ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК
ОСНОВНАЯ ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА
«ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

КУРСОВАЯ РАБОТА

Программный проект на тему:
«Кредитный скоринг. Сравнение линейных моделей с более сложными моделями машинного обучения»

Выполнила: студентка группы БПМИ194 3 курса,
Смирнова Анна Романовна

Руководитель КР:
Преподаватель Воробьева Мария Сергеевна

Оглавление

Аннотация

Программный проект на тему: «Кредитный скоринг. Сравнение линейных моделей с более сложными моделями машинного обучения»

Выполнила: Смирнова А. Р.

Руководитель: Воробьева М. С.

В ходе данной курсовой работы был выполнен анализ существующих методов решения задачи кредитного скоринга. Главной целью являлось реализовать и протестировать существующие методы решения данной задачи, после чего метрически оценить результаты и выбрать лучшую модель. Методы, которые были изучены: логистическая регрессия, WOE преобразование, RidgeClassifier, древесные алгоритмы, SHAP.

Abstract

Program course work: « Credit Scoring. Linear Models Versus Modern Data Science Models»

Student: Smirnova A. R.

Teacher: Vorobeva M. S.

During this course work, an analysis of existing methods of solving the problem of credit scoring was carried out. The main goal was to implement and test existing methods of solving this problem, evaluate the results with different metrics, and choose the best model. Methods that have been studied: logistic regression, WOE conversion, RidgeClassifier, tree algorithms, and SHAP.

Ключевые слова

- *Кредитный скоринг* – это оценивание кредитоспособности лица, основанная на статистических методах.
- *Решающие деревья* – это семейство моделей, позволяющее восстанавливать нелинейные зависимости между признаками произвольной сложности.
- *Логистическая регрессия* – это метод построения линейного классификатора, позволяющий оценивать вероятности принадлежности объектов классам.
- *Метрика качества* – это численное значение, отражающее степень качества предсказаний определенной модели.
- *Weight of Evidence (WOE)* – это предсказательная сила каждого признака, вычисляемая как логарифм отношения доли числа клиентов с вероятным дефолтом к данным значением признака к доле клиентов с маловероятным дефолтом, и данным значением признака.
- *Information Value (IV)* – это общая предсказательная сила признака, вычисляемая как сумма по всем возможным значениям признака произведений WOE на разность долей числа объектов с положительным и отрицательным значением целевой переменной.

Введение

Задача кредитного скоринга возникает в ситуации, когда банку необходимо принять решение о выдаче или отказе по кредиту, при этом принимая во внимание множество несвязанных факторов. Одним из решений данной ситуации является субъективное заключение кредитного эксперта, однако человеческий фактор не всегда позволяет учесть все входные данные, потому что с течением времени количество факторов, влияющих на принятие решение о выдаче кредита, растет.

Банковская сфера активно расширяется, и каждой компании необходимо внедрять качественные автоматизированные системы, которые помогут эффективно обеспечивать контроль работы бизнес-процессов. Решить этот вопрос помогают системы кредитного скоринга, такие как, например, логистическая регрессия, древесные алгоритмы, или другие модели бинарной классификации, с помощью которых можно оценить больший набор признаков и учесть предыдущий опыт работы с кредиторами.

Кредитный скоринг, несмотря на свое название, применяется широко за пределами обычных кредитов. Например, аналогичный анализ проводят для выдачи кредитных карт, ипотеки и других долговых финансовых операциях. От оценки кредитоспособности заемщика так же зависит одобряемая сумма кредита, срок и процентная ставка. Кроме этого, кредитный скоринг используется не только для приятия решения о выдаче кредита, а также в процессе исполнения заемщиком своих обязательств, для того чтобы банк мог изучить поведение клиента до выплаты задолженности, и как можно раньше предвидеть возможный дефолт.

Главной **целью** этой работы является самостоятельно реализовать существующие методы решения задачи кредитного скоринга, после чего составить подробное описание каждого из методов и провести сравнительный анализ качества. **Актуальность** работы подтверждается постоянным совершенствованием банковской системы, и растущей необходимостью работать с большими объемами данных. Численность населения непременно

растет, в связи с чем увеличивается и количество предлагаемых банками услуг, в том числе различного рода кредитов, отличных от привычного понимания. Например, появлением лизинга[1] или популярности микрозаймов[2], которые так же нуждаются в анализе платежеспособности своих клиентов. Кроме расширения сферы применения, несмотря на популярность задачи, основным методом решения до сих пор является логистическая регрессия с весомыми признаками (WOE).

