Оценка кредитных рисков с помощью методов машинного обучения

Программа: Цифровые Профессии
Специализация: Аналитик больших данных
Подготовил: Овчинников Павел Александрович

г. Москва
2024

1. Оглавление

1. Введение

1.1. Предмет исследования и общее описание работы

Данный дипломный проект посвящен анализу и оценке рисков банковского кредитования с применением методов машинного обучения. Основной целью исследования является прогнозирование с использованием моделей XGBoost, LightGBM и CatBoost вероятности невозврата кредита клиентом. Эти модели способны предсказывать вероятность дефолта, помогая банкам принимать обоснованные решения.

Банковский кредитный риск представляет собой финансовую угрозу, связанную с возможностью невыполнения заемщиком своих финансовых обязательств перед банком, что означает риск возможного дефолта заемщика. Кредитный риск возникает в случае возможного невыполнения клиентом своих договорных обязательств по возврату заемных средств, включая, например, ипотечные кредиты, кредитные карты и другие формы кредитования. Этот вид риска в первую очередь оценивается кредитором и включает в себя потери, возникшие в результате неоплаты основного долга и процентов, потерянные денежные потоки и издержки, связанные с процедурой взыскания просроченной задолженности. Ущерб, который может быть как полным, так и частичным, используется для оценки уровня кредитного риска и принятия решений об оценке кредитоспособности заемщика.

В наше время финансовая сфера становится все более динамичным и критически важным сектором экономики. Она охватывает широкий спектр областей, включая инвестиции, банковское дело (транзакционный банковские операции и вклады, кредитование организаций и частных лиц), страхование, депозитарий и микрокредитование, каждая из которых требует специализированных методов и инструментов для успешного функционирования. Машинное обучение является одним из таких методов, которое находит применение во многих областях, включая финансы.

Одной из ключевых особенностей финансовой отрасли является обилие данных, которые требуют обработки и анализа для принятия обоснованных решений. К примеру, данные о кредитной истории содержат множество сведений о заемщиках, необходимых для выдачи кредита. Ранее этим занимались специалисты, и такая работа требовала много времени и усилий. Машинное обучение позволяет автоматизировать процесс анализа данных и принятия решений, ускоряя работу и повышая ее точность.

В результате экспериментов с вышеуказанными алгоритмами машинного обучения было установлено, что все три модели успешно справляются с задачей оценки кредитного риска, обладая достаточно высокой производительностью, скоростью и точностью. Это делает возможным для использования их в реальном времени, например, при личном обращении в банк или заполнении онлайн-заявок.

Таким образом, данная дипломная работа показывает потенциал использования техник машинного обучения в оценке кредитных рисков.

1.2. Актуальность и практическая ценности темы

В течение длительного времени, более века, финансовая отрасль является одной из наиболее динамично развивающихся и локомотивом развития других секторов экономики. Она охватывает множество областей, таких как инвестиции, банковское дело, страхование, а также кредитование. Каждая из этих областей имеет свои особенности и требует специальных методов и инструментов для эффективной работы использования. В настоящее время, одним из таких методов является машинное обучение, которое нашло свое применение во многих сферах, включая финансовую.

Финансовая сфера характеризуется обилием данных, требующих обработки и анализа для эффективного принятия решений. Например, данные о кредитной истории предоставляют обширную информацию о заемщиках, которую необходимо изучить для выдачи ссуды. Ранее эту задачу выполняли люди, что требовало значительных временных и трудовых затрат. Позднее

вводились математические алгоритмы в виде программного кода для расчета кредитных рейтингов. С помощью современного машинного обучения проведение анализа данных и принятие решений по ним становится автоматизированным процессом, что увеличивает скорость работы и повышает точность принимаемых решений.

Актуальность этой темы обусловлена необходимостью постоянного повышения эффективности и адаптивности деятельности финансовой отрасли в условиях быстро меняющейся экономической обстановки и увеличения объема обрабатываемых данных. Прогнозирование кредитных рисков является критически важным заданием для банков и других финансовых институтов, выдающих кредиты, поскольку неправильное предсказание рисков может привести к серьезным финансовым убыткам.

Таким образом, исследование, проведенное в данной работе, представляет практическую ценность для финансовых учреждений, специализирующихся на предоставлении кредитов. Она позволит определить оптимальную модель машинного обучения для оценки кредитного риска на основе данных компании Nubank и улучшить процесс принятия решений в этой области.

2. Цель и задачи работы

2.1. Изучение и анализ предметной области.

Цель данного исследования состоит в изучении и применения методов обработки табличных данных с использованием машинного обучения для оценки кредитного риска банка на основе набора данных о клиентах финансовой компании Nubank. Известно, что Nubank это бразильский цифровой банк, один из крупнейших в Латинской Америке.

Набор данных acquisition_train.csv скачан с URL-адреса сети Интернет: *https://files.pythonhosted.org/packages/52/39/128fff65072c8327371e3c594f3c826d29c85b21cb6485980353b168e0e4/catboost-0.24.2-cp36-none-manylinux1_x86_64.whl (66.1MB)*

В наборе данных (после загрузки данных в память компьютера - датасет) хэширована некоторая личная информация клиентов, чтобы сохранить их анонимность. Была проведена подробная предобработка и нормализация данных для подготовки их к дальнейшему анализу и использованию в моделях машинного обучения.

Для достижения этой цели в дипломной работе будут выполнены следующие задачи:

- Исследовать и проанализировать существующие методы и модели оценки кредитного риска. Для данной задачи анализа выбраны три алгоритма повышения градиента, чтобы определить, какой из них дает лучшие результаты.
- Предобработать данные о заемщиках, включая финансовые показатели, историю кредитования и другие соответствующие параметры.
- Изучить методы машинного обучения для разработки моделей оценки кредитного риска.
- Сравнить производительность трех выбранных моделей, оценив их точность, скорость работы и практическую применимость.

Эти задачи будут выполняться с использованием программного

обеспечения Python и библиотек машинного обучения, таких как numpy, pandas, mathlib. Для обработки данных применены методы предварительной обработки, визуализации и анализа данных, а также алгоритмы машинного обучения.

2.2. Подходы к оценке кредитного риска с помощью методов машинного обучения.

Существует множество подходов к оценке кредитного риска, от классических методов, таких как анализ финансовых показателей и статистические методы, до более современных, использующих методы машинного обучения. Классические методы оценки кредитного риска основаны на анализе финансовых показателей заемщика, таких как его доходы, расходы, имущество, а также его кредитная история [3]. Однако, такой подход имеет свои ограничения, поскольку он не учитывает некоторые важные факторы, такие как личная история заемщика, которая может повлиять на его способность выплатить кредит.

Методы машинного обучения в области оценки кредитного риска предоставляют новые возможности для определения риска заемщика. Эти методы основаны на использовании алгоритмов машинного обучения, которые обучаются на исторических данных, чтобы определить связи между различными факторами и вероятностью возврата кредита [4]. Такие методы могут использовать не только финансовые данные, но и другую информацию, например, данные социальных сетей, которые могут дать дополнительную информацию о заемщике.

Преимущества методов машинного обучения в оценке кредитного риска заключаются в возможности использования большего количества факторов для определения риска заемщика, а также в возможности улучшения точности прогнозирования. Недостатком таких методов может быть их сложность и необходимость использования больших объемов данных для обучения алгоритмов [5].

Необходимо отметить, что применение методов машинного обучения в

этой области сопряжено с определенными рисками, связанными с возможностью ошибок при обучении модели и прогнозировании. Поэтому особое внимание в работе будет уделено анализу этих рисков и оценке эффективности использования методов машинного обучения для прогнозирования кредитных рисков в сравнении с традиционными подходами анализа данных. Несмотря на это, изучение теоретических основ оценки кредитного риска и применения методов машинного обучения является важной задачей для финансовых организаций.

Также в рамках данной работы будет проведено сравнение результатов, полученных на основе расчетов трех моделей машинного обучения. Это позволит сделать выводы о преимуществах и недостатках использования методов машинного обучения в этой области и определить возможности для дальнейших исследований.

3. Теоретические основы методов анализа кредитных рисков с помощью машинного обучения

3.1. Изучение и анализ предметной области.

Оценка кредитного риска представляет собой ключевой инструмент, который финансовые учреждения применяют при выдаче кредитов. В сфере банковского дела существует разветвленная иерархия рисков, связанных с деятельностью кредитных организаций [1]. На рисунке 1 показана примерная схема банковских рисков.

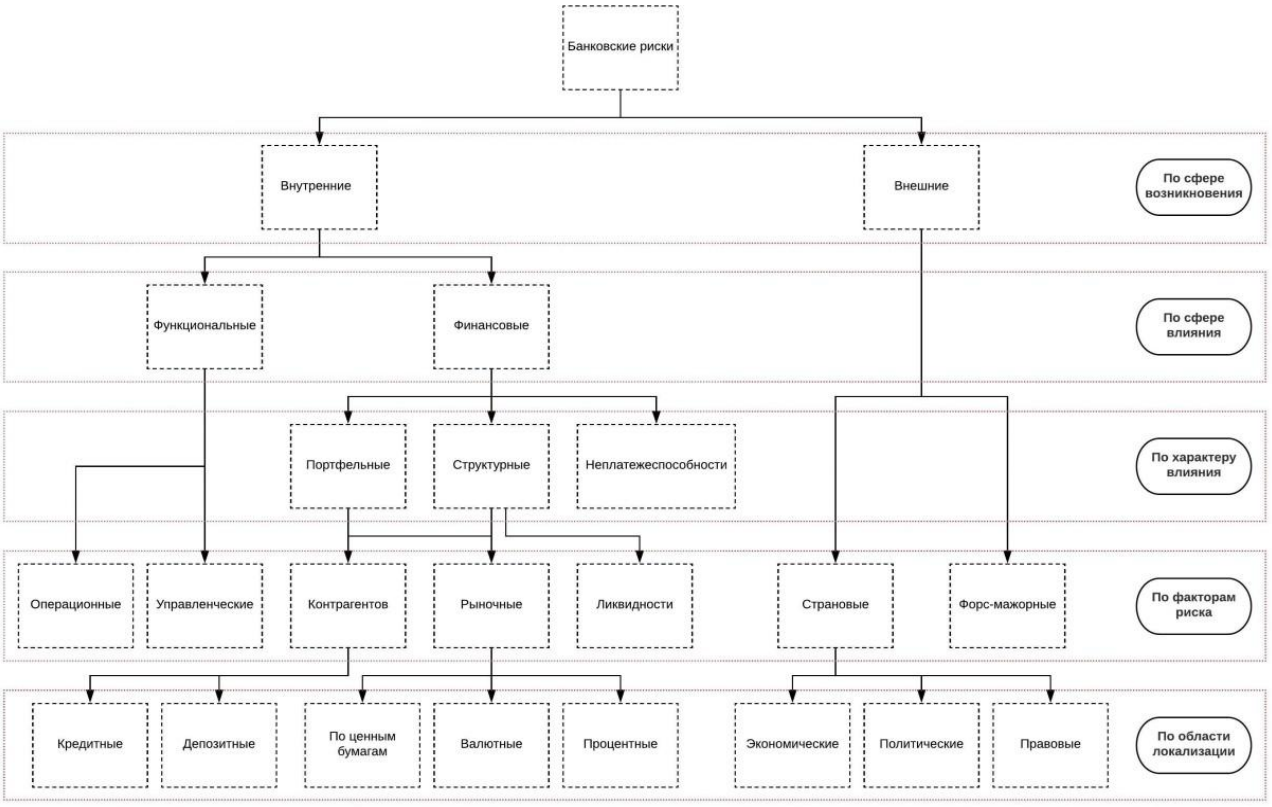


Рисунок 1 Иерархия банковских рисков

Схема на Рисунке 1 широко обсуждается в академической литературе и является важным объектом исследований. Основное внимание уделяется кредитным операциям, поскольку они составляют примерно 80% всех транзакций в коммерческих банках. Поэтому снижение кредитного риска является критической задачей для банковской системы.

Оценка и управление кредитным риском являются критически важными

задачами для банков, поскольку принятие неверного решения может привести к финансовым убыткам и отрицательным последствиям для всей банковской организации, включая другие подразделения банка, незадействованные в выдаче кредитов.

