**Риск невыполнения обязательств с использованием глубокого обучения**

Многие люди с трудом получают ссуды[[1]](#footnote-1)из-за плохой кредитной истории или ее отсутствия. Такие компании, как Home Credit, стремятся расширить доступ к финансовым услугам для населения, не охваченного банковской сферой, путем предоставления положительного и безопасного опыта заимствования. Чтобы убедиться, что у этого населения есть положительный опыт кредитования, Home Credit использует различные альтернативные данные (например, информацию о телефонных компаниях и транзакциях) для прогнозирования платежеспособности своих клиентов. Используя различные методы статистики, машинного обучения и глубокого обучения, раскрывается весь потенциал данных. Это гарантирует, что клиенты, способные к погашению, не будут отклонены и что ссуды будут выдаваться с указанием основной суммы, срока и календаря погашения, что позволит их клиентам добиться успеха.

Цель этого проекта - использовать исторические данные о заявках на получение ссуды, чтобы предсказать, сможет ли заявитель выплатить ссуду. Это стандартная задача классификации:

* Под наблюдением: метки, включающиеся в обучающие данные, и цель состоит в том, чтобы обучить модель научиться предсказывать метки на основе функций.
* Классификация: метка представляет собой двоичную переменную, 0 (погашает ссуду вовремя), 1 (будет трудно погасить ссуду).

Данные были взяты с сайта kaggle, ссылка на данные указана в использованной литературе.

Данная тема несет особую актуальность в текущее время, ссуда более выгодна для заемщика, т. к. она основывается на безвозмездной основе, в случае отказа, клиент ищет другие способы достичь желаемой цели, и подвергается большому риску попасть в очень невыгодные условия кредитования после отказа в ссуде, а иногда и вовсе оказаться на быстрых займах у мошенников. Поэтому, чтобы обезопасить и банк, и клиента, необходимо тщательно проработать механизм на выдачу ссуды заемщику.

В качестве оценки метрик качества модели в своем исследовании в исследовании использована функция roc\_curve, которая вычисляет рабочую характеристическую кривую приемника или кривую ROC[[2]](#footnote-2).

Формулу AUC, численно можно записать следующим образом:

*,*

Перед выполнением поставленной задачи, мною были исследованы и проанализированы различные решения с kaggle, ссылки которых расположены в списке использованных источников. Данные исследований основаны на применении простых моделей машинного обучения, их изучение помогло мне лучше узнать данные и, в какой-то, мере предметную область. Чаще всего авторы использовали метрику качества ROC\_AUC, данная метрика очень распространена для задач бинарной классификации, т. к. она учитывает сбалансированность классов. Рассмотрим более подробно одно из исследований, в котором автор использовал следующие методы:

* LabelEncoder – для преобразования категориальных переменных, данный служебный класс, помогает нормализовать метки таким образом, чтобы они содержали только значения от 0 до 1;
* Corr – для выявления взаимосвязей между признаками;
* Imputer – для работы с пропусками в данных;
* LogisticRegression – в качестве выбора baseline модели (полученный score 0.671);
* RandomForestClassifier, LGBMClassifier – в качестве улучшения работы baseline модели (полученный score 0.735);
* KFold – в качестве деления и обучения данных.

Мною был проведен предварительный анализ данных, который помог выявить закономерности для дальнейшей работы. В качестве определения зависимости был использован коэффициент корреляции метода Спирмена, который определил зависимость признаков от нашей целевой переменной, и показал результат, что тесная взаимосвязь прослеживается у:

* DAYS\_ID\_PUBLISH - за сколько дней до подачи заявки клиент сменил документ, удостоверяющий личность, с которым подавал заявку на получение кредита;
* NAME\_FAMILY\_STATUS – семейное положение;
* NAME\_HOUSING\_TYPE - жилищная ситуация клиента (аренда, проживание с родителями и пр.);
* OWN\_CAR\_AGE – возраст машины клиента, этот показатель также указывает на ее фактическое наличие;
* NAME\_CONTRACT\_TYPE – тип продукта договора (Кредит наличными, потребительский кредит и пр.) предыдущей заявки;
* DAYS\_EMPLOYED – сколько дней отработал клиент на текущем месте работы;
* CODE\_GENDER – гендерный фактор (мужчина/женщина);
* NAME\_EDUCATION\_TYPE – уровень образования;
* DAYS\_BIRTH – возраст клиента;
* AMT\_CREDIT - объема ссуды, т. е. объем суммы, которую запрашивает клиент, оценки внешнего источника данных.

