МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**по курсу**

«Data Science Pro»

**Тема:** «**Прогнозирование вероятности дефолта клиентов»**

Слушатель **Щекина Татьяна Сергеевна**

Москва, 2025

**Содержание**

[Введение 3](#__RefHeading___Toc5172_1330607712)

[1 Аналитическая часть 4](#__RefHeading___Toc5200_1330607712)

[1.1 Постановка задачи 4](#__RefHeading___Toc5198_1330607712)

[1.2 Описание используемых методов 6](#__RefHeading___Toc5196_1330607712)

[1.2.1 Логистическая регрессия 6](#__RefHeading___Toc5210_1330607712)

[1.2.2 Дерево решений 7](#__RefHeading___Toc9083_1330607712)

[1.2.3 Ансамблевые методы 8](#__RefHeading___Toc9079_1330607712)

[1.2.3.1 Адаптивный бустинг AdaBoost 9](#__RefHeading___Toc7347_1832855893)

[1.2.3.2 Экстремальный градиентный бустинг XGBoost 10](#__RefHeading___Toc7349_1832855893)

[1.2.3.3 Градиентный бустинг LightGBM 11](#__RefHeading___Toc3076_229205395)

[1.2.4 Метрики качества моделей классификации 11](#__RefHeading___Toc5194_1330607712)

[1.3 Разведочный анализ данных 13](#__RefHeading___Toc5192_1330607712)

[2 Практическая часть 14](#__RefHeading___Toc5190_1330607712)

[2.1 Предобработка данных 14](#__RefHeading___Toc5188_1330607712)

[2.2 Разработка и обучение моделей 17](#__RefHeading___Toc5186_1330607712)

[2.3 Разработка нейронной сети 21](#__RefHeading___Toc5182_1330607712)

[2.4 Тестирование модели 22](#__RefHeading___Toc5184_1330607712)

[2.5 Сохранение модели 23](#__RefHeading___Toc1017_229205395)

[2.6 Создание удаленного репозитория GitHub и загрузка результатов 24](#__RefHeading___Toc5178_1330607712)

[Заключение 25](#__RefHeading___Toc1505_1560672135)

[Библиографический список 27](#__RefHeading___Toc1507_1560672135)

**Введение**

Тема данной работы — прогнозирование дефолта клиентов по платежам, на основании данных, предоставленных банком Home Credit Bank в 2018 году для соревнования на платформе Kaggle.

Многим людям сложно получить кредит из-за недостаточной или отсутствующей кредитной истории. Задача состоит в построении модели, которая будет оценивать вероятность беспроблемного обслуживания долга заявителем на основании имеющейся статики, которая включает в себя не только кредитную историю клиентов, но также и многочисленные альтернативные данные, включая информацию о телекоммуникационных компаниях и транзакциях, здании, в котором проживает клиент, рейтинга региона, для прогнозирования платежеспособности своих клиентов.

1. **Аналитическая часть**
   1. **Постановка задачи**

Сформулированная организаторами соревнования Kaggle задача является классической задачей классификации: нулевой класс соответствует беспроблемным кредитам, первый класс — кредитам, по которым заявитель допускал просрочки более некоторого количества дней.

В рамках этой работы поставлена задача:

* провести разведочный анализ данных
* при необходимости провести предобработку данных
* добавить в основной датасет агрегированные показатели, используя данные из бюро кредитных историй, данные кредитных карт и предыдущих кредитных заявок
* провести отбор наиболее информативных переменных-предикторов
* применить методы классического машинного обучения к задаче классификации, сравнить их эффективность
* разработать нейронную сеть для решения задачи классификации
* при построении моделей провести поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10.
* создать публичный репозиторий в GitHub и разместить там код и документацию исследования

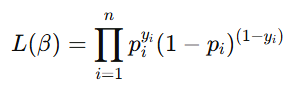
Предоставленный датасет состоит из 7 основных файлов в табличном виде. Рассмотрим их по-порядку:

1. application\_{train|test} – это основная таблица, разбитая на 2 файла — для обучения и тестирования, содержит статические данные для всех кредитных заявок. Одна строка представляет один кредит в выборке данных. В обучающей выборке содержится 307 тысяч заявок, в тестовой — 48 тысяч заявок. Количество признаков (не считая целевой переменной и технического идентификатора заявки SK\_ID\_CURR ) — 120. Для данной исследовательской работы будет использоваться только файл с обучающей выборкой: 70% выборки используется для обучения моделей, 30% - для тестирования моделей.
2. bureau - все предыдущие кредиты клиента, предоставленные другими финансовыми учреждениями, о которых было сообщено в кредитное бюро (для клиентов, имеющих кредит в нашей выборке). Для каждого кредита в нашей выборке имеется столько строк, сколько кредитов было у клиента в кредитном бюро до даты подачи заявки. Содержит 1.7 млн записей и 15 признаков (не считая ключей SK\_ID\_CURR и SK\_BUREAU\_ID, которые используются для джойна между таблицами).
3. bureau\_balance - ежемесячные остатки по предыдущим кредитам в кредитном бюро. В этой таблице есть одна строка для каждого месяца истории каждого предыдущего кредита, сообщенного в кредитное бюро, т.е. в таблице есть строки (количество кредитов в выборке \* количество относительных предыдущих кредитов \* количество месяцев, в которых у нас есть некоторая история, наблюдаемая для предыдущих кредитов). Таблица содержит 27.3 млн записей и 2 признака.
4. POS\_CASH\_balance - ежемесячные справки о балансе предыдущих кредитов наличными, которые клиент имел в банке Home Credit. По одной записи для каждого кредита, каждый месяц. Таблица содержит 10.0 млн записей и 6 признаков.
5. credit\_card\_balance - ежемесячные балансы кредитных карт, которые заявитель имел в Home Credit. Таблица содержит 3.8 млн записей и 21 признак.
6. previous\_application - все предыдущие заявки на кредиты Home Credit клиентов, имеющих кредиты в нашей выборке. Таблица содержит 1.6 млн записей и 35 признаков.
7. installments\_payments - история погашения ранее выданных кредитов в Home Credit, относящихся к кредитам в нашей выборке. Имеется a) одна строка для каждого произведенного платежа плюс b) одна строка для каждого пропущенного платежа. Таблица содержит 13.6 млн записей и 6 признаков.
   1. **Описание используемых методов**
      1. **Логистическая регрессия**

Логистическая регрессия использует логистическую функцию (сигмоиду) для моделирования вероятности принадлежности объекта к классу 1. Логистическая функция задается формулой:

 (1)

Оптимизационная задача формулируется как максимизация правдоподобия (Maximum Likelihood Estimation, MLE). Функция правдоподобия для бинарной классификации:

 (2)

где:

* – вероятность класса 1 для объекта iii,
* – истинная метка (0 или 1),
* ​ – вектор признаков,
*  – вектор коэффициентов.

Часто вместо функции правдоподобия максимизируют ее логарифм (логарифмическую функцию правдоподобия, log-likelihood).

В отличие от линейной регрессии, логистическая регрессия не имеет аналитического решения и на практике для поиска оптимального вектора коэффициентов используются численные методы. Самым распространенным методом численной оптимизации является градиентный спуск.

К преимуществам метода относятся:

* интерпретируемость коэффициентов
* математическая обоснованность (модель основана на вероятностном подходе)
* возможность расширить до обобщенных линейных моделей при использовании регуляризации L1 или L2

К недостаткам метода относятся:

* линейность (предполагается что логарифм вероятности линейно зависит от признаков)
* работает плохо при большом количестве признаков. Эта проблема решается добавлением регуляризации
* плохо справляется с нелинейными зависимости без дополнительного преобразования признаков

Для практической реализации алгоритма логистической регрессии используется LogisticRegression из библиотеки sklearn.linear\_model.

* + 1. **Дерево решений**

**Процесс построения деревьев решений заключается в последовательном, рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Процесс разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями. Объявление узла листом может произойти естественным образом (когда он будет содержать единственный объект, или объекты только одного класса), или по достижении некоторого условия остановки, задаваемого пользователем (например, минимально допустимое число примеров в узле или максимальная глубина дерева).**

**Теоретически, алгоритм обучения дерева решений будет работать до тех пор, пока в результате не будут получены абсолютно «чистые» подмножества, в каждом из которых будут примеры одного класса. Правда, возможно при этом будет построено дерево, в котором для каждого примера будет создан отдельный лист. Очевидно, что такое дерево окажется бесполезным, поскольку оно будет переобученным — каждому примеру будет соответствовать свой уникальный путь в дереве, а следовательно, и набор правил, актуальный только для данного примера.**

**Решением проблемы является принудительная остановка построения дерева, пока оно не стало переобученным. Для этого разработаны следующие подходы:**

* **ранняя остановка — алгоритм останавливается при достижении некоторого критерия**
* **ограничение глубины дерева**
* **задание минимально допустимого числа примеров в узле. Это позволит избежать создания тривиальных разбиений и малозначимых правил.**

К преимуществам метода относятся:

* интерпретируемость
* гибкость — деревья могут выявлять нелинейные зависимости
* не требуется нормализация признаков (признаки могут иметь различный масштаб)

Для практической реализации алгоритма дерева решений используется DecisionTreeClassifier из библиотеки sklearn.tree.

