это подтверждает их практическую применимость для задач финансового мониторинга, где критически важна способность выявлять редкие случаи мошенничества среди большого объема легальных операций.

Рассмотренные методы работы с несбалансированными данными банковских транзакций, включающие применение техник андерсэмплинга, оптимизацию гиперпараметров и комплексную оценку по соответствующим метрикам, продемонстрировали свою эффективность и применимость в решении поставленной прикладной задачи. Все исследуемые модели машинного обучения продемонстрировали показатели точности свыше 92% по ключевым оценочным метрикам, что соответствует требованиям, предъявляемым к промышленным системам финансового мониторинга.

Полученные в ходе работы результаты предполагают дальнейшие исследования по улучшению качества классификации банковских транзакций с целью выявления новых паттернов мошеннических операций и комбинирование различных моделей в единый ансамбль для повышения точности за счет агрегации преимуществ каждой из моделей.

Таким образом, проведенное исследование подтвердило высокий уровень релевантности современных алгоритмов машинного обучения для задачи обнаружения мошеннических транзакций. Развитие методов анализа данных, оптимизации моделей и внедрения новых технологий позволит создать более надежные и масштабируемые системы для защиты от мошенничества, содействуя цифровизации финансовой сферы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Adekoya F., Tagbo K. A Systematic Literature Review of Machine Learning Techniques in Financial Fraud Prevention and Detection / Adekoya F., Tagbo K. // ResearchGate [Electronic resource]. – Mode of access: https://www.researchgate.net/publication/371998161. – Date of access: 26.05.2025.

2. ACFE Occupational Fraud 2024: Report to the nation / Association of Certified Fraud Examiners [Electronic resource]. – 2024. – Mode access: https://www.acfe.com/-/media/files/acfe/pdfs/rttn/2024/2024-report-to-the-nations.pdf. – Date of access: 17.04.2025.

3. APWG Phishing Activity Trends Report Q4 2024 / Anti-Phishing Working Group [Electronic resource]. – 2024. – Mode access: https://docs.apwg.org/reports/apwg_trends_report_q4_2024.pdf. – Date of access: 18.04.2025.

4. Bhattacharya S., Jha S., Tharakunnel K., Westland C. Data mining for credit card fraud: A comparative study / Bhattacharya S., Jha S., Tharakunnel K., Westland C. // Decision Support Systems. – 2001. – P. 602–613.

5. Carminati M., Caron R., Maggi F., Ilenia E. BankSealer: A Decision Support System for Online Banking Fraud Analysis and Investigation / Carminati M., Caron R., Maggi F., Ilenia E. // Computers & Security. – 2015. – P. 380–394.

6. Chen T., Guestrin C. Anti-Money XGBoost: A Scalable Tree Boosting System / Chen T., Guestrin C. // arXiv [Electronic resource]. – 2016. – Mode access: https://arxiv.org/abs/1603.02754. – Date of access: 20.04.2025.

7. Dal Pozzolo A., Boracchi G., Caelen O., Alippi C., Bontempi G. Credit card fraud detection: a realistic modeling and a novel learning strategy / Dal Pozzolo A., Boracchi G., Caelen O., Alippi C., Bontempi G. // IEEE Transactions on Neural Networks and Learning Systems. – 2015. – P. 9–45.

8. Dal Pozzolo A., Caelen O., Le Borgne Y. A., Waterschoot S., Bontempi G. Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective / Dal Pozzolo A., Caelen O., Le Borgne Y. A., Waterschoot S., Bontempi G. // Expert Systems with Applications. – 2014. – P. 12–17.

9. Fernández A., García S., Galar M., Prati R.C., Krawczyk B., Herrera F. Learning from Imbalanced Data Sets / Fernández A., García S., Galar M., Prati R.C., Krawczyk B., Herrera F. // Springer. – 2018. – P.19-46.

10. He H., Ma Y. Imbalanced Learning: Foundations, Algorithms, and Applications / He H., Ma Y. // IEEE. – 2013. – P.135–152.

11. Machine Learning Group ULB Credit Card Fraud Detection Database, Anonymized credit card transactions labeled as fraudulent or genuine / Machine

Learning Group ULB // Kaggle [Electronic resource]. – 2018. – Mode access: https://www.kaggle.com/mlg-ulb/creditcardfraud. – Date of access: 18.04.2025.

12. Nilson Report: Card Fraud Losses Worldwide in 2023 / Nilson Report [Electronic resource]. – 2023. – Mode access: https://nilsonreport.com/articles/card-fraud-losses-worldwide-in-2023. Date of access: 17.04.2025.