Основные **задачи**, которые предстоит выполнить в ходе работы:

1. Провести разведочный анализ данных
2. Осуществить обработку данных
3. Реализовать различные методы машинного обучения: в контексте моей работы это различные древесные алгоритмы и вариации логистической регрессии, которые было решено использовать для сравнения.
4. Провести сравнительный анализ полученных результатов используя подходящие метрики качества: F1, ROC AUC, Precision, GINI, Accuracy и другие.
5. Сформулировать предложения о том, можно ли внедрить лучшую модель для решения бизнес-задач и эффективной работы в банковской сфере.

Исходный код проекта можно найти по ссылке на Github репозиторий: https://github.com/AnnaSmirnova-study/CreditScoring.

[1] Лизинг — это долгосрочная аренда определенных объектов собственности (оборудование, машины, сооружения) с погашением задолженности в течение нескольких лет.

[2] Микрозаймы — это кредиты на небольшие денежные суммы, которые обычно даются на короткий период.

Обзор литературы

На данный момент существует огромное множество работ, связанных не только с классической задачей кредитного скоринга, где решение принимается не за счет автоматизированных систем, но работ, в которых сравниваются различные модели машинного обучения. Однако, большинство из них, несмотря на исследования современных библиотек алгоритмов и получения эффективных моделей, не предлагает решений по интеграции в существующие банковские системы.

Одной из исследованных мною работ является «Big Data for Credit Scoring: towards the End of Discrimination on the Credit Market? Evidence from Lending Club» 2017-2018 годов, в которой Par Orphée Van Essche исследует то, как с течением времени приходится анализировать все большее количество признаков, влияющих на выдачу кредита. Однако, эта работа в основном об исследовании анализа больших данных, и никак не реализует полученные идеи. Кроме того, большинство статей затрагивающих современные решения задачи кредитного скоринга ориентированы в основном на за рубежную систему кредитования, что затрудняет интеграцию полученных современных моделей в России.

Другим источником является «Credit scoring approaches guidelines» 2019, разработанный The World Bank Group. Компания состоит из пяти всемирных организаций, деятельность которых направлена на оказание финансовой и технической помощи. Данная работа представляет собой детальное полное описание задачи кредитного скоринга и различных подходов к ее решению, но не затрагивает современные решения, такие как, например, CatBoost.

Таким образом, несмотря на большое количество разных типов исследований, современные модели до сих пор не интегрированы в Российскую банковскую систему. Одной из причин является недостаток практических исследований, направленных на отечественную систему кредитования, и использующих последние подходы машинного обучения. Моя работа может стать началом одного из таких исследований.

1. Анализ предоставленных данных

Предварительная обработка данных необходима перед решением любой задачи машинного обучения. При сборе данных для обучения модели могут быть использованы различные источники, что может послужить наличию выбросов, пропущенных данных, расхождения формата и настоящего значения признака и другого.

Зачастую предобработка данных занимает больше всего времени, однако тщательное и подробное изучение имеющегося набора признаков – залог успешного решения задачи, поскольку более качественные данные помогают получить более совершенную модель. Для того чтобы принять решение, как поступать с той или иной проблемой, существует множество методов. Мы посмотрим на подробное описание каждой из них, и постепенно применим наиболее подходящие к нашему набору данных.

Обзор входных данных

В качестве входных данных использовался набор «Python_Credit Scoring»[3] из соревнования на Kaggle. В нем 26 различных признаков, не включая целевую переменную. Ниже представлены краткие описания каждого из признаков, которые мы рассмотрим подробнее позже:

1. *loan_amnt* – Запрашиваемая у банка сумма для кредита.
2. *funded_amnt* — Общая сумма обязательств по кредиту на данный момент.
3. *funded_amnt_inv* — Общая сумма обязательств инвесторов.
4. *term* — Количество платежей по запрошенной сумме.
5. *int_rate* — Процентная ставка кредита.
6. *installment* — Размер первоначального взноса.
7. *grade* — Оценка кредитного риска.