В мире разработаны много подходов к оценке кредитного риска: от традиционных методов, например анализ финансовых показателей либо статистические методы, до современных подходов, основанных на методах машинного обучения. Традиционные методы оценки опираются на анализ финансовых данных заемщика, таких как доходы, расходы, имущество и кредитная история [3]. Вместе с тем, классический подход имеет свои ограничения, так как он не учитывает такие важные факторы, такие как индивидуальная кредитная история, которая может влиять на его возможности по возврату кредита.

Использование методов машинного обучения в сфере оценки кредитного риска открывает новые перспективы для определения риска заемщика. Эти подходы основаны на применении алгоритмов машинного обучения, которые анализируют исторические данные для выявления связей между различными факторами и вероятностью погашения кредита [4]. Они могут использовать не только финансовые данные, но и другую информацию, включая данные из социальных сетей, что дает дополнительное понимание о заёмщике.

Достоинство и превосходство методов машинного обучения в оценке кредитного риска заключаются в возможности учета большего количества факторов для определения риска заемщика и повышения точности прогнозирования. Однако недостатком таких методов является их сложность и требование к использованию большого объема данных для обучения алгоритмов.

Поэтому изучение теоретических основ оценки кредитного риска и применение методов машинного обучения в этой области являются ключевыми задачами для банков.

3.2.Методология.

Изучены три широко известных алгоритма прогнозирования кредитоспособности по известным характеристикам заемщика на основе методов машинного обучения (кластеризации, регрессионного анализа, классификации). XGBoost от компании-разработчика The XGBoost Contributors, LightGBM, первоначально разработанную Microsoft и CatBoost от Яндекс с лицензией Apache 2.0. Данные алгоритмы позволяет использовать как отдельные модели, так и все возможные их комбинации. В рассматриваемом подходе также предлагается провести предварительный анализ данных (дискретизация, поиск статистически значимых характеристик заемщика) и использовать различные критерии качества для выбора оптимальной структуры. На основе полученных результатов клиенты банка по уровню кредитоспособности делятся на заданное число классов k.

Задачу прогнозирования кредитоспособности с точки зрения статистической теории принятия решений можно рассматривать как задачу классификации новых клиентов на основе информации о прошлых клиентах [1]. В области кредитования эта задача решается в рамках анкетного скоринга [2–5].

Формальная постановка данной задачи заключается в следующем. [http://www.fin-izdat.ru/journal/fc/]

Пусть имеется множество клиентов банка { Zi }, i = 1,..., n, каждый из которых характеризуется p-мерным вектором признаков Xi = $(x_{i1}, ..., x_{ip})^T$. Известна также принадлежность каждого клиента Yi к одному из двух классов кредитоспособности:

$$Y = \begin{cases} y = 1 - клиент\ кредитоспособен; \\ y = 0 - клиент\ некредитоспособен. \end{cases}$$

Соответствующая выборка является обучающей: на ее основе необходимо описать процедуры, с помощью которых можно было бы с наибольшей точностью отнести новых клиентов к одному из классов $k \geq 2$, имея в качестве входной информации только наборы признаков Xi = $(x_{i1}, ..., x_{ip})^T$, описывающих

новых клиентов. Поскольку клиенты могут характеризоваться как количественными, так и качественными признаками, возникает задача классификации клиентов в пространстве разнотипных признаков.

Набор признаков x1 ,..., xp, как правило, представляет собой данные из анкет, которые заполняются при подаче заявки на кредит. Поскольку задан лишь примерный перечень информации для анализа финансового положения заемщика, существуют множество различных форм анкет. Основной набор признаков: ФИО, дата рождения, паспортные данные, образование, адрес, семейное положение, ФИО и дата рождения родственников, сведения об основной работе, ежемесячные доходы и расходы, информация об имеющемся имуществе, информация об имеющихся долгах и обязательствах, информация об инвалидности, запрашиваемая сумма и т.д.

Помимо данных из анкеты, если клиент уже брал кредит в прошлом или имеет кредит в настоящее время, в качестве признаков могут также выступать данные о кредитной истории, полученные из кредитного бюро посредством запроса. Возможно несколько вариантов представления информации о кредитной истории. В первом случае это может быть обобщенное решение: кредитная история — положительная либо отрицательная. Отрицательной историей обычно считается история при задержке платежей сроком на три и более месяца. В другом случае в качестве признаков могут использоваться данные, полученные из кредитного бюро (см: https://www.tcsbank.ru/tournament).

Следует отметить, что один и тот же набор признаков может быть дан как кредитоспособным, так и некредитоспособным клиентом, поэтому принципиально невозможно достигнуть абсолютно точной классификации.

Дополнительно, как будет далее показано в работе клиентов из зоны риска неплатежеспособности можно отнести к классификации с низким, средним и высоким рискам. Иначе, например, в случае двух классов при распределениях 90%/10% и 55%/45% клиент будет кредитоспособен в обоих случаях, однако с очевидной разницей.

Помимо вышеуказанной модели существуют другие алгоритмы, например, Деревья принятия решений [11], где клиентов последовательно разделяют на классы по одной из переменных так, чтобы эти классы максимально возможно отличались по величине кредитного риска. При этом на первом шаге разделение производится по самому значимому фактору. Далее процесс продолжается до того момента, пока оставшиеся классы не становятся настолько малы, что следующее разбиение не приведет к статистически значимому различию на уровне риска. Количество классов на каждом шаге процедуры построения дерева решений выбирается автоматически. К преимуществам деревьев решений относятся быстрая обработка больших объемов данных, легкая интерпретируемость результатов, работа с пропущенными, числовыми и нечисловыми типами данных, а также отсутствие ограничений на коррелируемость между зависимыми переменными. К недостаткам метода можно отнести неоднозначность алгоритма построения структуры дерева, а также вопрос о том, когда стоит прекратить дальнейшее разделение на классы.

Таким образом, каждый из перечисленных методов имеет свои преимущества и недостатки: нет универсальной модели, с помощью которой можно было бы оценить кредитоспособность того или иного клиента с явным преимуществом. Представляет интерес использование особенностей всех перечисленных выше моделей по отдельности и во всевозможных комбинациях, поскольку комбинированный подход дает возможность компенсировать недостатки одних моделей при помощи других, следовательно, он направлен на повышение точности прогнозирования.

3.3.Подготовка исходных данных

Прежде чем приступить к построению моделей, необходимо подготовить исходные данные к анализу, поскольку в реальных статистических данных приходится сталкиваться с рядом проблем [15, 16]. Это необходимо для проверки полноты и целостности данных, а также для подготовки их к

использованию в моделях. Признаки могут иметь как числовую, так и нечисловую природу.

Важно провести масштабирование и преобразование данных в целях достижения более точных результатов моделей. Если у различных переменных значительно отличаются диапазоны значений, их необходимо привести к одному масштабу [7].

Предварительная подготовка, преобразование и последующий анализ первичных данных играют важную роль в разработке точных и надежных моделей машинного обучения. Этот этап позволяет проверить качество данных и подготовить их к использованию в моделях для оценки кредитных рисков.

Применяя метод машинного обучения для оценки кредитных рисков, неоходимо определить набор факторов (атрибутов) и целевую функция, которые могут влиять на вероятность невыполнения долговых обязательств и сроков их погашения. Эти атрибуты могут включать доход заемщика, его кредитную историю, сумму кредита и другие соответствующие переменные. Затем обучение модели на исторических данных, где известен исход каждого кредита (долг выплачен или нет, был ли допущен выход на просрочку). В процессе обучения логистическая регрессия оценивает веса каждого фактора и формирует линейную комбинацию, которая предсказывает вероятность невыполнения кредитных обязательств [10]. После обучения модель может быть использована для предсказания вероятности невыполнения кредита для новых заемщиков.

Сначала необходимо провести заполнение пропущенных данных, чтобы избежать ошибок при анализе. Затем важно обработать выбросы и других негативных факторов, которые могут повлиять на результаты моделей. Это можно сделать с применением различных методов, таких как замена выбросов на среднее или медиану, а также замена значений модой для текстовых данных.

Для удобства анализа и построения моделей классификации нечисловые признаки обычно кодируются определенным образом. Различают также дискретные признаки (пол клиента, образование, семейное положение и т.д.) и

непрерывные признаки (возраст, стаж работы, ежемесячные доходы и расходы и т.д.). В том случае, если модель требует непрерывных данных, необходимо заменить дискретные переменные большим числом переменных, которые будут принимать значение 0 либо 1. То есть вместо набора дискретных признаков $\{ X_i \}$, $i = 1,..., n$, каждый из которых принимает от 1 до r_i значений, получим новый набор непрерывных признаков

$\{ X_i^j \}$, $i = 1,...,n$, $j = 1,..., r_i$

Для таких моделей, как деревья классификации и байесовский классификатор, имеет смысл использовать дискретные переменные. В этом случае область значений каждой непрерывной переменной разбивается на отрезки, номера которых представляют значения новой дискретной переменной: этот процесс называется дискретизацией. На практике часто исходные данные содержат пропущенные значения, которые могут быть как случайными, так и неслучайными. В первом случае информация может быть просто не введена в базу данных, либо данные получены из разных источников, например собраны из различных кредитных бюро, а также в результате появления новых вопросов в анкете. Неполная априорная информация, во-первых, усложняет процесс построения и применения моделей классификации, а во-вторых, может повлиять на ухудшение качества оценки параметров. Если отсутствующих данных относительно немного, то можно просто удалить данные случаи. Напротив, при больших объемах отсутствующих значений, чтобы не потерять важную информацию, прибегают к методам восстановления [13].

4. Расчет риска дефолта клиента с помощью моделей XGBoost, LightGBM и CatBoost

4.1. Модели машинного обучения

В условиях современного мира машинное обучение и искусственный интеллект играют важную роль в анализе кредитных рисков, поскольку используется для создания моделей, способных категоризировать клиентов на основе имеющихся данных. В данном разделе диплома приводятся основы машинного обучения и три широко распространенных алгоритмов XGBoost, LightGBM и CatBoost, которые будут применяться в практической части диплома для анализа данных и оценки кредитных рисков.

Машинное обучение — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач. Машинное обучение исследует алгоритмы и модели, способные автоматически обучаться по данным и выполнять прогнозы или принимать решения без прямого программирования.

Машинное Обучение по прецедентам, или индуктивное обучение, основано на выявлении эмпирических закономерностей в данных. имеет следующие ключевые понятия:

- Обучающая выборка: набор данных используемых для обучения модели. В ней содержатся примеры, каждый из которых состоит из набора признаков (атрибутов) и соответствующую им целевую переменную.
- Признаки (атрибуты): характеристики или свойства, описывающие объекты в данных. В наборе данных Nubank признаки используются для описания заемщиков и их финансовых параметров.
- Модель: математическое представление, созданное на основе обучающей выборки, которое может делать прогнозы или классифицировать новые данные. Модель является результатом процесса обучения.
- Обучение: процесс настройки модели на основе обучающей выборки. При

обучении модель "обучается" выявлять закономерности и связи между признаками и целевой переменной.

- Тестовая выборка: независимый набор данных, используемый для оценки производительности модели на новых данных. Тестовая выборка помогает оценить способность модели к обобщению.

Алгоритмы классификации используются для прогнозирования принадлежности объектов к определенным классам. В контексте оценки кредитных рисков, классификационные алгоритмы могут помочь определить, относится ли заемщик к низкому, среднему или высокому риску.

4.2.Описание трех алгоритмов машинного обучения

В целях расчет риска дефолта клиента и чтобы определить, какая из моделей дает лучшие результаты экспериментируем со следующими алгоритмами:

- XGBoost (eXtreme Gradient Boosting) — это оптимизированная распределенная библиотека с открытым исходным кодом, используемая в машинном обучении и предоставляющая функциональность для решения задач, связанных с регуляризацией градиентного бустинга, т.е. для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. Разработчик The XGBoost Contributors.
- LightGBM, сокращение от «машина для повышения градиента света» — представляет собой бесплатную распределенную среду повышения градиента с открытым исходным кодом для машинного обучения, первоначально разработанную Microsoft. Он основан на алгоритмах дерева решений и используется для ранжирования, классификации и других задач машинного обучения. Основное внимание при разработке уделяется производительности и масштабируемости.
- CatBoost (Categorical Boosting) - это библиотека для машинного обучения

с открытым исходным кодом, разработанная Яндексом. Библиотека создавалась инженерами и специалистами Яндекса в качестве преемника Матрикснета — алгоритма, применяемого для ранжирования и прогнозирования, а также лежащего в основе рекомендательных технологий. CatBoost использует более универсальный алгоритм, поэтому она подходит для решения и других задач.