На первый взгляд, всё логично и выдача ссуды зависит от стандартных факторов при кредитовании.

В данных имелось множество пропусков, для их заполнения я использовала обычный алгоритм константы, данный алгоритм был выбран мной, чтобы выделить людей, по которым нет данных в отдельную группу, т.к. закономерность, в данном случае, нельзя чем-либо определить. Например, у группы клиентов отсутствует DAYS\_EMPLOYED, что это может значить? У клиента нет работы, или это просто намеренно незаполненное поле, или работа неофициальная? В ходе исследования я выяснила, что если удалить пропуски, то теряется более 90% информации, и в данном случае некорректно заполнять пропуски чем-либо, это может испортить качество данных и работу модели, поэтому более релевантной работой будет определить клиентов в отдельную группу.

Были обработаны и исследованы категориальные переменные, переменные, которые содержали более 10 типов были исключены из анализа, поскольку их преобразование могло привести к перегрузки будущей модели. Также была преобразована переменная DAYS\_BIRTH, которая содержала отрицательные значения поскольку возраст клиента рассчитывался относительно подачи заявки

После первичной обработки данных выборка была разделена на тренировочную и тестовый в соответствии с пропорцией 70/30. После визуализации основной переменной относительно признаком стало очевидным, что в качестве baseline модели лучше всего подойдет LogisticRegression, чтобы избежать переобучения использовался параметр С=0.0001, модель показала хороший результат 92%, но вот метрики оценки качества модели были низкие, площадь под кривой AUC показала результат в 0.59, что очень близко к результату случайной модели. Логистическая регрессия практически никак не регулирует сильный дисбаланс классов, наш дисбаланс был примерно 92% и 8% от всей выборки, поэтому в дальнейшем я сосредоточилась на том, чтобы повысить именно метрики качества, а не результат модели. Логистическую регрессию математически можно записать:

Для определения других закономерностей в данных была применена модель LGBMClassifier, которую я обучала дважды, усреднив в последующем результаты, чтобы избежать переобучения. Данный алгоритм был использован для цели прохождения этапа Feature Selection, т.е. был установлен определенный порог важностей признаков в 90%, после чего признаки ниже порога, были исключены из анализа. Таким образом, выборка была очищена от признаков, которые имели либо нулевой вес, либо слишком низкий вес, либо корреляция была нулевой, это позволило избавиться от шумов в данных, а также удешевить процесс сбора данных и сократить время обработки данных и запуска модели. В построение модели были настроены гиперпараметы: boosting\_type='goss'[[3]](#footnote-3), настроено достаточно глубокое дерево с помощью ‘n\_estimators’, а также установлен параметр ‘class\_weight’, который помог учесть несбалансированность классов . Выбор модели обоснован возможностью работы с пропусками.

LightGBM расширяет алгоритм градиентного бустинга, добавляя тип автоматического выбора объектов, а также фокусируясь на примерах бустинга с большими градиентами. Алгоритм может привести к резкому ускорению обучения и улучшению прогнозных показателей.

LGBM помог выявить другие закономерности, например, внешние оценки EXT\_SOURCE\_1, EXT\_SOURCE\_2, EXT\_SOURCE\_3 также оказывают влияние на нашу переменную, я решила использовать эти похожие метрики для генерации объединяющей оценки, использовав для этого полиномиальную функцию PolynomialFeatures[[4]](#footnote-4), указав не слишком большую глубину равную 2ум, опять же, чтобы избежать переобучения. На выходе алгоритма модель создала 10 новых признаков.

В ходе изучения данных также были сгенерированы 4 новых признака из предметной области:

- CREDIT\_INCOME\_PERCENT: процент суммы кредита относительно дохода клиента;

- ANNUITY\_INCOME\_PERCENT: процент аннуитета ссуды относительно дохода клиента;

- CREDIT\_TERM: продолжительность платежа в месяцах;

- DAYS\_EMPLOYED\_PERCENT: процент отработанных дней по отношению к возрасту клиента;

Первой попыткой повышения качества модели был вышеописанный алгоритм LightGBMClassifier, гиперпараметры которого были подобраны следующим образом:

* objective='binary' – указывает на бинарную классификацию;
* boosting\_type = 'gbdt' – указывает на тип бустинга, традиционное дерево решений градиентного повышения;
* n\_estimators = 100 – кол-во деревьев;
* is\_unbalance=True – указывает на дисбаланс классов.