13. Verizon: 2023 Data Breach Investigations Report / Verizon [Electronic resource]. – 2023. – Mode access: https://www.verizon.com/business/resources/reports/2023-data-breach-investigations-report-dbir.pdf. – Date of access: 20.04.2025.

14. Weber M., Domeniconi G., Chen J., Karl D., Bellei C., Robinson T., Leiserson C. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics / Weber M., Domeniconi G., Chen J., Karl D., Bellei C., Robinson T., Leiserson C. // arXiv [Electronic resource]. – 2019. – Mode access: https://arxiv.org/abs/1908.02591. – Date of access: 20.04.2025.

15. West J., Bhattacharya M. Intelligent Financial Fraud Detection: A Comprehensive Review / West J., Bhattacharya M. // Computers & Security. – 2016. – P. 4–39.

ПРИЛОЖЕНИЕ А

ЛИСТИНГ ФАЙЛА ANTIFRAUD-MODELS.IPYNB

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import RobustScaler

df = pd.read_csv('archive/creditcard.csv')

display(df.head())
display(df.describe())
print(f"Max null values in any column:
{df.isnull().sum().max()}")
print(f"Columns: {list(df.columns)}")

fraud_pct = df['Class'].value_counts(normalize=True) *
100
print(f"Non-Fraud: {fraud_pct[0]:.2f}%")
print(f"Fraud: {fraud_pct[1]:.2f}%")

plt.figure(figsize=(8,5))
sns.countplot(data=df, x='Class', palette=["#0101DF",
"#DF0101"])
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18,4))
sns.histplot(df['Amount'], ax=ax1, color='r', kde=True)
ax1.set(xlabel='Amount', ylabel='Density',
xlim=(df['Amount'].min(), df['Amount'].max()))
sns.histplot(df['Time'], ax=ax2, color='b', kde=True)
ax2.set(xlabel='Time', ylabel='Density',
xlim=(df['Time'].min(), df['Time'].max()))
plt.tight_layout()
plt.show()
```

```
scaler = RobustScaler()
df[['scaled_amount', 'scaled_time']] =
scaler.fit_transform(df[['Amount', 'Time']])
df = df.drop(['Amount', 'Time'], axis=1)
df = df[['scaled_amount', 'scaled_time'] + [col for col
in df if col not in ['scaled_amount', 'scaled_time']]]
display(df.head())

class_dist = df['Class'].value_counts(normalize=True) *
100
print(f"Non-Fraud: {class_dist[0]:.2f}%")
print(f"Fraud: {class_dist[1]:.2f}%")

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

skf = StratifiedKFold(n_splits=5, shuffle=False)
for train_idx, test_idx in skf.split(features, target):
    X_train, X_test = features.iloc[train_idx],
features.iloc[test_idx]
    y_train, y_test = target.iloc[train_idx],
target.iloc[test_idx]

X_train, X_test = X_train.values, X_test.values
y_train, y_test = y_train.values, y_test.values

train_dist = np.unique(y_train,
return_counts=True)[1]/len(y_train)
test_dist = np.unique(y_test,
return_counts=True)[1]/len(y_test)
print('-'*100)
print("Label Distributions:\nTrain:", train_dist,
"\nTest:", test_dist)

balanced_df = pd.concat([
    df[df['Class'] == 1],
    df[df['Class'] == 0].sample(n=492, random_state=42)
]).sample(frac=1, random_state=42)
```

```
display(balanced_df.head())

plt.figure(figsize=(8,5))
sns.countplot(data=balanced_df, x='Class',
palette=["#0101DF", "#DF0101"])
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution in Balanced Dataset')
plt.show()

plt.figure(figsize=(12,10))
plt.figure(figsize=(12,10))
sns.heatmap(new_df.corr(), cmap='coolwarm_r',
annot=False,
            annot_kws={'size': 10}, fmt='.2f')
plt.title('Balanced Dataset Correlation Matrix')
plt.tight_layout()
plt.show()

features = new_df.drop('Class', axis=1)
target = new_df['Class']

dim_reduction = {
    'TSNE': TSNE(n_components=2, random_state=42),
    'PCA': PCA(n_components=2, random_state=42),
    'TruncatedSVD': TruncatedSVD(n_components=2,
algorithm='randomized', random_state=42)
}

reduced_features = {}
for name, model in dim_reduction.items():
    start = time.time()
    reduced_features[name] =
model.fit_transform(features.values)
    print(f"{name} completed in {time.time()-start:.2f}
seconds")