Предварительно, в соответствии с методологией нужно разделить данные на обучающие и тестовые наборы. После этого, в связи с несбалансированным набором данных, стандартизируем и и заново дискретизируем обучающий набор с функциями StandardScaler и RandomUnderSampler соответственно.

4.3. Критерии оценочных показателей моделей

Оценка эффективности моделей является одним из ключевых этапов при работе с данными и моделями машинного обучения. Для этого необходимо выбрать метрики, которые наилучшим образом отражают цель задачи. Точность (Precision) – это мера того, как много объектов было классифицировано правильно. Она определяется как отношение числа правильно классифицированных объектов ко всем объектам.

Для оценки и сравнения моделей мы должны рассматривать три оценочных показателя каждой их модели Precision (Точность), Recall (Отзывчивость, Отзыв) и F1 Score (Оценка F1) в качестве общих показателей оценки:

- **Точность (Precision)** - доля действительно правильных положительных идентификаций. Определяется как:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

- **Отзыв (Recall)** - доля реальных положительных результатов, которые были правильно идентифицированы. Определяется как:

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

- **Оценка F1 (F1 Score)** - показатель баланса между Точностью и Отзывчивостью. Формула определяется как:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Поскольку мы стремимся к снижению убытков компании путем прогнозирования риска дефолта клиента, важно, чтобы модель показала хорошую скорость отзыва, чтобы точно определить максимальное количество клиентов, которые могут прекратить вовремя выплачивать свои долги, минимизируя при этом количество ложных отказов.

Поэтому можно говорить, что Точность и Отзыв в известной степени антогонисты. Однако всегда есть компромисс между Точностью и Отзывом. В этой статье мы решили Поэтому полагаем необходимым сделать больший упор на Оценку F1, используя это в качестве показателя оценки.

4.4. Калибровка их параметров моделей обучения

Один из ключевых этапов работы с моделями машинного обучения — это калибровка их параметров для достижения наилучшей производительности. Чтобы добиться лучших результатов, вместо простого разделения данных на набор для обучения и тестирования, используем перекрестную проверку методом cross_validate, что позволяет лучше использовать данные и разбивает наши обучающие данные на k складок. Оставив значение k по умолчанию выполним пятикратную перекрестную проверку.

	Recall
XGBClassifier	0.662726
LGBMClassifier	0.647915
CatBoostClassifier	0.645911

В результате расчета видим, что все три модели показали одинаковые результаты с параметрами моделей по умолчанию.

Следующим шагом проведем настройку некоторых гиперпараметров моделей, чтобы узнать, можно ли добиться более высоких показателей точности. Метод, который мы применяем здесь — GridSearchCV, будет искать оптимальные значения параметров для каждого оцениваемого алгоритма.

Для всех трех применяемых моделей согласно официальной документации настроим следующие гиперпараметры:

4.4.1. XGBoost

- n_estimators - количество деревьев в модели
- max_depth - максимальная глубина дерева
- min_child_weight - минимальная сумма веса экземпляра, необходимая для дочернего элемента
- gamma - минимальное снижение потерь, необходимое для создания следующего раздела на листовом узле дерева
- learning_rate - в обновлении используется уменьшение размера шага для предотвращения переобучения

```
param_grid = {'n_estimators': range(0,1000,50),
 'max_depth': [1, 3, 5],
 'min_child_weight': [1, 3, 6],
 'gamma': [0, 1, 5],
 'learning_rate': [0.0001, 0.001, 0.01, 0.1]}
```

Наилучший коэффициент запоминания: 0,81 для {'n_estimators': 50, ' max_depth': 3, 'min_child_weight': 6, 'gamma': 1, 'Learning_rate': 0,0001 }

4.4.2. LightGBM

- max_depth - максимальная глубина дерева
- learning_rate - коэффициент усадки

- num_leaves - максимальное количество листьев в одном дереве
- min_data_in_leaf - минимальное количество данных на одном листе

```
param_grid = {'max_depth': np.arange(5, 75, 10),
 'learning_rate' : [0.001, 0.01, 0.1],
 'num_leaves': np.arange(20, 220, 50),
 'min_data_in_leaf': np.arange(100, 1000, 100)}
```

Наилучший коэффициент запоминания: 0,69 для { ‘learning_rate’: 0,01, ‘max_depth’: 5, ‘num_leaves’: 70, ‘min_data_in_leaf’: 400 }

4.4.3. CatBoost

- depth - глубина дерева
- learning_rate - как мы уже знаем, скорость обучения
- l2_leaf_reg - коэффициент при l2 регуляризационном члене функции стоимости

```
param_grid = {'depth': [6, 8, 10],
 'learning_rate': [0.03, 0.1],
 'l2_leaf_reg': [1, 5, 10]}
```

Лучший коэффициент запоминания: 0,65 для { «глубины»: 6, «l2_leaf_reg»: 5, «Learning_rate»: 0,03 }

4.4.4. Итоги настройки гиперпараметров моделей:

По итогам настройки гиперпараметров каждая из трёх моделей показала свои лучшие результаты:

- XGBoost продемонстрировал значительное увеличение результатов
- LightGBM и CatBoost показали незначительное улучшение.

4.5.Интерпретация результатов.

Оценив эффективность каждой и рассматриваемой модели машинного обучения на основе данных компании Nubank необходимо тщательно проанализировать и проинтерпретировать полученные результаты в рамках

оценки кредитного риска. Это поможет понять, какие факторы влияют на оценку кредитного риска и какие параметры моделей могут быть улучшены. Для интерпретации результатов будут использованы методы анализа важности признаков [18, С.176]. Этот шаг позволяет выделить ключевые аспекты, влияющие на оценку кредитного риска, и проанализировать их важность. Получив значения коэффициентов для всех признаков в модели, их можно ранжировать по возрастанию. Таким образом, получается некая относительная оценка важности признаков при использовании модели. Несмотря на свою простоту, имеет место несколько нюансов, которые могут повлиять на результат интерпретации.

Также будут проанализированы примеры ошибочной классификации, чтобы определить, какие типы ошибок наиболее часто возникают и как их можно уменьшить. Результаты интерпретации могут быть использованы для улучшения моделей машинного обучения. Например, если наиболее важным признаком является доход заемщика, то увеличение объема данных о доходах заемщиков может улучшить модель. Кроме того можно попробовать использовать другие методы машинного обучения или изменить параметры текущих моделей, чтобы улучшить их эффективность.

Таким образом, интерпретация результатов является важным этапом оценки кредитного риска на основе методов машинного обучения. Она позволяет определить факторы, влияющие на оценку кредитного риска, и выявить возможности для улучшения моделей машинного обучения. Помимо этого, интерпретация результатов также важна с точки зрения принятия решений и практической применимости. Результаты моделей машинного обучения могут быть использованы финансовыми учреждениями для принятия решений о выдаче кредитов. Понимание, какие факторы влияют на оценку кредитного риска, позволяет принимать обоснованные и информированные решения.

Например, если модель показывает, что наличие надежного залога

является одним из наиболее важных факторов для оценки кредитного риска, финансовые учреждения могут принимать во внимание наличие или отсутствие залога при принятии решения о выдаче кредита. Также результаты интерпретации могут помочь в разработке стратегий управления рисками и принятии решений о ценообразовании.

Помимо этого, интерпретация результатов может помочь в обеспечении соблюдения законодательства и этических норм. Если модель показывает, что определенный фактор (например, пол или раса) оказывает существенное влияние на оценку кредитного риска, это может указывать на наличие потенциальной дискриминации. В таком случае, необходимо провести дополнительный анализ и принять меры для устранения возможных негативных влияний и обеспечения справедливости и равноправия при выдаче кредитов.

5. Практическая часть

5.1.Анализ данных и Предварительная обработка

Ддя исследований в рамках дипломного проекта выбран и скачан набор данных acquisition_train.csv с URL-адреса сети Интернет:
https://files.pythonhosted.org/packages/52/39/128fff65072c8327371e3c594f3c826d29c85b21cb6485980353b168e0e4/catboost-0.24.2-cp36-none-manylinux1_x86_64.whl (66.1MB)

После загрузки набора данных получаем DataSet (датасет) — представление данных в памяти, которое обеспечивает согласованную модель реляционного программирования независимо от источника данных — содержащий 43 атрибута для 45 000 клиентов.

Атрибут target_default - это функция True / False и целевая переменная, которую мы пытаемся предсказать.

Подготовка и анализ данных является важным этапом в построении моделей машинного обучения для оценки кредитных рисков. Этот этап включает в себя работу с данными, которые будут использоваться в моделях. Он помогает убедиться в качестве и целостности данных и подготовить их к использованию в моделях. В целом, подготовка и анализ данных является ключевым этапом в создании точных и надежных моделей машинного обучения для оценки кредитных рисков. Это позволяет убедиться в качестве данных и подготовить их для использования в моделях.

Предобработка данных является одним из наиболее важных шагов в анализе данных, так как именно на этом этапе можно сильно повлиять на результаты анализа.

Предобработка данных включает в себя ряд различных процессов, таких как заполнение пропущенных значений, удаление дубликатов, обработка выбросов и масштабирование признаков. Рассмотрим некоторые наиболее распространенные методы предобработки данных и преобразования признаков.

Заполнение пропущенных значений. Одна из наиболее распространенных проблем – это пропущенные значения. Пропущенные значения могут возникать

по разным причинам, таким как неправильное считывание данных, либо некоторые значения могут быть неизвестны. Пропущенные значения могут исказить результаты анализа данных, поэтому необходимо заполнять их.

Удаление дубликатов. Дубликаты могут возникать по разным причинам, таким как ошибки ввода или дублирование данных. Удаление дубликатов является важным шагом в предобработке данных, так как они могут привести к искажению результатов анализа данных.

Обработка выбросов. Выбросы — это значения, которые находятся за пределами ожидаемого диапазона значений. Выбросы могут возникать по разным причинам, таким как ошибки ввода данных или ошибки измерений. Выбросы могут значительно исказить результаты анализа данных, поэтому их необходимо обрабатывать.

Масштабирование признаков.— это процесс изменения масштаба значений признаков. Масштабирование признаков может улучшить производитель алгоритмов машинного обучения, т.к. многие алгоритмы чувствительны к различным масштабам значений признаков. Существуют различные методы масштабирования признаков, такие как стандартизация и нормализация.

Преобразование категориальных признаков. Категориальные признаки — это признаки, которые могут принимать ограниченное количество значений. Примерами категориальных признаков могут служить пол, город проживания, тип автомобиля и т.д. Для анализа данных необходимо преобразовать категориальные признаки в числовые, так как большинство алгоритмов машинного обучения работают только с числовыми значениями.

Удаление лишних признаков. В данных могут быть признаки, которые не имеют важности для анализа или которые могут исказить результаты. Удаление этих признаков может упростить анализ и улучшить производительность модели.

Работа с несбалансированными данными. В некоторых задачах машинного обучения данные могут быть несбалансированными, то есть классы могут иметь

различные размеры. Несбалансированные данные могут привести к смещению модели в сторону более крупных классов, что может привести к неверным результатам. Для работы с несбалансированными данными можно использовать методы, такие как изменение порогового значения классификации, использование взвешивания классов или использование методов сэмплирования, таких как андерсэмплинг или оверсэмплинг.

Визуализация данных. Визуализация данных является важным инструментом в анализе данных. Она позволяет наглядно представить данные и выявить закономерности и зависимости между признаками. Для визуализации данных можно использовать различные библиотеки Python, такие как Matplotlib и Seaborn.

Преобразование признаков. Преобразование признаков может улучшить производительность модели, сделать данные более информативными и уменьшить размерность данных. Преобразование признаков может включать в себя такие методы, как нормализация, стандартизация, преобразование категориальных признаков в числовые и т.д.

Выбор признаков. Выбор признаков может улучшить производительность модели, сделать данные более информативными и уменьшить размерность данных. Некоторые признаки могут быть неинформативными или коррелировать с другими признаками, поэтому необходимо выбирать только значимые признаки. Для выбора признаков можно использовать методы, такие как анализ корреляции, методы, основанные на статистике, рекурсивное удаление признаков и т.д.