Ключевые параметры алгоритма:

* criterion – критерий разбиения (gini или entropy или log\_loss), по умолчанию применяется gini
* max\_depth – максимальная глубина дерева (защита от переобучения)
* min\_samples\_split – минимальное число объектов для разбиения узла
* min\_samples\_leaf – минимальное число объектов в листе
* max\_features – число признаков, учитываемых при разбиении.
  + 1. **Ансамблевые методы**

Ансамблевые методы строят несколько моделей (базовых алгоритмов или слабых моделей) и объединяют их предсказания, чтобы получить более устойчивый и точный результат. Объединение происходит по методу голосования слабых моделей. Существует два основных типа ансамблей — это бустинг и бэггинг. В данной работе мы используем несколько бустинговых ансамблей. Бустинг последовательно обучает слабые модели, исправляя ошибки предыдущих. Каждая новая модель пытается предсказать ошибки предыдущих, а финальное предсказание получается путем взвешенной суммы предсказаний всех моделей. Все три метода LightGBM, AdaBoost и XGBoost используют деревья решений в качестве базовых моделей.

* + - 1. **Адаптивный бустинг AdaBoost**

Адаптивный бустинг — один из наиболее популярных алгоритмов бустинга. Ключевой особенностью алгоритма AdaBoost является использование всей обучающей выборке для каждого последующего классификатора, от итерации к итерации изменяются только веса каждого объекта таким образом, что вес верно классифицированных объектов уменьшается, а ошибочно классифицированных - увеличивается. Алгоритм итеративно обучает T моделей, финальный классификатор строится как взвешенная сумма предсказаний всех моделей, вес модели определяется на основании ошибки моделей — более точные модели имеют больший вес. На каждом из шагов производится выборка с заменой, то есть один объект может попасть в обучающую выборку более одного раза.

К недостаткам метода относится низкая скорость, относительно других методов бустинга.

Для практической реализации классификации адаптивного бустинга AdaBoost используется AdaBoostClassifier из библиотеки sklearn.ensemble. Ключевые параметры алгоритма:

* n\_estimators – количество слабых классификаторов. По умолчанию их количество равно 50
* learning\_rate – вес, применяемый к каждому классификатору на каждой итерации. Более высокая скорость обучения увеличивает вклад каждого классификатора. По умолчанию значение равно единице
* estimator – базовый классификатор, по умолчанию используется DecisionTreeClassifier(max\_depth=1) — так называемые «пни», но можно заменить на другой классификатор, например на более глубокое дерево или логистическую регрессию.
  + - 1. **Экстремальный градиентный бустинг XGBoost**

XGBoost строит последовательность деревьев решений, используя градиентный спуск для минимизации ошибки.

1. **Инициализация**: модель начинается с базового предсказания (например, среднего значения целевой переменной).
2. **Обучение деревьев** (итеративный процесс):
   * Каждое новое дерево обучается на остатках ошибок предыдущей модели.
   * Веса ошибок рассчитываются с учетом градиента (направления, в котором модель должна скорректироваться).
   * В отличие от классического бустинга, XGBoost использует второй производный член (Гессиан) для более точного обновления весов.
3. **Обновление предсказаний**: предсказание итоговой модели — это сумма всех деревьев. Функция потерь включает L1/L2-регуляризацию, что снижает переобучение.

Для практической реализации классификации экстремального градиентного бустинга XGBoost используется XGBClassifier из библиотеки sklearn.ensemble.

К ключевым гиперпараметрам относятся:

* n\_estimators – количество деревьев
* max\_depth – глубина деревьев
* learning\_rate – скорость обучения
* gamma – минимальный прирост информации, необходимый для разбиения (аналог штрафа на сложность)
* lambda и alpha – коэффициенты L1 и L2 регуляризации соответственно
* scale\_pos\_weight — для несбалансированных классов параметр увеличивает вес редкого класса

Алгоритм XGBoost быстро и эффективно работает на больших данных (более 10 тысяч объектов).

* + - 1. **Градиентный бустинг LightGBM**

LightGBM — это мощный алгоритм градиентного бустинга, оптимизированный для высокой скорости и работы с большими данными. Он улучшает XGBoost за счёт более эффективного построения деревьев.

Как и XGBoost, LightGBM строит последовательность деревьев решений, обучающихся на остатках ошибок предыдущей модели.

Главное отличие – метод роста деревьев:

* XGBoost и классический градиентный бустинг строят деревья по уровням (level-wise), то есть добавляют разбиения равномерно на каждом уровне.
* LightGBM использует рост по листьям (leaf-wise): выбирает самый информативный лист и делит его.

Такой подход ускоряет обучение, но может привести к переобучению (если не контролировать глубину дерева).

Для практической реализации классификации градиентного бустинга LightGBM используется LGBMClassifier из библиотеки lightgbm.