Данная модель заметно повысила метрику качества и площадь под кривой ROC\_AUC в 0.72, что говорит о том, что данной модели можно больше доверять. Следует учесть, что перед повышением качества модели данные были стандартизированы и применялись в такой нормализации в последующих моделях.

В качестве улучшения оценки и метрики качества второй модели сначала был построен StackingClassifier, который показал качество в ROC\_AUC 0.74, а потом был проведен эксперимент в подборе гиперпараметров для ансамблевой модели стекинга. Алгоритм является составным обобщением, который состоит из суммирования выходных данных отдельного оценщика и использования классификатора для вычисления окончательного прогноза. Стекинг позволяет использовать силу каждого отдельного оценщика, используя их выходные данные в качестве входных данных для окончательного оценщика.

Для подбора гиперпаметров можно потратить много времени и ресурсов, причем не всегда параметры будут логичны и очевидны для модели, поэтому для решения данной проблемы был проведен эксперимент в подборе гиперпараметров, который состоял из 2ух частей:

1. Исследовать каждую модель из структуры Стекинга отдельно и воспользоваться различными методами подборов гиперпараметров: для более структурно сложных моделей – байесовская оптимизация, для средних моделей – сеточный поиск;
2. Исследовать всю модель целиком используя библиотеку optuna[[5]](#footnote-5).

Стоить отметить, что для получения наиболее качественной и выгодной модели использовались различные значения гиперпараметров в путях решения.

После подбора оптимальных гиперпараметров был построен Стекинг, который имел следующую структуру:

* SVC- классификатор C-опорных векторов;
* LGBMClassifier – алгоритм описан выше в данном докладе;
* RandomForestClassifier - метаоценка, которая соответствует ряду классификаторов дерева решений на различных подвыборках набора данных и использует усреднение для повышения точности прогнозирования и контроля избыточной подгонки;
* KNeighborsClassifier – классификатор, реализующий голосование k-ближайших соседей;
* В качестве финальной модели для принятия решения была использована LogisticRegression.

Путь1 – подбор параметров по отдельности. Для подбора гиперпараметров LGBM была использована библиотека hyperopt, которая основывается на байесовской оптимизации. Байесовская оптимизация использует вероятность, чтобы найти минимум функции. Конечная цель - найти входное значение для функции, которая может дать нам минимально возможное выходное значение. Поэтому в алгоритме hyperopt мы используем нашу метрику качества со знаком «-», поскольку данную метрику необходимо максимизировать.

Байесовская оптимизация более эффективна, чем случайный, сеточный или ручной поиск. Поэтому байесовская оптимизация может привести к повышению производительности на этапе тестирования и сокращению времени оптимизации. Наиболее подходящие параметры по результатам оптимизации алгоритма LGBM: is\_unbalance=True, max\_depth=9, max\_features=2, n\_estimators=80, num\_leaves=10, objective='binary'.

Для подборов параметров RandomForest и KNeighbors был использован тот же алгоритм, что и для LGBM. Для модели RandomForest алгоритм показал следующий результат для гиперпараметров: bootstrap=False, max\_features=2, min\_samples\_leaf=5, min\_samples\_split=0.107, n\_estimators=1500. Для knn наиболее оптимальные гиперпараметры: n\_neighbors=50.

Для модели LogisticRegression использовался алгоритм GridSearch[[6]](#footnote-6), в котором мы настраиваем сетку гиперпараметров и обучаем /тестируем модель на каждой из возможных комбинаций, параметр cv - делит тренировочную выборку. Сеточный поиск определил, что гиперпараметры LogisticRegression должны быть: C=1000, penalty=l2.

Поскольку алгоритм GridSearch перебирает и проверяет каждое значение из сетки, он становиться времязатратным, поэтому подбора гиперпараметров модели SVC использовался случайный поиск алгоритма RandomizedSearchCV: C=1, gamma=10

Путь2 – подбор параметров для всей модели. Для подбора всех гиперпараметров моделей был использован один и тот же алгоритм optuna. Преимущество алгоритма заключается в его скорости, optuna также алгоритм байесовской оптимизации, он более чем в 3 раза быстрее полного перебора и более, чем в 1.5 раза быстрее, чем hyperopt, несмотря на то, что в обоих случаях используется TPE и EI. В данной задачи — это наиболее важно из-за большого массива данных, алгоритм сэкономит колоссальное количество времени, а также за счет использования большего количества итераций, позволит привести к лучшему решению.