Создание новых признаков. Создание новых признаков может улучшить производительность модели и сделать данные более информативными. Новые признаки могут быть созданы на основе существующих признаков, используя методы, такие как агрегирование, бинаризация, генерация признаков на основе текстовых данных и т.д. Оценка качества данных. Оценка качества данных является важным шагом в предобработке данных и преобразовании признаков. Необходимо убедиться в том, что данные достаточно качественные и

соответствуют требованиям задачи. Для оценки качества данных можно использовать методы, такие как анализ выбросов, анализ корреляции, анализ распределения данных и т.д.

В заключение, предобработка данных и преобразование признаков являются неотъемлемыми шагами в анализе данных и машинном обучении. Необходимо учитывать особенности данных и выбирать соответствующие методы для их обработки. Различные библиотеки Python позволяют автоматизировать процесс предобработки данных, что позволяет ускорить работу и повысить эффективность анализа. Однако необходимо помнить, что выбор методов предобработки данных и преобразования признаков может существенно влиять на результаты модели, поэтому следует проявлять осторожность и тщательно проверять результаты.

Изучение набора данных показало, что часть функции имеют пропущенные значения и выбросы. Дополнительно, в целях экономии системных вычислительных ресурсов, удалили переменные, которые не добавляют ценности модели.

Для начала необходимо произвести заполнение пропущенных значений, что поможет избежать ошибок при анализе данных. Далее следует обработка выбросов и шумов, которые могут исказить результаты моделей. Это может быть осуществлено с помощью различных методов, таких как замена выбросов на среднее значение или медиану, замена значений модой для текстовых типов данных.

Полученное распределение числовых характеристик атрибутов датасет визуализируем с помощью гистограмм на Рисунке 2.

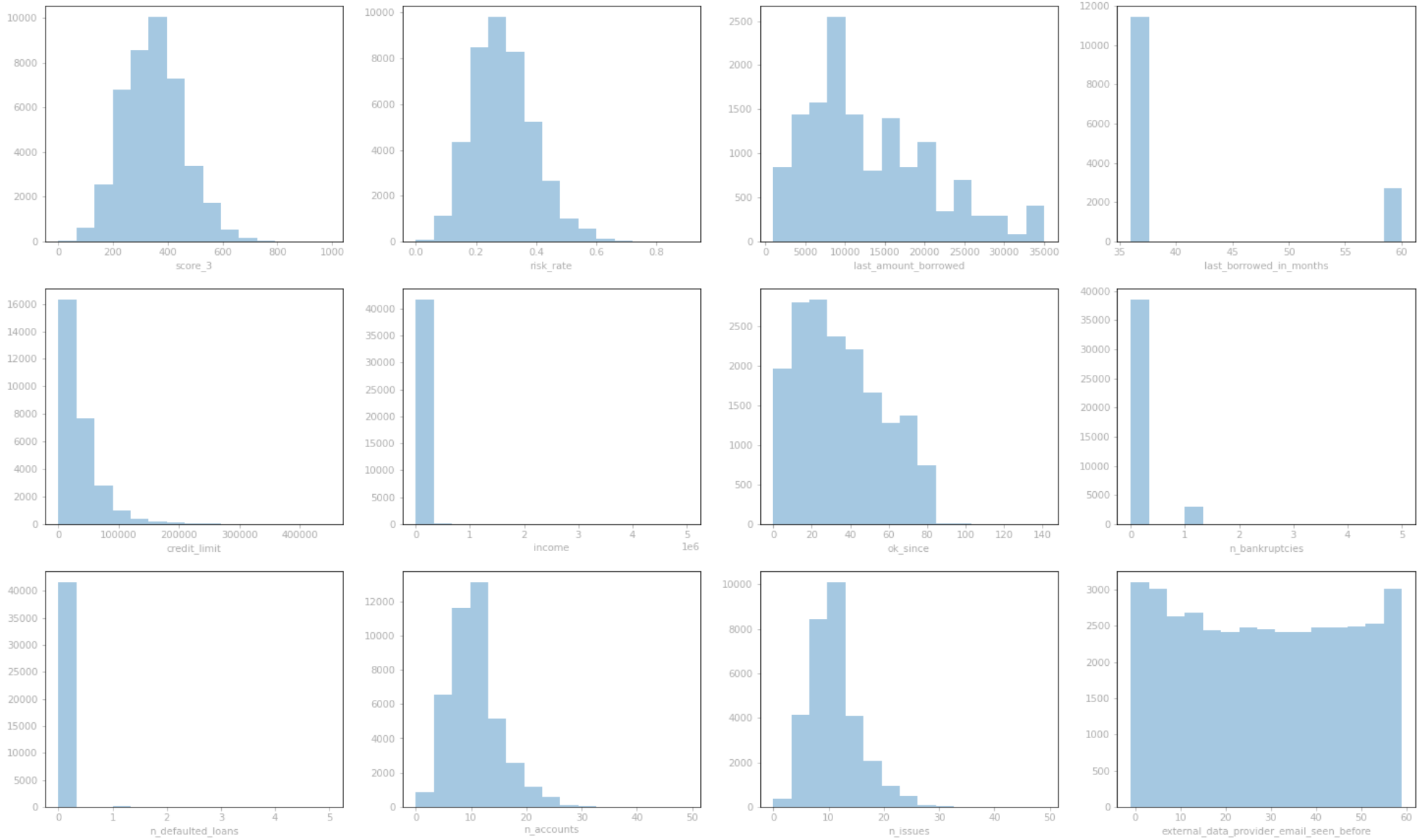


Рисунок 2 Визуальный анализ распределения данных

Сначала необходимо провести заполнение пропущенных данных, чтобы избежать ошибок при анализе данных и обучении моделей. Все перечисленные выше артефакты имеют пропущенные значения, которые необходимо обработать. Как мы можем видеть, они имеют искаженное распределение, что указывает на то, что мы должны заполнить пропущенные значения медианным значением для каждого объекта.

Недостающие значения в оставшихся столбцах заполним в соответствии с особенностями каждой функции:

- Для категориальных переменных используем самое часто встречающееся значение для заполнения.
- Числовые переменные будут заполнены средними значениями.
- В определенных случаях мы заполним пропуски значениями нуля, так как разумно предположить, что не у всех клиентов будут данные по этим переменным.

Затем важно обработать выбросы и шумы, которые могут повлиять на результаты моделей. Это можно сделать с применением различных методов, таких как замена выбросов на среднее или медиану, а также замена значений модой для текстовых данных.

Некоторые из наиболее важных признаков включают образование и тип занятости заемщика, так как это может указывать на потенциальный доход заемщика и его способность выплачивать кредиты. Информация о задолженностях по текущим и прошлым кредитам также является важным показателем, так как это может указывать на риски невозврата кредита. Данные о наличии недвижимости и транспортных средств также могут быть полезны при оценке кредитного риска, так как это может служить дополнительным обеспечением для займа.

Как видно на Рисунке 3, часть признаков в датасете имеет недостающие значения, которые предстоит заменить в рамках подготовки данных к дальнейшей работе.

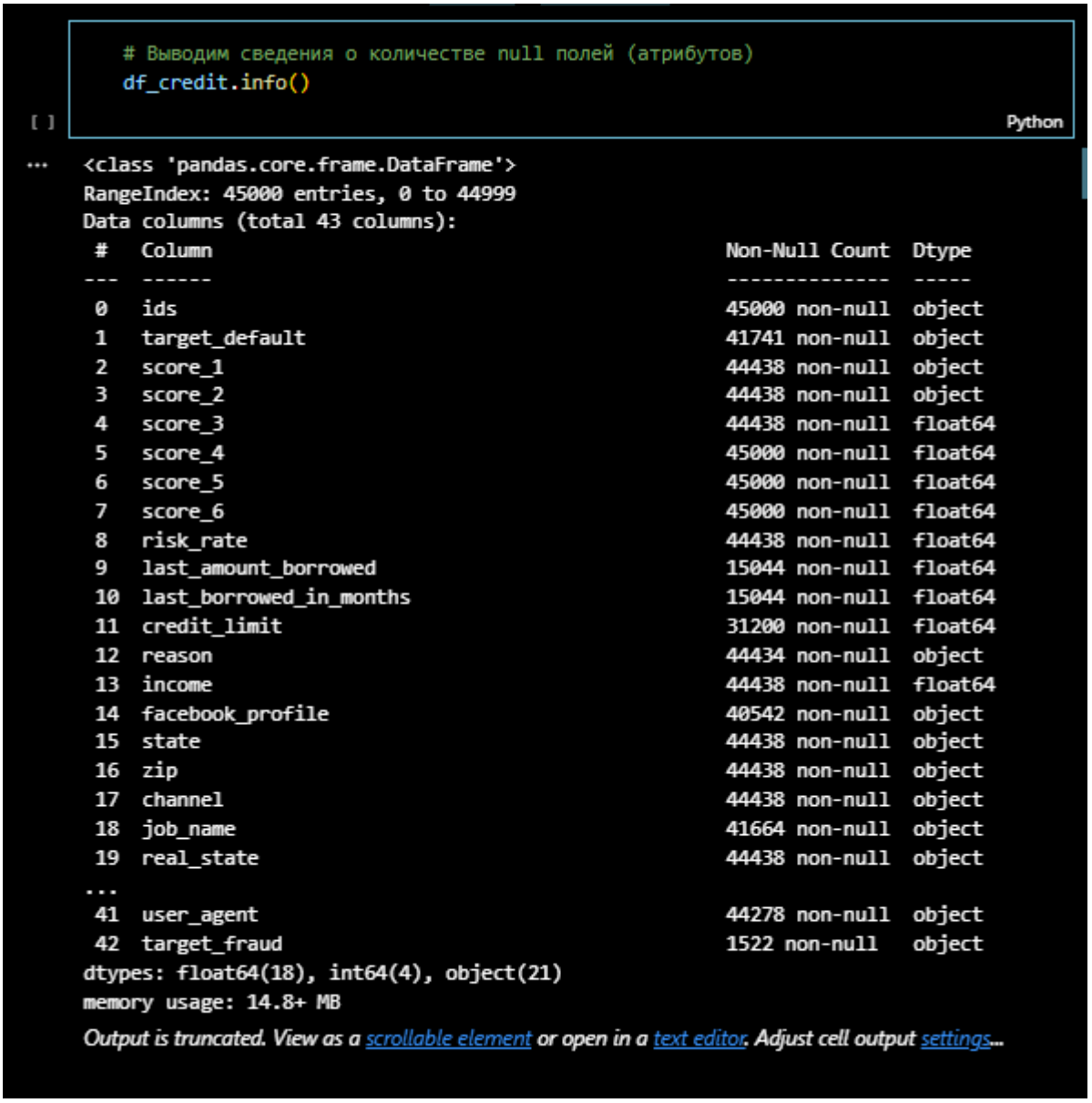
```
# процент пропущенных значений для каждого поля (атрибута)
print((df_credit.isnull().sum() * 100 / df_credit.shape[0]).sort_values(ascending
```

Python

```
target_fraud                                  96.617778
last_amount_borrowed                          66.568889
last_borrowed_in_months                       66.568889
ok_since                                      58.988889
external_data_provider_credit_checks_last_2_year  50.284444
external_data_provider_credit_checks_last_year    33.608889
credit_limit                                  30.666667
n_issues                                      25.653333
facebook_profile                               9.906667
marketing_channel                              7.951111
job_name                                       7.413333
target_default                                 7.242222
external_data_provider_email_seen_before       4.962222
lat_lon                                        3.028889
user_agent                                     1.604444
n_bankruptcies                                 1.548889
n_defaulted_loans                              1.275556
reason                                         1.257778
income                                         1.248889
real_state                                     1.248889
state                                          1.248889
zip                                            1.248889
channel                                        1.248889
score_3                                        1.248889
score_2                                        1.248889
...
external_data_provider_credit_checks_last_month    0.000000
email                                              0.000000
ids                                                0.000000
dtype: float64
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Рисунок 3 Доля пропущенных значений для каждого атрибута



Одним из ключевых аспектов, определяющих возможность одобрения заявки на кредит, является финансовая устойчивость заемщика. В этом контексте важными категориями переменных являются доходы и затраты заемщика, а также общая задолженность перед другими кредиторами.