Ключевые параметры алгоритма:

* n\_estimators – количество деревьев
* learning\_rate – скорость обучения
* max\_depth – ограничивает глубину деревьев (защита от переобучения).
* num\_leaves – число листьев в дереве (чем больше, тем сложнее модель).
* is\_unbalance=True – автоматически корректирует веса классов.
* scale\_pos\_weight – вручную задаёт баланс классов.
  + 1. **Метрики качества моделей классификации**

Существует множество различных метрик для оценки качества классификаторов. К наиболее широко используемым относятся:

1. матрица ошибок (confusion matrix)
2. правильность (accuracy), в некоторых источниках также называется меткостью
3. правильность с учетом цены (cost-sensitive accuracy)
4. точность, полнота и F-мера (precision и recall, F-score)
5. площадь под кривой рабочей характеристики приемника (Receiver Operating Characteristic, ROC) - AUC

Матрица ошибок для задачи бинарной классификации представляет собой матрицу размером 2 x 2, по одной оси располагается метка, предсказанная моделью, по другой оси — фактическая метка. При этом в различных источниках горизонтальная и вертикальные оси будут быть подписаны по-разному. В этой работе буду использовать матрицу ошибок в том виде, как представлено на рисунке 1.

|  |  |  |
| --- | --- | --- |
|  | Положительная предсказанная метка | Отрицательная предсказанная метка |
| Положительная фактическая метка | TP | FN |
| Отрицательная фактическая метка | FP | TN |

Рисунок 1 — матрица ошибок

Правильность определяется как отношение правильно классифицированных экземпляров к общему количеству:

Правильность = TP + TN / (TP + TN + FP + FN) (1)

В нашем случае датасет несбалансирован — отрицательных меток в 11 раз больше, чем положительных. В этом случае имеет смысл использовать правильность с учетом цены, то есть присваивать ложно-положительным и ложно-отрицательным результатам разные веса.

Точность (precision) это отношение истинно положительных прогнозов к общему количеству положительных прогнозов:

Точность = TP / (TP + FP) (2)

Полнота (recall) определяется как отношение истинно положительных прогнозов к обществу количеству положительных экземпляров:

Полнота = TP / (TP + FN) (3)

Полнота и точность не зависят, в отличие от правильности, от соотношения классов и подходят лучше для нашей задачи.

F-мера представляет собой агрегированный критерий качества, объединяющий собой полноту и точность с некоторыми весами, которые зависят от задачи и предпочтений заказчика. F-мера рассчитывается как среднее гармоническое с весами по формуле.

F-меру удобно использовать в процессе оптимизации параметров алгоритма, ее значение достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

ROC кривая строится на основании комбинации доли истинно-положительных результатов (TPR) и доли ложно-положительных результатов (FPR). Диапазон оценок достоверности дискретизируется, затем каждое дискретное значение используется в качестве порога при прогнозировании. Чем больше площадь под ROC кривой, тем лучше классификатор.

Организаторы соревнования Kaggle определили в качестве основной метрики качества моделей площадь под кривой ROC. В работе будут использоваться несколько метрик качества классификаторов, поскольку не все рассматриваемые алгоритмы возвращают вероятность принадлежности к классу, а не сам класс.

* 1. **Разведочный анализ данных**

Разведочный анализ данных включает в себя:

* нормировку данных — нормировка данных может быть полезна для дальнейшей интерпретации весов модели (например, для логистической регрессии)
* анализ на аномалии (пропущенные данные, выбросы)
* корректировка аномалий там, где это возможно

Разведочный анализ данных, перечисленный выше, был также дополнен агрегированным показателями, отсутствующими в исходных таблицах.

1. **Практическая часть**
   1. **Предобработка данных**

Визуализация первичных данных позволяет наглядно увидеть, какие из признаков имеют значительные выбросы. Некоторые признаки из датасета уже нормализованы (такие как внешние рейтинги EXT\_RATINGS 1-3).

Предобработка данных состояла из последовательных шагов:

1. Анализ бинарных и категориальных признаков. Для категориальных признаков с большим количеством категорий (более 10) была сделана смысловая группировка с целью сокращения количества категорий.
2. Добавление дополнительных числовых признаков, которые характеризуют кредитную нагрузку клиента, а также агрегированные показатели из бюро кредитных историй:
   * Отношение аннуитетного платежа к кредиту (APPS\_ANNUITY\_CREDIT\_RATIO) – показывает, какую часть кредита заемщик выплачивает ежегодно.
   * Среднее и стандартное отклонение внешних скоринговых оценок (APPS\_EXT\_SOURCE\_MEAN, APPS\_EXT\_SOURCE\_STD) – характеризуют уровень и стабильность кредитного риска.
   * Отношение стоимости товара к кредиту (APPS\_GOODS\_CREDIT\_RATIO) – отражает долю кредита в финансировании покупки.
   * Отношение аннуитетного платежа к доходу (APPS\_ANNUITY\_INCOME\_RATIO) – измеряет кредитную нагрузку заемщика.
   * Отношение дохода к стажу работы (APPS\_INCOME\_EMPLOYED\_RATIO) – средний доход за день работы.
   * Отношение стажа к возрасту (APPS\_EMPLOYED\_BIRTH\_RATIO) – доля жизни, проведенная в занятости.
   * Возраст авто к стажу работы (APPS\_CAR\_EMPLOYED\_RATIO) – отражает момент приобретения автомобиля относительно стажа.
   * Доход на день жизни (APPS\_INCOME\_BIRTH\_RATIO) – общий уровень финансового состояния.
   * Разница между запрошенной и одобренной суммой кредита (PREV\_CREDIT\_DIFF) – показывает, на сколько сумма заявки превышала фактически выданный кредит.
   * Разница между запрошенной суммой и стоимостью товара (PREV\_GOODS\_DIFF) – отражает разницу между суммой заявки и ценой покупки.
   * Отношение одобренного кредита к запрашиваемой сумме (PREV\_CREDIT\_APPL\_RATIO) – оценивает, какую долю от заявки одобрил банк.
   * Отношение стоимости товара к запрашиваемой сумме (PREV\_GOODS\_APPL\_RATIO) – показывает, насколько заявка соотносится с реальной стоимостью покупки.
   * Процентная ставка кредита (PREV\_INTERESTS\_RATE) – рассчитывает среднюю эффективную ставку на основе суммы выплат (AMT\_ANNUITY \* CNT\_PAYMENT) и суммы кредита.
   * Разница между плановой и фактической датой закрытия кредита (BUREAU\_ENDDATE\_FACT\_DIFF) – показывает, насколько фактическое закрытие отклоняется от запланированного.
   * Разница между датой выдачи кредита и его фактическим закрытием (BUREAU\_CREDIT\_FACT\_DIFF) – отражает срок фактического погашения кредита.
   * Разница между датой выдачи кредита и плановой датой закрытия (BUREAU\_CREDIT\_ENDDATE\_DIFF) – показывает первоначально запланированный срок кредита.
   * Отношение текущего долга к сумме кредита (BUREAU\_CREDIT\_DEBT\_RATIO) – оценивает, какая доля кредита остается непогашенной.
   * Разница между текущим долгом и суммой кредита (BUREAU\_CREDIT\_DEBT\_DIFF) – показывает величину уже выплаченного кредита (отрицательное значение означает, что долг превышает сумму кредита).
   * Флаг наличия просрочки (BUREAU\_IS\_DPD) – принимает значение 1, если есть текущая просрочка.
   * Флаг серьезной просрочки (более 120 дней) (BUREAU\_IS\_DPD\_OVER120) – указывает на длительную задолженность.RR.

Для дополнительных фичей из таблиц бюро кредитной истории и предыдущих заявок были дополнительно применены функции агрегации (минимальное, максимальное, среднее значения, количество записей и т.д.)

1. Поскольку не по всем кредитам есть информация в кредитном бюро, то по отсутствующим значениям признаки заполнены нулями.
2. Удаление признаков с высоким процентом отсутствующих значений (в качестве критерий выбран 25%).
3. Значения типа «бесконечность» заменяются отсутствующими значениями NaN
4. Применяется анализ на выбросы методом межквартильного размаха. Метод 3-сигм работает плохо в нашем случае, поскольку данные в основном имеют асиметричное распределение
5. Для признаков с процентом заполнения более 75% применяется стандартный Imputer из библиотеки sklearn: для категориальных и бинарных переменных используется наиболее частотное значение, для числовых — среднее значение.
6. Для категориальных переменных применяется метод One-Hot Encoding, цель которого преобразовать каждый признак с N категориями в N-1 бинарный признак.
7. Для непрерывных переменных применяется стандартизация MinMaxScaler() либо StandardScaler().

Следует отметить, что применение StandardScaler() может быть не самым удачным решением для данного датасета, поскольку очень много признаков с непрерывным распределением имеют асимметричное распределение, в то время как StandardScaler() хорошо работает для распределения, приближенного к нормальному. В качестве одного из способов улучшения пайплайна может быть внедрение шага трансформации асиметричных признаков с помощью log-трансформации или box-cox трансформации.

* 1. **Разработка и обучение моделей**

Для решения задачи прогнозирования дефолта обученные модели из следующих классов:

* Логистическая регрессия
* Дерево решений
* Градиентный бустинг LightGBM
* Адаптивный бустинг AdaBoost
* Экстремальный градиентный бустинг XGBoost
* Многослойная полносвязная нейронная сеть (перцептрон)

После процедуры подготовки данных (расчета статистических показателей, объединения таблиц и стандартизации данных) в выборке содержится 307 тысяч строк и 303 признака. Выборка была разбита на обучающую и тестовую в отношении 70% на 30% (252 и 92 тысячи), при разбиении было сохранено соотношение классов в целевой переменной, для этого использовался метод stratify.