[3] Ссылка на источник с данными: https://www.kaggle.com/code/aashofteh/python-credit-scoring-ml-elastic-net-regression/data

8. *emp_title* — Должность, представленная заемщиком при получении кредита.
9. *emp_length* — Трудовой опыт в годах.
10. *home_ownership* — Статус собственности жилья на момент открытия кредита.
11. *annual_inc* — Годовой доход.
12. *verification_status* — Статус верификации.
13. *issue_d* — Месяц, в который получено финансирование.
14. *purpose* — Цель взятия кредита.
15. *addr_state* — Государство, указанное в заявке на получение кредита.
16. *dti* — Процент ежемесячного валового дохода потребителя, который идет на выплату долгов.
17. *delinq_2yrs* — Количество просроченных платежей более чем на 30 дней за последние два года.
18. *earliest_cr_line* — Месяц открытия самой ранней кредитной истории.
19. *inq_last_6mths* — Количество обращений кредитора в бюро кредитных историй за последние 6 месяцев.
20. *open_acc* — Это количество открытых кредитов в данный момент.
21. *revol_bal* — Общий кредитный оборотный остаток.
22. *revol_util* — Доля утилизации кредита
23. *total_acc* — Общее количество активных и закрытых кредитов.
24. *out_prncp* — Оставшаяся непогашенная сумма.
25. *total_pymnt* — Выплаты, полученные на сегодняшний день.
26. *loan_status* — Текущий статус кредита.
27. *risk* — Дефолт (1) или не дефолт (0).

Создание новых признаков

Иногда отдельный признак на первый взгляд может не нести полезной информации, однако, как мы обсудим позже, может быть очень полезен в совокупности с другими, или при создании новых признаков. В нашем наборе данных нет очевидных комбинаций для построения признаков, которые могли бы положительно влиять, но, чтобы попробовать этот метод, я добавила колонку «mean_payment», которая считает среднемесячный платеж по кредиту, без учета процентов.

Разведочный анализ данных

Exploratory Data Analysis (EDA) – это важнейшая ступень подготовки к построению модели. Этот этап обычно занимает больше всего времени, потому что мы вручную исследуем каждый признак на предмет значимости для решения нашей задачи.

Для рассмотрения категориальных признаков основной задачей является посмотреть на уникальные значения и их интерпретируемость. Например, в изначальном наборе данных у нас есть переменная *loan_status,* которая принимает такие значения: Fully Paid, Charged Off, Default, In Grace Period, Late (16–30 days), Late (31–120 days). Если мы знаем каждое из значений, то можем более точно оценить, какие категориальные переменные подойдут нам лучше. Здесь нам важнее оставить только два из них – Fully Paid, Charged Off, поскольку нас интересует только два сценария.

Для числовых признаков ситуация обстоит иначе: нам часто сложно интерпретировать значения, поэтому важнее будет посмотреть на их математические характеристики. Например, на график распределения, который поможет применить правильные модели отбора переменных в дальнейшем, на «ящик с усами» для проверки наличия выбросов, наличие отрицательных значений, пропусков.

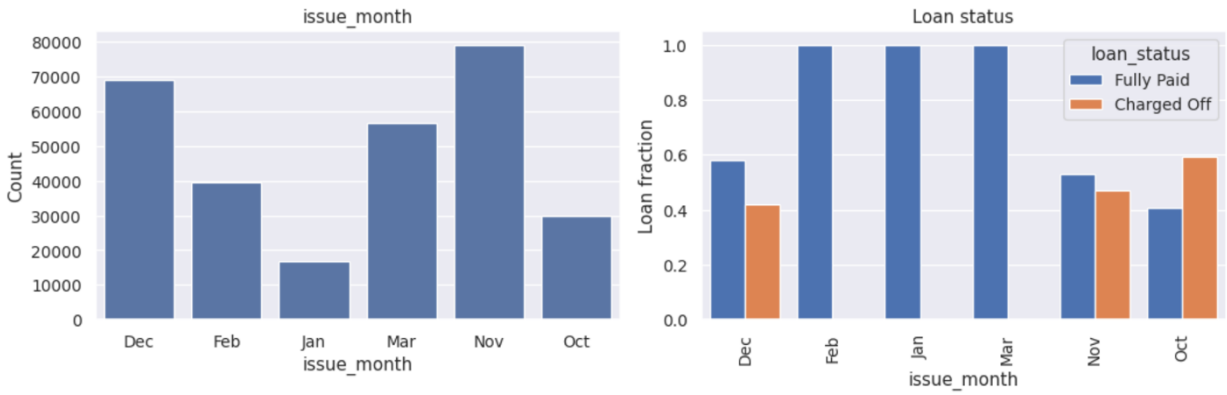


Рисунок 1.1 Необычная зависимость месяца, в который было получено финансирование, и погашения кредита

Обработка данных

Перед тем как переходить к точечным методам, стоит провести общую обработку данных. Она будет включать в себя следующие шаги:

Поскольку в моей задаче достаточное количество таргета обоих значений, я буду удалять любые признаки с более чем 85% пропущенных значений, не боясь потерять ценные данные. Удалению также подлежат признаки, у которых только одно уникальное значение в совокупности с пропущенными данными. Таких колонок в моем дата сете обнаружено не было.