X_train, X_test, y_train, y_test = train_test_split(
```

```
    features.values,
    target.values,
    test_size=0.2,
    random_state=42,
    stratify=target
)

models = {
    "LogisticRegression": LogisticRegression(),
    "KNeighbors": KNeighborsClassifier(),
    "SVC": SVC(),
    "DecisionTree": DecisionTreeClassifier(),
    "RandomForest": RandomForestClassifier(),
    "XGBoost": XGBClassifier()
}

from sklearn.model_selection import cross_val_score

for model_name, model in classifiers.items():
    model.fit(X_train, y_train)
    cv_scores = cross_val_score(model, X_train, y_train,
cv=5)
    print(f"{model_name}: {cv_scores.mean():.2%} mean
accuracy")

param_grids = {
    "LogisticRegression": {
        "penalty": ['l1', 'l2'],
        'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]
    },
    "KNeighbors": {
        "n_neighbors": range(2, 5),
        'algorithm': ['auto', 'ball_tree', 'kd_tree',
'brute']
    },
    "SVC": {
        'C': [0.5, 0.7, 0.9, 1],
        'kernel': ['rbf', 'poly', 'sigmoid', 'linear']
    },
```

```
    "DecisionTree": {
        "criterion": ["gini", "entropy"],
        "max_depth": range(2, 4),
        "min_samples_leaf": range(5, 7)
    },
    "RandomForest": {
        "criterion": ["gini", "entropy"],
        "max_depth": range(2, 4),
        "min_samples_leaf": range(5, 7)
    },
    "XGBoost": {
        "learning_rate": [0.01, 0.05, 0.1],
        "max_depth": range(2, 4),
        "gamma": [0, 0.1],
        "reg_alpha": [0, 0.25, 0.5],
        "reg_lambda": [1, 1.25, 1.5],
        "scale_pos_weight": range(1, 100, 10),
        "eval_metric": ["aucpr"]
    }
}

best_estimators = {}
for name in models.keys():
    grid_search = GridSearchCV(
        estimator=models[name],
        param_grid=param_grids[name],
        cv=5,
        n_jobs=-1,
        verbose=1
    )
    grid_search.fit(X_train, y_train)
    best_estimators[name] = grid_search.best_estimator_
    print(f"Best {name} params:
{grid_search.best_params_}")

model_scores = {
    "Logistic Regression": cross_val_score(log_reg,
X_train, y_train, cv=5),
```

```
    "K-Nearest Neighbors":
cross_val_score(knears_neighbors, X_train, y_train,
cv=5),
    "Support Vector": cross_val_score(svc, X_train,
y_train, cv=5),
    "Decision Tree": cross_val_score(tree_clf, X_train,
y_train, cv=5),
    "Random Forest": cross_val_score(forest_clf, X_train,
y_train, cv=5),
    "XGBoost": cross_val_score(xgb_clf, X_train, y_train,
cv=5)
}

for model_name, scores in model_scores.items():
    print(f"{model_name} CV Score: {scores.mean():.2%}")

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

for train_idx, test_idx in sss.split(features, target):
    X_train, X_test = features.iloc[train_idx],
features.iloc[test_idx]
    y_train, y_test = target.iloc[train_idx],
target.iloc[test_idx]

X_train, X_test = X_train.values, X_test.values
y_train, y_test = y_train.values, y_test.values

metrics = {
    'accuracy': [],
    'precision': [],
    'recall': [],
    'f1': [],
    'auc': []
}

X_resampled, y_resampled =
NearMiss().fit_resample(features.values, target.values)
```

```
print(f"Resampled class distribution:
{Counter(y_resampled)}")

for train_idx, test_idx in sss.split(X_train, y_train):
    pipeline = imbalanced_make_pipeline(
        NearMiss(sampling_strategy='majority'),
        log_reg
    )

    model = pipeline.fit(X_train[train_idx],
y_train[train_idx])
    y_pred = model.predict(X_train[test_idx])


metrics['accuracy'].append(pipeline.score(original_Xtrain
[test_idx], original_ytrain[test_idx]))

metrics['precision'].append(precision_score(original_ytra
in[test_idx], y_pred))

metrics['recall'].append(recall_score(original_ytrain[tes
t_idx], y_pred))

metrics['f1'].append(f1_score(original_ytrain[test_idx],
y_pred))

metrics['auc'].append(roc_auc_score(original_ytrain[test_
idx], y_pred))

from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt
import numpy as np

def plot_learning_curves(estimators, X, y, ylim=None,
cv=None,
                         n_jobs=1,
train_sizes=np.linspace(.1, 1.0, 5)):
    fig, axes = plt.subplots(3, 2, figsize=(20, 24),
sharey=True)
```