```python
# Удалим колонки "channel" и "external_data_provider_credit_checks_last_2_year"
df_credit.drop(labels=['channel', 'external_data_provider_credit_checks_last_2_year'], axis=1, inplace=True)
```

Python

```python
# удалим некоторые столбцы, которые не добавляют ценности модели
df_credit.drop(labels=['email', 'reason', 'zip', 'job_name', 'external_data_provider_first_name', 'lat_lon',
                       'shipping_zip_code', 'user_agent', 'profile_tags', 'marketing_channel',
                       'profile_phone_number', 'application_time_applied', 'ids'], axis=1, inplace=True)
```

Python

```python
# проверим, есть ли в наборе данных выбросы. Покажем статистику
df_credit.describe()
```

Python

	score_3	score_4	score_5	score_6	risk_rate	last_amount_borrowed	last_borrowed_in_months	credit_limit	income	ok_since	n_bankruptcies	n_defaulte
count	41741.000000	41741.000000	41741.000000	41741.000000	41741.000000	14133.000000	14133.000000	28632.000000	4.174100e+04	17276.000000	41606.000000	41729
mean	346.459836	100.006820	0.499416	99.919399	0.294451	13328.104095	40.588410	33877.220453	7.108012e+04	35.192174	0.076696	0
std	110.102271	3.183821	0.288085	10.022703	0.101561	7918.698433	9.437936	36141.985884	5.225978e+04	21.629577	0.274820	0
min	0.000000	86.191572	0.000035	60.663039	0.000000	1005.180000	36.000000	0.000000	4.821180e+03	0.000000	0.000000	0
25%	270.000000	97.862546	0.251595	93.182517	0.220000	7210.280000	36.000000	9975.000000	4.401958e+04	17.000000	0.000000	0
50%	340.000000	100.017950	0.500174	99.977774	0.290000	12011.050000	36.000000	25213.000000	6.004409e+04	32.000000	0.000000	0
75%	420.000000	102.143100	0.747630	106.630991	0.360000	18030.160000	36.000000	46492.500000	8.503289e+04	50.000000	0.000000	0
max	990.000000	113.978234	0.999973	142.192400	0.900000	35059.600000	60.000000	448269.000000	5.000028e+06	141.000000	5.000000	5

Для оценки финансовой устойчивости заемщика по итогам очистки набора данных и обработки/дополнения недостающих значений получим новый датасет со следующими атрибутами:

```
target_default
score_1
score_2
score_3
score_4
score_5
score_6
risk_rate
last_amount_borrowed
last_borrowed_in_months
credit_limit
income
facebook_profile
state
real_state
ok_since
n_bankruptcies
n_defaulted_loans
n_accounts
n_issues
application_time_in_funnel
external_data_provider_credit_checks_last_month
external_data_provider_credit_checks_last_year
external_data_provider_email_seen_before
external_data_provider_fraud_score
reported_income
shipping_state
```

Проверяем новый датафрейм на наличие нулевых значений (атрибутов)

```
# Проверяем полученный новый датафрейм на наличие нулейвых значений полей (атрибу
df_credit.isnull().sum()
```

Python

```
target_default                                0
score_1                                       0
score_2                                       0
score_3                                       0
score_4                                       0
score_5                                       0
score_6                                       0
risk_rate                                     0
last_amount_borrowed                          0
last_borrowed_in_months                       0
credit_limit                                  0
income                                        0
facebook_profile                              0
state                                         0
real_state                                    0
ok_since                                      0
n_bankruptcies                                0
n_defaulted_loans                             0
n_accounts                                    0
n_issues                                      0
application_time_in_funnel                    0
external_data_provider_credit_checks_last_month    0
external_data_provider_credit_checks_last_year     0
external_data_provider_email_seen_before      0
external_data_provider_fraud_score            0
reported_income                               0
shipping_state                                0
dtype: int64
```

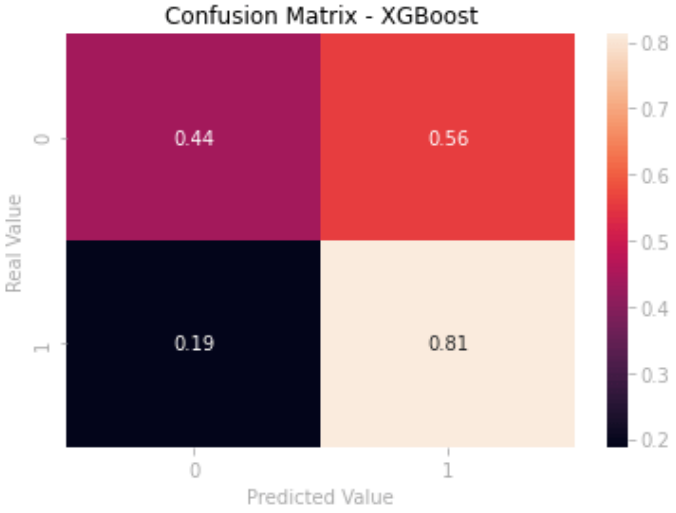
Перед тем, как приступить к настройке алгоритмов машинного обучения, необходимо провести предварительную обработку данных. Учитывая, что многие алгоритмы машинного обучения эффективнее работают с числовыми данными, мы планируем предварительно обработать наши данные с использованием LabelEncoder из библиотеки Scikit-Learn для бинарных переменных и функции get_dummies из библиотеки Pandas для остальных категориальных переменных.

Также необходимо выполнить масштабирование и преобразование данных, чтобы обеспечить более точные результаты моделей. Например, некоторые переменные могут иметь сильно различные диапазоны значений, их следует преобразовать к единому масштабу [7].

Следующим шагом необходимо проверить, как выбранные нами модели работают на тестовой выборке. Для того чтобы визуализировать результаты, строим матрицу неточностей для каждого из них.

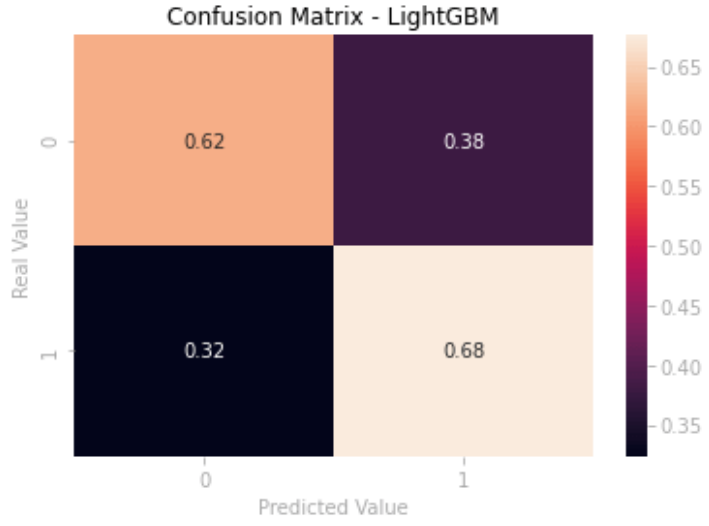
5.1.1. XGBoost

	precision	recall	f1-score	support
0	0.92	0.44	0.60	8771
1	0.22	0.81	0.34	1665
accuracy			0.50	10436
macro avg	0.57	0.63	0.47	10436
weighted avg	0.81	0.50	0.56	10436



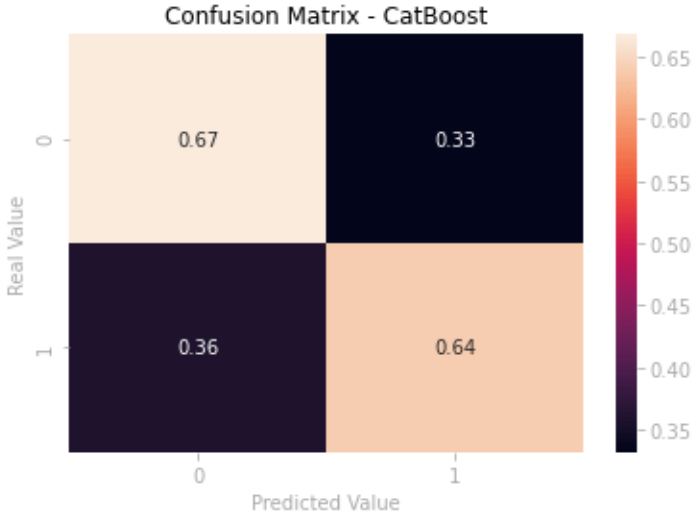
5.1.2. **LightGBM**

	precision	recall	f1-score	support
0	0.91	0.62	0.74	8771
1	0.25	0.68	0.37	1665
accuracy			0.63	10436
macro avg	0.58	0.65	0.55	10436
weighted avg	0.81	0.63	0.68	10436



5.1.3. **CatBoost**

	precision	recall	f1-score	support
0	0.91	0.67	0.77	8771
1	0.27	0.64	0.38	1665
accuracy			0.66	10436
macro avg	0.59	0.65	0.57	10436
weighted avg	0.81	0.66	0.71	10436



Результаты прогнозирования кредитоспособности первых 10 клиентов

Клиент банка	Прогнозируемая вероятность кредитоспособности	Прогнозируемый класс кредитоспособности	Реальный класс клиента
1	0,8432	IV	I
2	0,8569	IV	I
3	0,9769	IV	I
4	0,4630	II	I
5	0,9078	IV	I
6	0,8559	IV	I
7	0,9668	IV	I
8	0,8604	IV	I
9	0,7015	III	I
10	0,7418	III	I

6. Заключение

Основная цель данной дипломной работы заключается в изучении и применения методов обработки табличных данных с использованием известных моделей машинного обучения на примере оценки кредитного риска банка.

Все три исследованных модели XGBoost, LightGBM и CatBoost показали свою работоспособность по выявлению потенциальных должников на основе набора данных о клиентах финансовой компании Nubank, тем самым, способность уменьшать риски несения банками потенциальных убытков.

Предпочтительной из трех исследованных моделей стала модель с минимальным количеством ложноотрицательных результатов при выявлении возможных клиентов-неплательщиков в базе, небольшой долей ошибки отрицательной классификации для добросовестных плательщиков.

Из трех исследованных моделей машинного обучения, использующих алгоритмы повышения градиента, лучшие результаты показала модель XGBoost с коэффициентом Отзыва 81%, однако при этом метод показал 56% ложных результатов. Две другие модели - LightGBM и CatBoost - продемонстрировали более низкий процент ложных срабатываний — 38% и 33% соответственно, но при этом их доля ложноотрицательные Отзывов была существенно выше, чем у XGBoost, что привело к менее высокому значению Точности.

Дипломная работа показывает противоречие в области оценочных метрик между Точностью и Отзывом, подразумевая, что увеличение одной метрики обычно снижает другую. Соблюдение требований баланса F1 Score может быть весьма сложной задачей. Учитывая необходимость минимизации финансовых потерь банка, предлагаем уделить большее внимание снижению ложных срабатываний и исследованию оптимальных гиперпараметров, способных повысить коэффициент Отзыва. В данной ситуации принимающие решения ответственные лица банка должны провести анализ общей картины с применением алгоритмов машинного обучения и выбрать оптимальный путь действий для последующего выполнения.

7. Список литературы

1. Айвазян С.А., Бухштабер В.М., Енюков И.С., Мешалкин Л.Д. Прикладная статистика: Классификация и снижение размерности. М.: Финансы и статистика, 1989. 607 с.

2. Готовкин И. Комплексная скоринговая модель оценки дефолта клиента // Банковские технологии. 2006. № 1. С. 27–35.

3. Литвинова С.А. Скоринговые системы как средство минимизации кредитного риска банка // Аудит и финансовый анализ. 2010. № 2. С. 396–397.

4. Черный И.М. Кредитный скоринг: российский вариант развития // Банковские услуги. 2006. № 4. С. 12–17.

5. Ишина И.В., Сазонова М.Н. Скоринг — модель оценки кредитного риска // Аудит и финансовый анализ. 2007. № 4. С. 297–304.

6. Дубров А.М., Мхитарян В.С., Трошин Л.И. Многомерные статистические методы: учеб. М.: Финансы и статистика, 2003. 352 с.

7. Сорокин А.С. Построение скоринговых карт с использованием модели логистической регрессии //Науковедение. URL: http://naukovedenie.ru/PDF/180EVN214.pdf.

8. Васильев Н.П., Егоров А.А. Опыт расчета параметров логистической регрессии методом Ньютона — Рафсона для оценки зимостойкости растений // Математическая биология и биоинформатика. 2011. Т. 6. № 2. С. 190–199.

9. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. М.: Мир, 1992. 184 с.

10. Андреева Г.В. Скоринг как метод оценки кредитного риска // Банковские технологии. 2000. № 6. С. 14–19.

11. Якупов А.И. Применение деревьев решений для моделирования кредитоспособности клиентов коммерческого банка // Искусственный интеллект. 2008. № 4. С. 208–213.

12. Вапник В.Н. Восстановление зависимостей по эмпирическим данным. М.: Наука, 1979. 448 с.

13. Айвазян С.А., Енюков И.С., Мешалкин Л.Д. Прикладная статистика. Основы моделирования и первичная обработка данных. М.: Финансы и статистика, 1983. 471 с.

14. Вагин В.Н., Головина Е.Ю., Загорянская А.А., Фомина М.В. Достоверный и правдоподобный вывод в интеллектуальных системах / под ред. В.Н. Вагина, Д.А. Поспелова. М.: ФИЗМАТЛИТ, 2004. 704 с.

15. Малюгин В.И., Гринь Н.В. Об эффективности статистических алгоритмов кредитного скоринга // Банкаўскі веснік. № 31. 2010. С. 39–46.

16. Литтл Р.Дж. А., Рубин Д.Б. Статистический анализ данных с пропусками. М.: Финансы и статистика, 1990. 336 с.

17. Валеев С.Г. Регрессионное моделирование при обработке данных. Казань: ФЭН, 2001. 296 с.

18. Дрейпер Н., Смит Г. Прикладной регрессионный анализ. Множественная регрессия. М.: Диалектика, 2007. 912 с.

19. Dingyu Xue, Yangquan Chen. Solving applied mathematical Problems with MATLAB. London: Taylor & Francis Group, 2009. 418 p.

20. Powers D.M.W. Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation // Journal of Machine Learning Technologies. 2011. Vol. 2. Iss. 1. P. 37–63.

8. Приложение «Реализация исследования на языке Python»