Наиболее оптимальные гиперпараметры для моделей по решению алгоритма являются:

1. LGBM: boosting\_type='dart', drop\_rate=7.608827030451904e-07, learning\_rate=0.059564331166248735, max\_depth=4, min\_data\_in\_leaf=116, num\_leaves=16, objective='binary'
2. RandomForest: max\_features='sqrt', min\_samples\_leaf=5, n\_estimators=649
3. Knn: algorithm='ball\_tree', n\_neighbors=3, weights='distance'
4. LogisticRegression: C=0.009614329288366374, solver='saga'
5. SVC: C=0.01151679563583205, gamma=0.029107251885168536, kernel='linear', penalty= 'l2'.

Описание гиперпараметров:

1. LGBM

* Learning\_rate - повышение скорости обучения;
* boosting\_type – ’goss’ – односторонняя выборка на основе градиента 'rf', Случайный лес, ’gbdt’ - традиционное дерево решений градиентного повышения;
* max\_depth – максимальная глубина дерева для базовых учащихся;
* n\_estimators - количество увеличиваемых деревьев для соответствия;
* num\_leaves – максимальное количество листьев дерева для базовых учеников;
* min\_data\_in\_leaf – минимальное количество данных на одном листе.

1. RandomForest – случайный лес

* Criterion – ‘gini’[[7]](#footnote-7) – метрика качества, которая часто используется при оценке предсказательных моделей в задачах бинарной классификации в условиях сильной несбалансированности классов целевой переменной. Данный критерий наиболее часто используется для задач банковского кредитования;
* n\_estimators - количество деревьев в лесу;
* max\_depth – максимальная глубина дерева;
* max\_features – количество функций, которые следует учитывать при поиске лучшего сплита;
* min\_samples\_leaf – минимальное количество выборок, которое требуется для конечного узла;
* min\_samples\_split – минимальное количество выборок, необходимое для разделения внутреннего узла;
* bootstrap – используются ли образцы начальной загрузки при построении деревьев.

1. SVC – классификация C-опорных векторов

* kernel – ядро;
* gamma – коэффициент ядра;
* C – параметр регуляризации.

1. KNeighbors – к-ближайших соседей

* n\_neighbors – количество соседей;
* weights – весовая функция, используемая в прогнозе. единый вес. ’uniform’ - все точки в каждом районе имеют одинаковый вес.

1. LogisticRegression – логистическая регрессия

* С – инверсия силы регуляризации;
* penalty – штраф за ошибку модели;
* solver – алгоритм для использования в задаче оптимизации, ‘saga' -быстрее для больших наборов данных.

После прохождения этапа обучения модели показали следующий результат:

1. путь 1: roc\_auc=0.86
2. путь 2: roc\_auc=0.73

|  |  |
| --- | --- |
| model | roc\_auc |
| LogisticRegression | 0.60 |
| LightGBM | 0.76 |
| StackingClassifier | 0.74 |
| StackingClassifier1 | 0.74 |
| StackingClassifier2 | 0.74 |

Из всех исследуемых алгоритмах наиболее оптимальным можно считать 4ий, поскольку он принес самый высокий результат на целевую метрику качества

Данную модель можно использовать в банке Home Credit, именно на их данных удалось провести исследования, также модель можно применять в любых других организациях, которые предоставляют услуги выдачи ссуд.

В исследовании было применено несколько методов выбора функций. Эти методы необходимы для уменьшения количества функций, повышения интерпретируемости модели, уменьшения времени выполнения модели и повышения производительности обобщения на тестовом наборе. Для выбора характеристик были использованы такие методы, как: удаление всех элементов с нулевой важностью, как это определено моделью для повышения градиента, были оставлены только функции с 90% важностью в модели для повышения градиента, сгенерированы новые полиномиальные признаки, а также усредненные признаки по метрикам, а также проработан механизм подбора гиперпараметров.

Для улучшения модели можно продолжить генерировать полезные функции, объем выборки на данном этапе позволяет это сделать, например, в данных есть такие признаки как отметка документов, данных признаков в наборе порядка 22, можно сгенерировать новый признак, который бы имел значения в зависимости от всех 22 документов. Процесс разработки функций - выбор функции является итеративным, и может потребоваться еще несколько проходов, прежде чем достигнуть хорошего результата.

Также имеет смысл запустить мой разработанный алгоритм повысив порог важности, т. е. оставить признаки, которые имели порог важности хотя бы в 5% или вовсе оставить все имеющиеся признаки.