Некоторые рассматриваемые методы имеют встроенный механизм работы с несбалансированными классами (class\_weight="balanced"): это относится к дереву решений и логистической регрессии. Для обучения же ансамблевых методов и нейронной сети был построена сбалансированная обучающая выборка методом Undersampling: размер обучающей выборки сократился до 34 тысяч.

Учитывая ограничения времени и ресурсов компьютера, я решила также сократить количество признаков, оставив только 50 самых значимых для прогнозирования целевой переменной выборки. Для отбора значимость использовался метод feature\_importances\_ библиотеки LightGBM.

|  |  |
| --- | --- |
| **Признак** | **Описание** |
| APPS\_ANNUITY\_CREDIT\_RATIO | Отношение аннуитета к телу кредита |
| DAYS\_BIRTH | Возраст клиента в днях на момент подачи заявки |
| APPS\_EXT\_SOURCE\_MEAN | Среднее значение внешних скорингов |
| EXT\_SOURCE\_2 | Нормализованный скоринговый балл из внешнего источника |
| DAYS\_ID\_PUBLISH | Количество дней до подачи заявки, когда клиент сменил документ, удостоверяющий личность |
| APPS\_ANNUITY\_INCOME\_RATIO | Отношение аннуитета к доходу |
| EXT\_SOURCE\_3 | Нормализованный скоринговый балл из внешнего источника |
| APPS\_GOODS\_CREDIT\_RATIO | Отношение стоимости товара к телу кредита |
| AMT\_ANNUITY | Аннуитетный платеж по кредиту |
| EXT\_SOURCE\_1 | Нормализованный скоринговый балл из внешнего источника |
| DAYS\_LAST\_PHONE\_CHANGE | Количество дней до подачи заявки, когда клиент сменил телефон |
| APPS\_INCOME\_EMPLOYED\_RATIO | Отношение дохода к длительности рабочего стажа |
| PREV\_CNT\_PAYMENT\_MEAN | Среднее количество платежей по предыдущим заявкам, выполненных к моменту подачи текущей кредитной заявки |
| APPS\_EXT\_SOURCE\_STD | Стандартное отклонение значений внешних скорингов |
| DAYS\_REGISTRATION | Количество дней до подачи заявки, когда клиент сменил регистрацию |
| APPS\_EMPLOYED\_BIRTH\_RATIO | Отношение трудового стажа к возрасту |
| PREV\_PREV\_INTERESTS\_RATE\_MAX | Максимальное значение процентной ставки в предыдущих заявках |
| PREV\_DAYS\_DECISION\_MIN | Минимальное количество дней с дня принятия решения по предыдущим заявкам |
| REGION\_POPULATION\_RELATIVE | Нормализованное население региона, где живет клиент (чем выше значение, тем более населенный регион) |
| PREV\_DAYS\_DECISION\_MAX | Максимальное количество дней с дня принятия решения по предыдущим заявкам |
| BUREAU\_DAYS\_ENDDATE\_FACT\_MAX | Максимальное количество дней с даты закрытия предыдущего кредита |
| APPS\_INCOME\_BIRTH\_RATIO | Отношение дохода к возрасту |
| PREV\_PREV\_CREDIT\_APPL\_RATIO\_MAX | Максимальное отношение фактической суммы кредита к величине одобренной суммы |
| DAYS\_EMPLOYED | Количество дней до подачи заявки, когда клиент начал текущую работу |
| PREV\_PREV\_INTERESTS\_RATE\_MEAN | Минимальное значение процентной ставки в предыдущих заявках |

Таблица 1 —

описание 25 самых значимых признаков

На примере модели LightGBM можно увидеть, что балансировка классов имеет критическое значение для построения адекватной модели: показали f1, recall и precision модели обученной на несбалансированных данных в разы ниже сбалансированной модели.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AUC-ROC** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Train** | 0.86 | 0.92 | 0.85 | 0.08 | 0.14 |
| **Test** | 0.78 | 0.92 | 0.55 | 0.04 | 0.07 |

Таблица 2 —

метрики LightGBM на несбалансированных классах

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AUC-ROC** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Train** | 0.84 | 0.72 | 0.2 | 0.81 | 0.32 |
| **Test** | 0.78 | 0.71 | 0.17 | 0.7 | 0.28 |

Таблица 3 —

метрики LightGBM на сбалансированных классах

Для модели дерева решений была выполнена процедура поиска оптимальных параметров GridSearchCV с перекрестной проверкой с количеством блоков 10. Оптимизируемые параметры — глубина дерева и функция измерения качества разбиения (gini / entropy). Получены оптимальные параметры - {'criterion': 'entropy', 'max\_depth': 5}. Отмечу, что глубина дерева 5 совпадает с глубиной, выбранной для модели lightGBM (для которой подбор гиперпараметров не проводился).