```
!pip install catboost
     Collecting catboost
     anylinux1_x86_64.whl (66.1MB)
     ent already satisfied: graphviz in /usr/local/lib/python3.6/dist-
     packages (from catboost) (0.10.1)
     Requirement already satisfied: matplotlib in
     /usr/local/lib/python3.6/dist-packages (from catboost) (3.2.2)
     Requirement already satisfied: six in /usr/local/lib/python3.6/dist-
     packages (from catboost) (1.15.0)
     Requirement already satisfied: plotly in
     /usr/local/lib/python3.6/dist-packages (from catboost) (4.4.1)
     Requirement already satisfied: scipy in
     /usr/local/lib/python3.6/dist-packages (from catboost) (1.4.1)
     Requirement already satisfied: pandas>=0.24.0 in
     /usr/local/lib/python3.6/dist-packages (from catboost) (1.1.2)
     Requirement already satisfied: numpy>=1.16.0 in
     /usr/local/lib/python3.6/dist-packages (from catboost) (1.18.5)
     Requirement already satisfied:
     pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
     /usr/local/lib/python3.6/dist-packages (from matplotlib->catboost)
     (2.4.7)
     Requirement already satisfied: cycler>=0.10 in
     /usr/local/lib/python3.6/dist-packages (from matplotlib->catboost)
     (0.10.0)
     Requirement already satisfied: kiwisolver>=1.0.1 in
     /usr/local/lib/python3.6/dist-packages (from matplotlib->catboost)
     (1.2.0)
     Requirement already satisfied: python-dateutil>=2.1 in
     /usr/local/lib/python3.6/dist-packages (from matplotlib->catboost)
     (2.8.1)
     Requirement already satisfied: retrying>=1.3.3 in
     /usr/local/lib/python3.6/dist-packages (from plotly->catboost)
     (1.3.3)
     Requirement already satisfied: pytz>=2017.2 in
     /usr/local/lib/python3.6/dist-packages (from pandas>=0.24.0-
     >catboost) (2018.9)
     Installing collected packages: catboost
     Successfully installed catboost-0.24.2
# Импорт библиотек
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.impute import SimpleImputer
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, cross_validate,
GridSearchCV, cross_val_score
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix,
classification_report
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import StratifiedKFold
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier

# задаем стиль графических построений
sns.set_style()

# скрываем предупреждающие сообщения
import warnings
warnings.filterwarnings('ignore')
      /usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31:
      FutureWarning: The module is deprecated in version 0.21 and will be
      removed in version 0.23 since we've dropped support for Python 2.7.
      Please rely on the official version of six
      (https://pypi.org/project/six/).
        "(https://pypi.org/project/six/).", FutureWarning)
      /usr/local/lib/python3.6/dist-
      packages/sklearn/utils/deprecation.py:144: FutureWarning: The
      sklearn.neighbors.base module is  deprecated in version 0.22 and
      will be removed in version 0.24. The corresponding classes /
      functions should instead be imported from sklearn.neighbors.
      Anything that cannot be imported from sklearn.neighbors is now part
      of the private API.
        warnings.warn(message, FutureWarning)
# установливаем параметры matplotlib по умолчанию
COLOR = '#ababab'
mpl.rcParams['figure.titlesize'] = 16
mpl.rcParams['text.color'] = 'black'
mpl.rcParams['axes.labelcolor'] = COLOR
mpl.rcParams['xtick.color'] = COLOR
mpl.rcParams['ytick.color'] = COLOR
mpl.rcParams['grid.color'] = COLOR
mpl.rcParams['grid.alpha'] = 0.1
# импортируем набор данных и создайте датафрейм
df_credit =
pd.read_csv('http://dl.dropboxusercontent.com/s/xn2a4kzf0zer0xu/acquisiti
on_train.csv?dl=0')
# Просмотр визуально  данных датафрейма
df_credit.head()
                                                ids  ... target_fraud
      0  343b7e7b-2cf8-e508-b8fd-0a0285af30aa  ...
           NaN
```

```
1  bc2c7502-bbad-0f8c-39c3-94e881967124  ...
     NaN
2  669630dd-2e6a-0396-84bf-455e5009c922  ...
     NaN
3  d235609e-b6cb-0ccc-a329-d4f12e7ebdc1  ...
     NaN
4  9e0eb880-e8f4-3faa-67d8-f5cdd2b3932b  ...
     NaN

[5 rows x 43 columns]
```

```
# Выводим сведения о количестве строк и число полей (атрибутов)
print('Number of rows: ', df_credit.shape[0])
print('Number of columns: ', df_credit.shape[1])
```

```
Number of rows:  45000
Number of columns:  43
```

```
# Выводим сведения о количестве null полей (атрибутов)
df_credit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45000 entries, 0 to 44999
Data columns (total 43 columns):
 #              Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0              ids                    45000 non-null  object
 1              target_default
     41741 non-null  object
 2              score_1                44438 non-null  object
 3              score_2                44438 non-null  object
 4              score_3                44438 non-null  float64
 5              score_4                45000 non-null  float64
 6              score_5                45000 non-null  float64
 7              score_6                45000 non-null  float64
 8              risk_rate              44438 non-null  float64
 9              last_amount_borrowed                        15044
non-null  float64
 10  last_borrowed_in_months                    15044 non-null
float64
 11  credit_limit                                            31200
non-null  float64
 12  reason                     44434 non-null  object
 13  income                     44438 non-null  float64
 14  facebook_profile                           40542 non-null
object
 15  state                  44438 non-null  object
 16  zip                        44438 non-null  object
 17  channel                44438 non-null  object
 18  job_name               41664 non-null  object
 19  real_state                                      44438 non-
null  object
 20  ok_since               18455 non-null  float64
```

```
 21  n_bankruptcies                                          44303 non-
null  float64
 22  n_defaulted_loans                                       44426 non-
null  float64
 23  n_accounts                                              44438 non-
null  float64
 24  n_issues              33456 non-null  float64
 25  application_time_applied                    45000 non-null
object
 26  application_time_in_funnel             45000 non-null  int64
 27  email                 45000 non-null  object
 28  external_data_provider_credit_checks_last_2_year  22372 non-
null  float64
 29  external_data_provider_credit_checks_last_month
     45000 non-null  int64
 30  external_data_provider_credit_checks_last_year
     29876 non-null  float64
 31  external_data_provider_email_seen_before
     42767 non-null  float64
 32  external_data_provider_first_name
     45000 non-null  object
 33  external_data_provider_fraud_score
          45000 non-null  int64
 34  lat_lon               43637 non-null  object
 35  marketing_channel                                       41422 non-
null  object
 36  profile_phone_number                          45000 non-null
object
 37  reported_income                                         45000 non-
null  float64
 38  shipping_state                                          45000 non-
null  object
 39  shipping_zip_code                                       45000 non-
null  int64
 40  profile_tags                                                 45000
non-null  object
 41  user_agent                                              44278 non-
null  object
 42  target_fraud                                                 1522
non-null              object
dtypes: float64(18), int64(4), object(21)
memory usage: 14.8+ MB
```

Значения некоторых полей незаполнены

```
# процент пропущенных значений для каждого поля (атрибута)
print((df_credit.isnull().sum() * 100 /
df_credit.shape[0]).sort_values(ascending=False))
     target_fraud                                    96.617778
```

```
last_amount_borrowed                                66.568889
last_borrowed_in_months                             66.568889
ok_since                    58.988889
external_data_provider_credit_checks_last_2_year    50.284444
external_data_provider_credit_checks_last_year
      33.608889
credit_limit                                              30.666667
n_issues                    25.653333
facebook_profile                                    9.906667
marketing_channel                                         7.951111
job_name                    7.413333
target_default                                            7.242222
external_data_provider_email_seen_before
      4.962222
lat_lon                     3.028889
user_agent              1.604444
n_bankruptcies                                            1.548889
n_defaulted_loans                                         1.275556
reason                      1.257778
income                      1.248889
real_state              1.248889
state                   1.248889
zip                         1.248889
channel                     1.248889
score_3                     1.248889
score_2                     1.248889
score_1                     1.248889
n_accounts              1.248889
risk_rate               1.248889
shipping_zip_code                                   0.000000
score_4                     0.000000
score_5                     0.000000
profile_tags                                              0.000000
score_6                     0.000000
application_time_in_funnel              0.000000
shipping_state                                      0.000000
reported_income                                     0.000000
application_time_applied                                          0.000000
profile_phone_number                                              0.000000
external_data_provider_fraud_score                  0.000000
external_data_provider_first_name                   0.000000
external_data_provider_credit_checks_last_month     0.000000
email                                               0.000000
ids                                                 0.000000
dtype: float64
```

```
# удалим все записи, где target_default (наша целевая переменная) равно null.
df_credit.dropna(subset=['target_default'], inplace=True)
# Удалим колонку с атрибутом "target_fraud", в которй отсутствуют почти все записи
df_credit.drop('target_fraud', axis=1, inplace=True)
```

```
# количество уникальных значений для каждого атрибута
df_credit.nunique().sort_values()
      channel                                          1
      external_data_provider_credit_checks_last_2_year 1
      last_borrowed_in_months                          2
      target_default                                   2
      facebook_profile                                 2
      external_data_provider_credit_checks_last_year   2
      external_data_provider_credit_checks_last_month  4
      real_state                                       5
      n_defaulted_loans                                5
      email                                            6
      n_bankruptcies                                   6
      score_1                                          7
      marketing_channel                                9
      shipping_state                                   25
      score_2                                          35
      n_issues                                         44
      n_accounts                                       44
      state                                            50
      external_data_provider_email_seen_before         62
      risk_rate                                        81
      score_3                                          87
      ok_since                                         100
      user_agent                                       297
      application_time_in_funnel                       501
      zip                                              823
      external_data_provider_fraud_score               1001
      last_amount_borrowed                             13480
      reason                                           14260
      credit_limit                                     19336
      lat_lon                                          21596
      profile_tags                                     24458
      shipping_zip_code                                26996
      job_name                30543
      external_data_provider_first_name                31183
      application_time_applied                         33560
      reported_income                                  37368
      income                                           38849
      score_5                                          41741
      profile_phone_number                             41741
      score_4                                          41741
      score_6                                          41741
      ids                                              41741
      dtype: int64
# Удалим колонки "channel" и
"external_data_provider_credit_checks_last_2_year"
df_credit.drop(labels=['channel',
'external_data_provider_credit_checks_last_2_year'], axis=1,
inplace=True)
# удалим некоторые столбцы, которые не добавляют ценности модели
```