Алгоритм нейронных сетей TabNet. TabNet — нейросеть из полносвязных слоев с последовательным механизмом внимания, которая:

* имитирует ансамбль, привлекая более точные измерения и больше шагов улучшения решения
* использует разреженный выбор объектов по экземплярам, полученный на основе обучающего набора данных;
* создает последовательную многоступенчатую архитектуру, в которой каждый шаг принятия решения может внести свой вклад в ту часть решения, которая основана на выбранных функциях;
* улучшает способность к обучению путем нелинейных преобразований выбранных функций.

В исследовании было применено несколько методов выбора функций. Эти методы необходимы для уменьшения количества функций, повышения интерпретируемости модели, уменьшения времени выполнения модели и повышения производительности обобщения на тестовом наборе. Для выбора характеристик были использованы такие методы, как: удаление всех элементов с нулевой важностью, как это определено моделью для повышения градиента, были оставлены только функции с 90% важностью в модели для повышения градиента, сгенерированы новые полиномиальные признаки, а также усредненные признаки по метрикам и проработан алгоритм подбора гиперпараметров.

Достигли результата AUC ROC, равным 0,86.

Список ссылок и использованных источников

1. [https://ru.wikipedia.org](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%83%D0%B4%D0%B0#%D0%94%D0%BE%D0%B3%D0%BE%D0%B2%D0%BE%D1%80_%D1%81%D1%81%D1%83%D0%B4%D1%8B_%D0%B2_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8)
2. <https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics>
3. <https://dyakonov.org/2017/07/28/auc-roc>
4. <https://www.kaggle.com/willkoehrsen/start-here-a-gentle-introduction>
5. <https://www.kaggle.com/codename007/home-credit-complete-eda-feature-importance>
6. https://www.kaggle.com/ogrellier/feature-selection-with-null-importances
7. <https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html>
8. https://www.machinelearningmastery.ru/hyperparameters-optimization-526348bb8e2d/
9. <https://github.com/optuna/optuna>
10. <https://medium.com/optuna/introduction-to-cma-es-sampler-ee68194c8f88>
11. <https://www.sciencedirect.com/topics/mathematics/grid-search>
12. <https://zen.yandex.ru/media/id/5ee6f73b7cadb75a66e4c7e3/samyi-prodvinutyi-instrument-dlia-podbora-giperparametrov-modeli-60d2cd6168a7aa40c2b14ac8>
13. <https://neerc.ifmo.ru/wiki/index.php?title=%D0%9D%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0_%D0%B3%D0%B8%D0%BF%D0%B5%D1%80%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%BE%D0%B2>
14. <https://deru.abcdef.wiki/wiki/Gini-Koeffizient>
15. <https://zen.yandex.ru/media/id/5fd12882382a85570c79c48c/cross-validation-v-mashinnom-obuchenii-prostymi-slovami-6087e1cb391ac9409029ee1e>
16. <https://hyperopt.github.io/hyperopt/>
17. https://habr.com/ru/post/540736/

1. Ссу́да (безвозмездное пользование) — договор, по которому одна сторона (ссудодатель) обязуется передать или передаёт вещь в безвозмездное временное пользование другой стороне (ссудополучателю), а последняя обязуется вернуть ту же вещь в том состоянии, в каком она её получила, с учётом нормального износа или в состоянии, обусловленном договором [↑](#footnote-ref-1)
2. Рабочая характеристика приемника (ROC), или просто кривая ROC, представляет собой график, который иллюстрирует работу системы двоичного классификатора при изменении ее порога дискриминации. Он создается путем построения графика доли истинных положительных результатов из положительных (TPR = частота истинных положительных результатов) по сравнению с долей ложных положительных результатов из отрицательных (FPR = частота ложных положительных результатов) при различных настройках пороговых значений. TPR также известен как чувствительность, а FPR - это единица минус специфичность или истинно отрицательный показатель. [↑](#footnote-ref-2)
3. Метод генерации признаков [↑](#footnote-ref-3)
4. [↑](#footnote-ref-4)
5. Фреймворк - байесовских алгоритм, есть возможность прунинга пространства гиперпараметров (удаления плохих точек из рассмотрения). По умолчанию удаляет точки, в которых модель дает качество ниже медианы из уже рассмотренных. [↑](#footnote-ref-5)
6. Исчерпывающий поиск по заданным значениям параметров для оценщика [↑](#footnote-ref-6)
7. Критерий для выбора правила принятия решения, в котором дочерние узлы являются максимально «чистыми» [↑](#footnote-ref-7)