Обучение модели логистической регрессии занимало значительное количество времени, поэтому поиск гиперпараметров по сетке не производился. Обучение логистической регрессии было выполнено в двух вариантах — с и без регуляризации “L1”, результаты модели с регуляризацией значительно превосходят модель без регуляризации.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AUC-ROC** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Train** | 0.66 | 0.63 | 0.13 | 0.6 | 0.21 |
| **Test** | 0.67 | 0.63 | 0.13 | 0.61 | 0.21 |

Таблица 4 —

метрики логистической регрессии без регуляризации

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AUC-ROC** | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
| **Train** | 0.75 | 0.69 | 0.16 | 0.67 | 0.26 |
| **Test** | 0.75 | 0.69 | 0.16 | 0.68 | 0.26 |

Таблица 5 —

метрики логистической регрессии с регуляризацией “L1”

* 1. **Разработка нейронной сети**

Для построения и обучения нейросети используется библиотека tensorflow.keras.

На вход нейронной сети подается стандартизованный и сбалансированный датасет. Датасет сбалансирован методом undersampling.

Исходя из количества признаков на входе (50), на первом слое выбрано 64 нейрона (минимальная степень двойки, которая превышает количество признаков на входе), на втором слое количество нейронов вдвое меньше, на выходе — 1 нейрон, соответствующий единственной целевой переменной. На скрытых слоях использована функция активации relu, на выходном слое — sigmoid, позволяющая на выходе получить значения от 0 до 1 (соответствуют вероятности первого класса). В качестве функции потерь использовалась бинарная кросс-энтропия, являющаяся стандартным выбором для задачи бинарной классификации. Слои BatchNorm и DropOut защищают от переобучения модели. Используем метод EarlyStopping, который мониторит метрику val\_loss и если метрика перестает улучшаться в течении 5 эпох, автоматически прекращает обучение нейросети



Рисунок — Архитектура

полносвязной нейронной сети

* 1. **Тестирование модели**

Каждый из рассмотренных классификаторов продемонстрировал точность выше, чем базовая модель, а модель LightGBM показала наилучший результат.

|  |  |  |
| --- | --- | --- |
| **Классификатор** | **Тренировочная выборка** | **Тестовая выборка** |
| Адаптивный бустинг AdaBoost | 0.7603 | 0.7577 |
| Градиентный бустинг LightGBM | 0.8286 | 0.7676 |
| Дерево решений | 0.7294 | 0.7278 |
| Логистическая регрессия | 0.7453 | 0.7488 |
| Многослойная нейронная сеть | 0.7610 | 0.5000 |
| Экстремальный градиентный бустинг XGBoost | 0.8980 | 0.7565 |

Таблица 6 — Сравнение метрики AUC

на тренировочном и тестовом датасетах

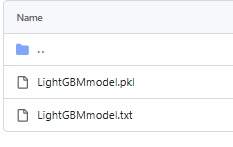
|  |  |  |
| --- | --- | --- |
| **Классификатор** | **Тренировочная выборка** | **Тестовая выборка** |
| Адаптивный бустинг AdaBoost | 0.2672 | 0.2674 |
| Градиентный бустинг LightGBM | 0.3107 | 0.2739 |
| Дерево решений | 0.2508 | 0.3362 |
| Логистическая регрессия | 0.2581 | 0.2595 |
| Многослойная нейронная сеть | 0.3336 | 0.2946 |
| Экстремальный градиентный бустинг XGBoost | 0.4040 | 0.2852 |

Таблица 7 — Сравнение метрики F1-score

на тренировочном и тестовом датасетах

* 1. **Сохранение модели**

По результатам тестирования наилучшей моделью является модель LightGBM. Для дальнейшего в приложении результаты этой модели были сохранены в текстовом формате и формате pickle. Кроме того, в файл requirements.txt сохранена версионность библиотек, использованных на этапе обучения.



* 1. **Создание удаленного репозитория GitHub и загрузка результатов**

Для целей данного исследования был создан публичный репозиторий на GitHub по адресу <https://github.com/Spring-31/HC_default_risk>, в котором размещена данная пояснительная записка, метаданные по исходному датасету, результирующие характеристики моделей, рабочие ноутбуки Jupyter, обученная модель LightGBM.