```
df_credit.drop(labels=['email', 'reason', 'zip', 'job_name',
'external_data_provider_first_name', 'lat_lon', 'shipping_zip_code',
'user_agent', 'profile_tags', 'marketing_channel',
'profile_phone_number', 'application_time_applied', 'ids'], axis=1,
inplace=True)
# проверим, есть ли в наборе данных выбросы. Покажем статистику
df_credit.describe()
                          score_3  ...  reported_income
      count  41741.000000  ...                  41741.0
      mean             346.459836  ...              inf
      std              110.102271  ...              NaN
      min              0.000000  ...               403.0
      25%              270.000000  ...             50910.0
      50%              340.000000  ...            101623.0
      75%              420.000000  ...            151248.0
      max              990.000000  ...              inf

      [8 rows x 20 columns]
# количество значений "inf" в "reported_income"
np.isinf(df_credit['reported_income']).sum()
66
# количество значений = -999 в "external_data_provider_email_seen_before"
df_credit.loc[df_credit['external_data_provider_email_seen_before'] == -
999, 'external_data_provider_email_seen_before'].value_counts()
-999.0  591
Name: external_data_provider_email_seen_before, dtype: int64
```

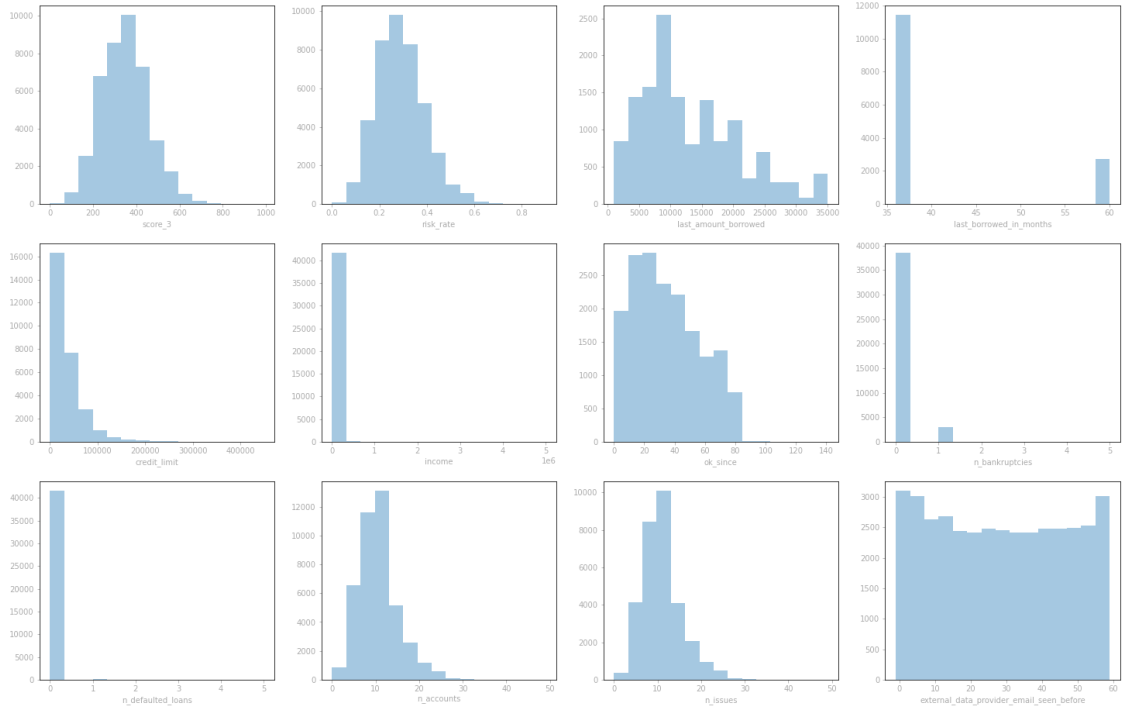
заменим эти значения "inf" и "-999" на NaN, чтобы мы могли построить гистограммы для визуализации распределения значений

```
# замена "inf" на "nan"
df_credit['reported_income'] =
df_credit['reported_income'].replace(np.inf, np.nan)

# замена "-999" на "nan"
df_credit.loc[df_credit['external_data_provider_email_seen_before'] == -
999, 'external_data_provider_email_seen_before'] = np.nan
```

Построим гистограммы для этих характеристик, что поможет изучить их распределение

```
# фрейм данных, содержащий числовые характеристики
df_credit_numerical = df_credit[['score_3', 'risk_rate',
'last_amount_borrowed','last_borrowed_in_months', 'credit_limit',
'income', 'ok_since','n_bankruptcies', 'n_defaulted_loans', 'n_accounts',
'n_issues', 'external_data_provider_email_seen_before']]
# построим гистограммы для каждого из приведенных выше объектов

nrows = 3
ncols = 4
```

```
fig, ax = plt.subplots(nrows=nrows, ncols=ncols, figsize=(25, 16))

r = 0
c = 0

for i in df_credit_numerical:
  sns.distplot(df_credit_numerical[i], bins=15,kde=False, ax=ax[r][c])
  if c == ncols - 1:
  r += 1
  c = 0
  else:
  c += 1

plt.show()
```



Все перечисленные выше объекты имеют пропущенные значения, которые необходимо обработать. Как мы можем видеть, они имеют искаженное распределение, что указывает на то, что мы должны заполнить пропущенные значения медианным значением для каждого объекта.

Заполняем значения из оставшихся 32 столбцов с пропущенными значениями в соответствии с особенностями каждого признака, как показано ниже:

- Категориальные переменные будут заполнены наиболее повторяющимся значением.
- Числовые переменные будут заполнены их средними значениями.
- В конкретных случаях last_amount_borrowed, last_borrowed_in_months и n_issues мы заполним недостающие значения нулем, поскольку разумно полагать, что не каждому клиенту будут присвоены значения этим переменным.

```
df_credit_num = df_credit.select_dtypes(exclude='object').columns
df_credit_cat = df_credit.select_dtypes(include='object').columns

# fill missing values for "last_amount_borrowed",
"last_borrowed_in_months" and "n_issues"
df_credit['last_amount_borrowed'].fillna(value=0, inplace=True)
df_credit['last_borrowed_in_months'].fillna(value=0, inplace=True)
df_credit['n_issues'].fillna(value=0, inplace=True)

# fill missing values for numerical variables
imputer = SimpleImputer(missing_values=np.nan, strategy='median')
imputer = imputer.fit(df_credit.loc[:, df_credit_num])
df_credit.loc[:, df_credit_num] = imputer.transform(df_credit.loc[:,
df_credit_num])

# fill missing values for categorical variables
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
imputer = imputer.fit(df_credit.loc[:, df_credit_cat])
df_credit.loc[:, df_credit_cat] = imputer.transform(df_credit.loc[:,
df_credit_cat])
# Проверяем полученный новый датафрейм на наличие нулейвых значений полей
(атрибутов)
df_credit.isnull().sum()
      target_default                                          0
      score_1                0
      score_2                0
      score_3                0
      score_4                0
      score_5                0
      score_6                0
      risk_rate              0
      last_amount_borrowed                                    0
      last_borrowed_in_months                                 0
      credit_limit                 0
      income                 0
      facebook_profile                                        0
      state                  0
      real_state             0
      ok_since               0
      n_bankruptcies                                          0
      n_defaulted_loans                                             0
      n_accounts             0
      n_issues               0
      application_time_in_funnel                              0
      external_data_provider_credit_checks_last_month  0
      external_data_provider_credit_checks_last_year                0
      external_data_provider_email_seen_before                      0
      external_data_provider_fraud_score                            0
      reported_income                                         0
      shipping_state                                          0
```

```
dtype: int64
```

Проведем предварительную обработку данных, преобразовав категориальные признаки в числовые значения. LabelEncoder будет использоваться для двоичных переменных, в то время как get_dummies будет использоваться для других категориальных переменных.

```
bin_var = df_credit.nunique()[df_credit.nunique() == 2].keys().tolist()
num_var = [col for col in df_credit.select_dtypes(['int',
'float']).columns.tolist() if col not in bin_var]
cat_var = [col for col in
df_credit.select_dtypes(['object']).columns.tolist() if col not in
bin_var]

df_credit_encoded = df_credit.copy()

# label encoding for the binary variables
le = LabelEncoder()
for col in bin_var:
  df_credit_encoded[col] = le.fit_transform(df_credit_encoded[col])

# encoding with get_dummies for the categorical variables
df_credit_encoded = pd.get_dummies(df_credit_encoded, columns=cat_var)

df_credit_encoded.head()
                    target_default  score_3  ...  shipping_state_BR-SP
shipping_state_BR-TO
0                                0    350.0  ...                     0
                                 0
1                                0    370.0  ...                     0
                                 0
2                                1    360.0  ...                     0
                                 0
3                                0    510.0  ...                     0
                                 0
4                                0    500.0  ...                     0
                                 0

[5 rows x 144 columns]
```

Работа над моделями машинного обучения.

```
# Feature  matrix – это инструмент управления продуктом, который помогает
командам определить, какие функции разрабатывать дальше.
X = df_credit_encoded.drop('target_default', axis=1)

# целевая переменная (целевой атрибут)
y = df_credit_encoded['target_default']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle=True,
stratify=y)
```

Поскольку мы имеем дело с несбалансированным набором данных, мы стандартизируем и повторно выполним выборку обучающего набора с помощью StandardScaler и RandomUnderSampler соответственно.

```
# стандартизируем числовые переменные
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)

# повторная выборка
rus = RandomUnderSampler()
X_train_rus, y_train_rus = rus.fit_sample(X_train, y_train)
# определим функцию val_model
def val_model(X, y, clf, show=True):
  """
  Apply cross-validation on the training set.

# Arguments                X: DataFrame containing the independent
variables.
                y: Series containing the target vector.
                clf: Scikit-learn estimator instance.

  # Returns float, mean value of the cross-validation scores.
  """

  X = np.array(X)
  y = np.array(y)

  pipeline = make_pipeline(StandardScaler(), clf)
  scores = cross_val_score(pipeline, X, y, scoring='recall')

  if show == True:
                print(f'Recall: {scores.mean()}, {scores.std()}')

  return scores.mean()
#оценим модели
xgb = XGBClassifier()
lgb = LGBMClassifier()
cb = CatBoostClassifier()

model = []
recall = []

for clf in (xgb, lgb, cb):
  model.append(clf.__class__.__name__)
  recall.append(val_model(X_train_rus, y_train_rus, clf, show=False))

pd.DataFrame(data=recall, index=model, columns=['Recall'])
# XGBoost
```

```
xgb = XGBClassifier()

# параметр, подлежащий поиску
param_grid = {'n_estimators': range(0,1000,50)}

# найдем наилучший параметр
kfold = StratifiedKFold(n_splits=3, shuffle=True)
grid_search = GridSearchCV(xgb, param_grid, scoring="recall", n_jobs=-1,
cv=kfold)
grid_result = grid_search.fit(X_train_rus, y_train_rus)

print(f'Best result: {grid_result.best_score_} for
{grid_result.best_params_}')
Best result: 0.6657327195142321 for {'n_estimators': 50}
# XGBoost
xgb = XGBClassifier(n_estimators=50)

# параметр, подлежащий поиску
param_grid = {'max_depth': [1, 3, 5], 'min_child_weight': [1, 3, 6]}

# найдем наилучший параметр
kfold = StratifiedKFold(n_splits=3, shuffle=True)
grid_search = GridSearchCV(xgb, param_grid, scoring="recall", n_jobs=-1,
cv=kfold)
grid_result = grid_search.fit(X_train_rus, y_train_rus)

print(f'Best result: {grid_result.best_score_} for
{grid_result.best_params_}')
Best result: 0.6701307550047045 for {'max_depth': 3, 'min_child_weight':
6}
# XGBoost
xgb = XGBClassifier(n_estimators=50, max_depth=3, min_child_weight=6)