Удалить признаки, значения которых повторяют друг друга, чтобы избежать мультиколлинеарности[4]. В нашем наборе признаков, точно повторяющих другие нет, но есть те, которые включают в себя значения другого признака. Например: *funded_amnt* и *funded_amnt_inv*, распределение которых достаточно близки. В такой ситуации имеет смысл оставить только один признак, для нас это *funded_amnt_inv*, так как он имеет бóльший объем данных. Можно это увидеть так же на рисунке 1.4, где коэффициент Пирсона между этими признаками равен единице.

[4] Мультиколлинеарность — случай, при котором наблюдается высокая корреляция между признаками. Для проверки на мультиколлинеарность можно посмотреть на коэффициент Пирсона между ними.

2. Отбор важных переменных

После удаления основных пропусков и повторений, стоит рассмотреть каждый признак в отдельности. Данный подход не всегда является оптимальным, поскольку признаков может быть слишком много, но в нашем случае это поможет построить более качественную модель. При рассмотрении каждого признака можно обратиться к нескольким методам.

При рассмотрении каждого признака в отдельности, могут возникнуть сложности при решении, будет нам полезна данная характеристика или нет. В спорных ситуациях важно помнить, что даже если отдельный признак кажется слабо коррелирующим с целевой переменной, в совокупности с другим он может показывать хорошую разделяющую способность. По этой причине, при отборе переменных мы будем рассматривать одномерные и многомерные методы.

Одномерные методы

Одномерными называются методы, отражающие зависимость между конкретным признаком и целевой переменной.

Самый популярный пример данного типа – корреляция. Корреляция – это показатель степени зависимости между двумя переменными. Интервал значения корреляции может быть от – 1 до 1. Положительная корреляция говорит о том, что при увеличении значения одной переменной будет увеличиваться и другая, и наоборот. При работе с данными чаще всего используют корреляции *Спирмена* и *Пирсона*.

$$r_s = 1 - \frac{6\sum d^2}{n\left(n^2 - 1\right)}$$

Рисунок 2.1 Корреляция Спирмена

Чаще всего при построении матрицы корреляции используется корреляция Спирмена. Здесь N – количество элементов в выборке, а значения

коэффициент принимает от [-1, 1], в которых 1 – самая сильная прямая линейная зависимость.

Есть и другие, менее известные коэффициенты. При выборе коэффициента стоит учитывать, к какому типу принадлежит исследуемая переменная. Например, при нормальном распределении переменной стоит смотреть на значение коэффициента Пирсона, поскольку она подходит для непрерывных метрических переменных, а в случае ненормального распределения – Спирмена, как мере, которая работает с признаками, измеренными в ранговой шкале.