**Заключение**

Для достижения поставленной цели был проведен полноценный pipeline машинного обучения, включающий в себя следующие этапы:

* изучение теоретических основ анализа данных и машинного обучения: типы классификаторов (деревья решений, ансамблевые методы, логистическая регрессия), методы обучения (понятие тренировочной, тестовой и валидационной метрики), метрик качества классификаторов
* изучение библиотек python для применения теоретических знаний на практике: scikit-learn, tensorflow, keras классификаторы библиотек scikit-learn
* знакомство с особенностями предметной области, в которой решается задача (оценка риска дефолта клиентов-физических лиц)
* проведение первичного анализа данных с использованием статистических методов
* обработка данных: извлечение данных, работа с типами данных, идентификация и удаление аномалий, заполнение отсутствующих значений, стандартизация данных
* инженерия признаков: генерация новых признаков (извлечение статистик, агрегация данных), преобразование данных, объединение данных, отбор значимых признаков)
* определение метрик измерения эффективности моделей машинного обучения, выбор наиболее подходящей для решаемой задачи метрики
* подбор оптимальных гиперпараметров для повышения эффекттивности работы моделей машинного обучения

В рамках проведенного исследования удалось познакомиться с каждым из перечисленных выше этапов. Наиболее эффективная модель классического машинного обучения характеризуется значением AUC 0.7676, в то время как показатель лидеров в соревновании Kaggle превышает 0.802.

Что касается разработки многослойного перцептрона, обучить хоть сколько-то эффективную нейронную сеть не удалось. При беглом анализе результатов соревнования Kaggle также не удалось найти решение, в котором использовались бы полносвязные нейронные сети. Среди лучших решений применялись модели LightGBM, XGBoost, Random Forest и ExtraTrees. Например, команда, занявшая 4-е место, использовала стек из моделей LightGBM, Random Forest, ExtraTree и Linear Regression.

С моей точки зрения, одно из слабых мест разработанных классификаторов — слишком большое количество признаков, не оптимальный выбор нормализаторов — для асимметричных распределений использование StandardScaler не оптимально.

Дальнейшие шаги по улучшению качества моделей могли бы включать:

* Более тонкую настройку стандартизации непрерывных переменных: к примеру, использование лог-трансформации непрерывных признаков с асимметричным распределением
* Снижение размерности, сокращение признаков в выборке, например методом главных компонент (PCA)
* Использование upsampling вместо understampling для балансировки обучающей выборки с целью не потерять значимую информацию для нулевого класса
* Дальнейший подбор гиперпараметров (невозможен в текущем исследовании из-за большого количества признаков)
* Обучение деревьев на случайных подвыборках (bagging и выбор признаков)
* Построение результирующей модели при помощи стекинга нескольких ансамблей

**Библиографический список**

1. Anna Montoya, inversion, KirillOdintsov, and Martin Kotek. Home Credit Default Risk. <https://kaggle.com/competitions/home-credit-default-risk> , 2018. Kaggle.
2. VanderPlas, Python Data Science Handbook [Текст]: Python Data Science Handbook: Essential Tools for Working with Data [2 ed.]. 2022
3. Бурков Андрей. Машинное обучение без лишних слов [Текст] — Спб.: Питер, 2020. - 192 с.
4. Рашка, Себастьян, Мирджалили, Вахид. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2, 3-е изд.: Пер. с англ. - СПб. : ООО "Диалектика", 2020. - 848 с.
5. Уэс Маккинни. Python и анализ данных: Первичная обработка данных с применением pandas, NumPy и Jupiter / пер. с англ. А. А. Слинкина. 3-е изд. – М.: МК Пресс, 2023. – 536 с.: ил.
6. Логистическая регрессия кредитного риска – Режим доступа <https://fastercapital.com/content/Credit-Risk-Logistic-Regression--How-to-Use-Logistic-Regression-to-Estimate-the-Probability-of-Default.html>
7. Документация по библиотеке numpy: – Режим доступа: https://numpy.org/
8. Документация по библиотеке pandas: – Режим доступа: <https://pandas.pydata.org/>
9. Документация по библиотеке matplotlib: – Режим доступа: https://matplotlib.org/
10. Документация по библиотеке numpy: – Режим доступа: https://scikit-learn.org/stable/api/index.html
11. Документация по библиотеке sklearn: – Режим доступа: https://scikit-learn.org/stable/api/index.html
12. Документация по библиотеке Tensorflow Keras – Режим доступа: <https://www.tensorflow.org/guide/keras>
13. Сравнение 11 классификаторов https://www.kaggle.com/code/davidfumo/comparing-11-classification-models
14. Документация по библиотеке LightGBM – Режим доступа: <https://lightgbm.readthedocs.io/en/stable/>
15. Документация по библиотеке seaborn: – Режим доступа: <https://seaborn.pydata.org/tutorial.html>
16. Документация по библиотеке impalanced-learn: – Режим доступа: [https://imbalanced-learn.org/stable/user\_guide.html#user-guide](https://imbalanced-learn.org/stable/user_guide.html" \l "user-guide)
17. Loginom Вики. Алгоритмы: – Режим доступа: <https://wiki.loginom.ru/algorithms.html>
18. Top 3 Methods for Handling Skewed Data - Режим доступа: <https://towardsdatascience.com/top-3-methods-for-handling-skewed-data-1334e0debf45/>
19. Деревья решений: общие принципы- Режим доступа: <https://loginom.ru/blog/decision-tree-p1>