# параметр, подлежащий поиску
param_grid = {'gamma': [0, 1, 5]}

# найдем наилучший параметр
kfold = StratifiedKFold(n_splits=3, shuffle=True)
grid_search = GridSearchCV(xgb, param_grid, scoring="recall", n_jobs=-1,
cv=kfold)
grid_result = grid_search.fit(X_train_rus, y_train_rus)

print(f'Best result: {grid_result.best_score_} for
{grid_result.best_params_}')
Best result: 0.6719385652158761 for {'gamma': 1}
# XGBoost
xgb = XGBClassifier(n_estimators=50, max_depth=3, min_child_weight=6,
gamma=1)

# параметр, подлежащий поиску
param_grid = {'learning_rate': [0.0001, 0.001, 0.01, 0.1]}
```

```
# найдем наилучший параметр
kfold = StratifiedKFold(n_splits=3, shuffle=True)
grid_search = GridSearchCV(xgb, param_grid, scoring='recall', n_jobs=-1,
cv=kfold)
grid_result = grid_search.fit(X_train_rus, y_train_rus)

print(f'Best result: {grid_result.best_score_} for
{grid_result.best_params_}')
Best result: 0.8170548699960465 for {'learning_rate': 0.0001}
# LightGBM
lbg = LGBMClassifier(silent=False)

# параметр, подлежащий поиску
param_grid = {"max_depth": np.arange(5, 75, 10),
                      "learning_rate" : [0.001, 0.01, 0.1],
                      "num_leaves": np.arange(20, 220, 50),
                      }

# найдем наилучший параметр
kfold = StratifiedKFold(n_splits=3, shuffle=True)
grid_search = GridSearchCV(lbg, param_grid, scoring="recall", n_jobs=-1,
cv=kfold)
grid_result = grid_search.fit(X_train_rus, y_train_rus)

print(f'Best result: {grid_result.best_score_} for
{grid_result.best_params_}')
Best result: 0.6883483723819858 for {'learning_rate': 0.01, 'max_depth':
5, 'num_leaves': 70}
lbg = LGBMClassifier(learning_rate=0.01, max_depth=5, num_leaves=50,
silent=False)

# параметр, подлежащий поиску
param_grid = {'min_data_in_leaf': np.arange(100, 1000, 100)}

# найдем наилучший параметр
kfold = StratifiedKFold(n_splits=3, shuffle=True)
grid_search = GridSearchCV(lbg, param_grid, scoring="recall", n_jobs=-1,
cv=kfold)
grid_result = grid_search.fit(X_train_rus, y_train_rus)

print(f'Best result: {grid_result.best_score_} for
{grid_result.best_params_}')
Best result: 0.6961582230489793 for {'min_data_in_leaf': 400}
# CatBoost
cb = CatBoostClassifier()

# параметр, подлежащий поиску
param_grid = {'depth': [6, 8, 10],
                      'learning_rate': [0.03, 0.1],
                      'l2_leaf_reg': [1, 5, 10],
                      }
```

```
# найдем наилучший параметрkfold = StratifiedKFold(n_splits=3,
shuffle=True)
grid_search = GridSearchCV(cb, param_grid, scoring="recall", n_jobs=-1,
cv=kfold)
grid_result = grid_search.fit(X_train_rus, y_train_rus)

print(f'Best result: {grid_result.best_score_} for
{grid_result.best_params_}')
```

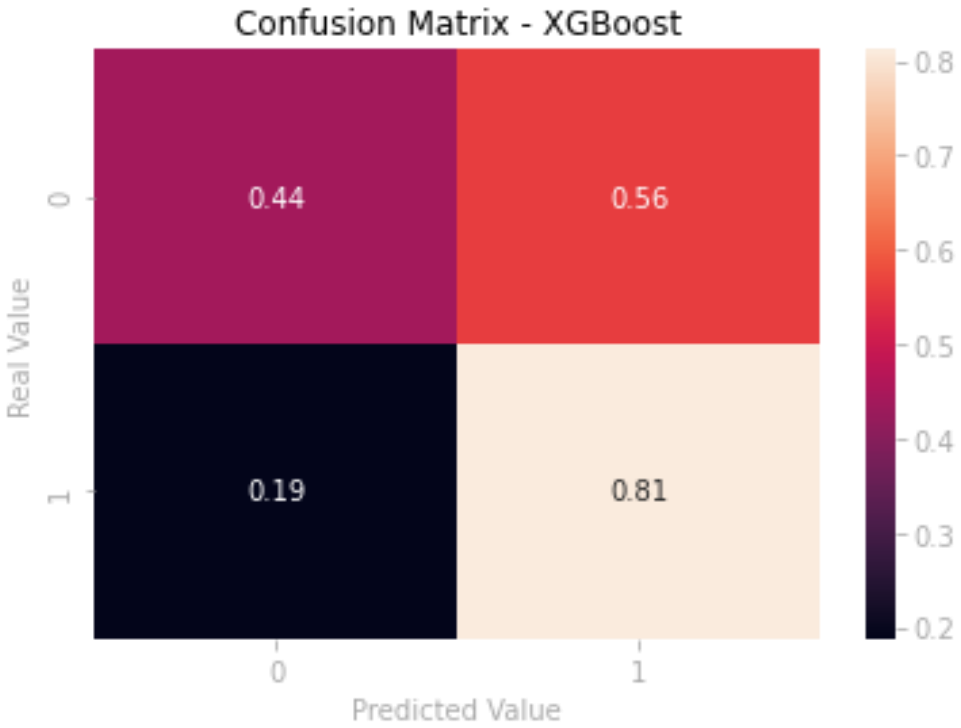
```
     0:    learn: 0.6895401 total: 9.95ms    remaining: 9.94s
     1:    learn: 0.6860938 total: 18.7ms    remaining: 9.31s
     2:    learn: 0.6829368 total: 26.7ms    remaining: 8.88s
     3:    learn: 0.6798871 total: 34.4ms    remaining: 8.57s
     4:    learn: 0.6772491 total: 42.5ms    remaining: 8.45s
     5:    learn: 0.6746800 total: 50.1ms    remaining: 8.31s
     6:    learn: 0.6724516 total: 58ms      remaining: 8.23s
     …
     995:  learn: 0.4568061 total: 8.17s     remaining: 32.8ms
     996:  learn: 0.4566922 total: 8.18s     remaining: 24.6ms
     997:  learn: 0.4565555 total: 8.19s     remaining: 16.4ms
     998:  learn: 0.4564079 total: 8.19s     remaining: 8.2ms
     999:  learn: 0.4563245 total: 8.2s      remaining: 0us
     Best result: 0.6561276762957435 for {'depth': 6, 'l2_leaf_reg': 5,
     'learning_rate': 0.03}
```

Матрица неточностей (англ. Confusion Matrix) — это таблица или диаграмма, показывающая точность прогнозирования классификатора в отношении двух и более классов. Прогнозы классификатора находятся на оси X, а результат (точность) — на оси Y.

```
# финальная модель XGBoost
xgb = XGBClassifier(max_depth=3, learning_rate=0.0001, n_estimators=50,
gamma=1, min_child_weight=6)
xgb.fit(X_train_rus, y_train_rus)

# Прогноз модели (Prediction)
X_test_xgb = scaler.transform(X_test)
y_pred_xgb = xgb.predict(X_test_xgb)

# вывод на экран классификации
print(classification_report(y_test, y_pred_xgb))

# confusion matrix
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_test, y_pred_xgb, normalize='true'),
annot=True, ax=ax)
ax.set_title('Confusion Matrix - XGBoost')
ax.set_xlabel('Predicted Value')
ax.set_ylabel('Real Value')
```

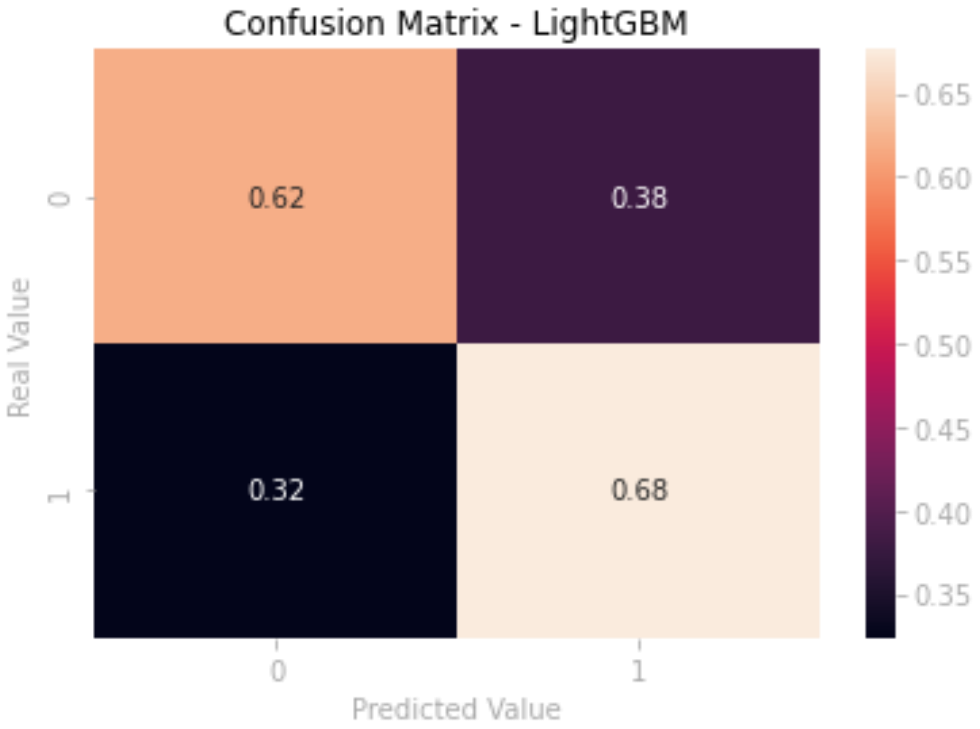
```
plt.show()
```

```
                           precision  recall  f1-score            support

                           0                  0.92              0.44
          0.60             8771
                           1                  0.22              0.81
          0.34             1665

        accuracy                                    0.50             10436
                    macro avg              0.57              0.63
          0.47             10436
      weighted avg                  0.81              0.50              0.56
                10436
```




```
# final LightGBM model
lgb = LGBMClassifier(num_leaves=70, max_depth=5, learning_rate=0.01,
min_data_in_leaf=400)
lgb.fit(X_train_rus, y_train_rus)

# Прогноз модели (Prediction)
X_test_lgb = scaler.transform(X_test)
y_pred_lgb = lgb.predict(X_test_lgb)

# вывод на экран классификации
print(classification_report(y_test, y_pred_lgb))

# confusion matrix
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_test, y_pred_lgb, normalize='true'),
annot=True, ax=ax)
ax.set_title('Confusion Matrix - LightGBM')
```

```
ax.set_xlabel('Predicted Value')
ax.set_ylabel('Real Value')

plt.show()
                    precision  recall  f1-score            support

                    0                0.91            0.62
     0.74           8771
                    1                0.25            0.68
     0.37           1665

  accuracy                                 0.63            10436
                 macro avg           0.58            0.65
     0.55           10436
weighted avg                0.81            0.63            0.68
     10436
```




```
# final CatBoost model
cb = CatBoostClassifier(learning_rate=0.03, depth=6, l2_leaf_reg=5,
logging_level='Silent')
cb.fit(X_train_rus, y_train_rus)

# Прогноз модели (Prediction)
X_test_cb = scaler.transform(X_test)
y_pred_cb = cb.predict(X_test_cb)

# Сведения о классификации
print(classification_report(y_test, y_pred_cb))

# Матрица неточностей (англ. Confusion Matrix) — это таблица или
диаграмма,
```

```
# показывающая точность прогнозирования классификатора в отношении двух и более классов.
# Прогнозы классификатора находятся на оси X, а результат (точность) — на оси Y.

fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_test, y_pred_cb, normalize='true'), annot=True, ax=ax)
ax.set_title('Confusion Matrix - CatBoost')
ax.set_xlabel('Predicted Value')
ax.set_ylabel('Real Value')

plt.show()
```

```
                    precision    recall  f1-score   support

                    0       0.91      0.67      0.77      8771
                    1       0.27      0.64      0.38      1665

  accuracy                           0.66     10436
                 macro avg       0.59      0.65      0.57     10436
weighted avg            0.81      0.66      0.71     10436
```