```
    axes = axes.flatten()

    if ylim is not None:
        plt.ylim(*ylim)

    model_names = [
        "Logistic Regression",
        "k-Nearest Neighbors",
        "Support Vector Classifier",
        "Decision Tree",
        "Random Forest",
        "XGBoost"
    ]

    colors = {
        "train": "#ff9124",
        "test": "#2492ff"
    }

    for idx, (estimator, ax, name) in
enumerate(zip(estimators, axes, model_names)):
        train_sizes, train_scores, test_scores =
learning_curve(
            estimator, X, y, cv=cv, n_jobs=n_jobs,
train_sizes=train_sizes)

        train_mean = np.mean(train_scores, axis=1)
        train_std = np.std(train_scores, axis=1)
        test_mean = np.mean(test_scores, axis=1)
        test_std = np.std(test_scores, axis=1)

        ax.fill_between(train_sizes, train_mean -
train_std,
                        train_mean + train_std,
alpha=0.1, color=colors["train"])
        ax.plot(train_sizes, train_mean, 'o-',
color=colors["train"],
                label="Training score")
```

```
        ax.fill_between(train_sizes, test_mean -
test_std,
                        test_mean + test_std, alpha=0.1,
color=colors["test"])
        ax.plot(train_sizes, test_mean, 'o-',
color=colors["test"],
                label="Cross-validation score")

        ax.set_title(f"Learning Curve: {name}",
fontsize=16)
        ax.set_xlabel("Training set size", fontsize=14)
        ax.set_ylabel("Score", fontsize=14)
        ax.grid(True)
        ax.legend(loc="best")

    plt.tight_layout()
    return plt

def evaluate_models(models, X, y, cv=5):
    results = {}
    for name, model in models.items():
        if name in ['LogisticRegression', 'SVC']:
            pred = cross_val_predict(model, X, y, cv=cv,
method="decision_function")
        else:
            pred = cross_val_predict(model, X, y, cv=cv)

        fpr, tpr, _ = roc_curve(y, pred)
        auc_score = roc_auc_score(y, pred)
        results[name] = {'fpr': fpr, 'tpr': tpr, 'auc':
auc_score}
    return results

models = {
    'LogisticRegression': log_reg,
    'KNearsNeighbors': knears_neighbors,
    'SVC': svc,
    'DecisionTree': tree_clf,
    'RandomForest': forest_clf,
```

```
    'XGBoost': xgb_clf
}

roc_results = evaluate_models(models, X_train, y_train)

plt.figure(figsize=(12,6))
for name, data in roc_results.items():
    plt.plot(data['fpr'], data['tpr'], label=f'{name}
(AUC = {data["auc"]:.4f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([-0.01, 1])
plt.ylim([0, 1.01])
plt.xlabel('False Positive Rate', fontsize=12)
plt.ylabel('True Positive Rate', fontsize=12)
plt.title('ROC Curve Comparison', fontsize=14)
plt.legend(loc='lower right')
plt.show()

def plot_confusion_matrices(y_true, y_preds,
model_names):
    fig, axes = plt.subplots(3, 2, figsize=(18, 22))
    axes = axes.flatten()

    for ax, y_pred, name in zip(axes, y_preds,
model_names):
        cm = confusion_matrix(y_true, y_pred)
        sns.heatmap(cm, ax=ax, annot=True, fmt='d',
cmap='coolwarm_r',
                    annot_kws={'size': 16}, cbar=False)
        ax.set_title(name, fontsize=16)
        ax.set_xticklabels(['no fraud', 'fraud'],
fontsize=14)
        ax.set_yticklabels(['no fraud', 'fraud'],
fontsize=14)
        ax.set_xlabel('Predicted', fontsize=14)
        ax.set_ylabel('Actual', fontsize=14)

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

```
y_preds = [
    y_pred_log_reg,
    y_pred_knear,
    y_pred_svc,
    y_pred_tree,
    y_pred_forest,
    y_pred_xgb
]

model_names = ["LogReg", "kNN", "SVC", "DTC", "RFC",
"XGB"]
plot_confusion_matrices(y_test, y_preds, model_names)

for name, y_pred in zip(model_names, y_preds):
    print(f'\n{name} Classification Report:')
    print(classification_report(y_test, y_pred))
```