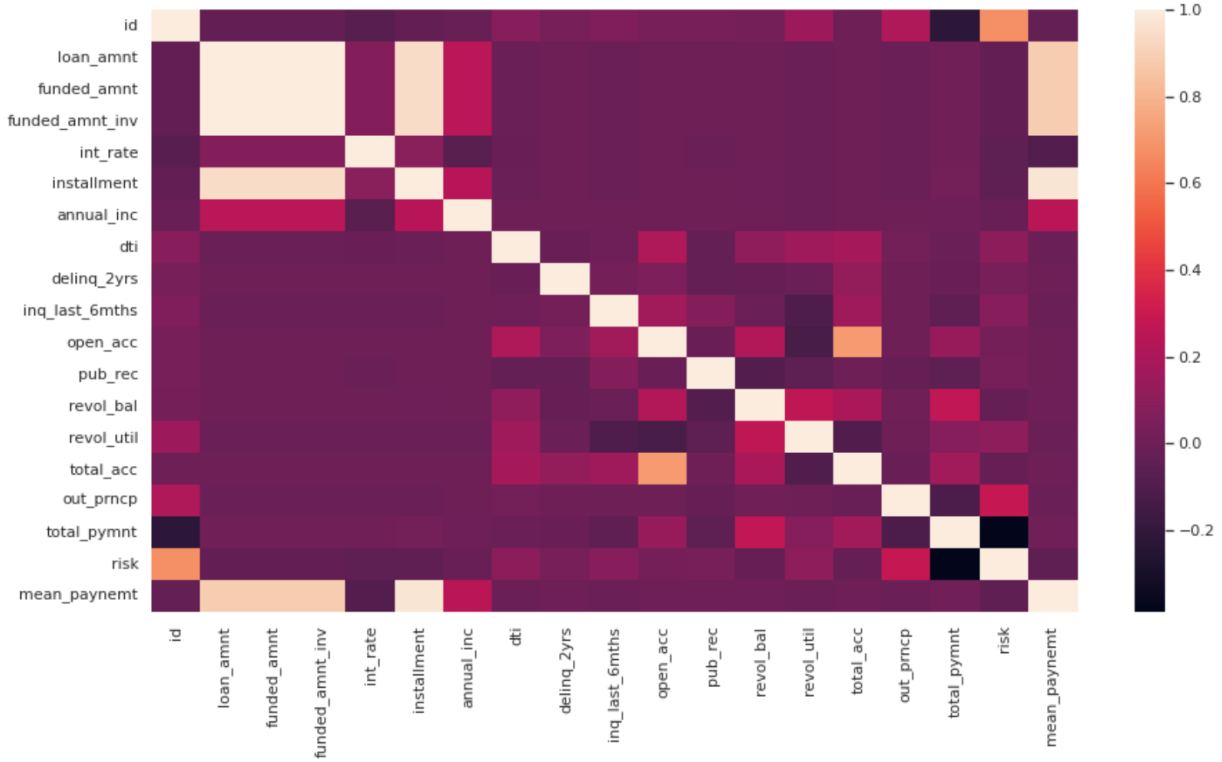


Рисунок 2.2 Матрица корреляции Пирсона.

Другой известной одномерной характеристикой является *Хи-квадрат* – статистический критерий, который используется для анализа связи между двумя категориальными переменными. Данный метод используется с самого зарождения задачи кредитного скоринга. В 1941 г. в «National Bureau of Economic Research»[5] Дэвид Дюран опубликовал свое исследование, в котором

[5] Исследование «Risk Elements in Consumer Instalment Financing»

использовал более семи тысяч «хороших» и «плохих» кредитных историй. Тогда хи-квадрат был использован для поиска отличительных черт, которые наиболее остро делили два типа кредитных историй, после чего Дэвидом был разработан индекс эффективности, предназначенный для демонстрации того, насколько эффективна данная характеристика для дифференциации степени риска среди заявителей на кредиты.

$$X^2 = \sum \frac{(\text{Observed value} - \text{Expected value})^2}{\text{Expected value}}$$

Рисунок 2.3 Формула критерия Хи-квадрат

Для примера можем принять за нулевую гипотезу, что переменные «Оценка кредитного риска» и «Дефолт» не зависят друг от друга. Для этого реализовываем тест Chi-Square и получаем P-значение равным $0.03267 < 0.05$, что говорит о необходимости отвергнуть нулевую гипотезу и принять, что эти признаки друг от друга зависят.

Многомерные методы

Если с одномерными переменными мы имеем четкий и структурированный набор методов, то в случае оценки эффективности нескольких переменных мы чаще пользуемся метриками качества, результаты которых сравниваем при разных наборах переменных.

Создание новых признаков может значительно улучшить качество работы модели, ровно как и наоборот. Ситуаций, в которых мы считаем полученную комбинацию признаков или новый созданный признак удачным, может быть несколько, как например:

- Повышение точности предсказаний. Измеряется с помощью различных метрик, о которых подробнее поговорим позже.
- Более легкая интерпретируемость результатов.

Стоит так же уточнить, что в некоторых ситуациях признаки, не содержащие значительной информации, могут быть полезны для создания

новых, более качественных признаков. Это особенно стоит брать во внимание, когда общее количество переменных небольшое и «выбрасывать» из рассмотрения целый столбец может быть неэффективно.

Посмотрим на несколько примеров многомерных методов. Один из самых наглядных – построить *диаграмму рассеяния*. С ее помощью можно наглядно отследить, наблюдается ли корреляция между двумя независимыми признаками, если мы заранее можем предположить сильную связь между ними.

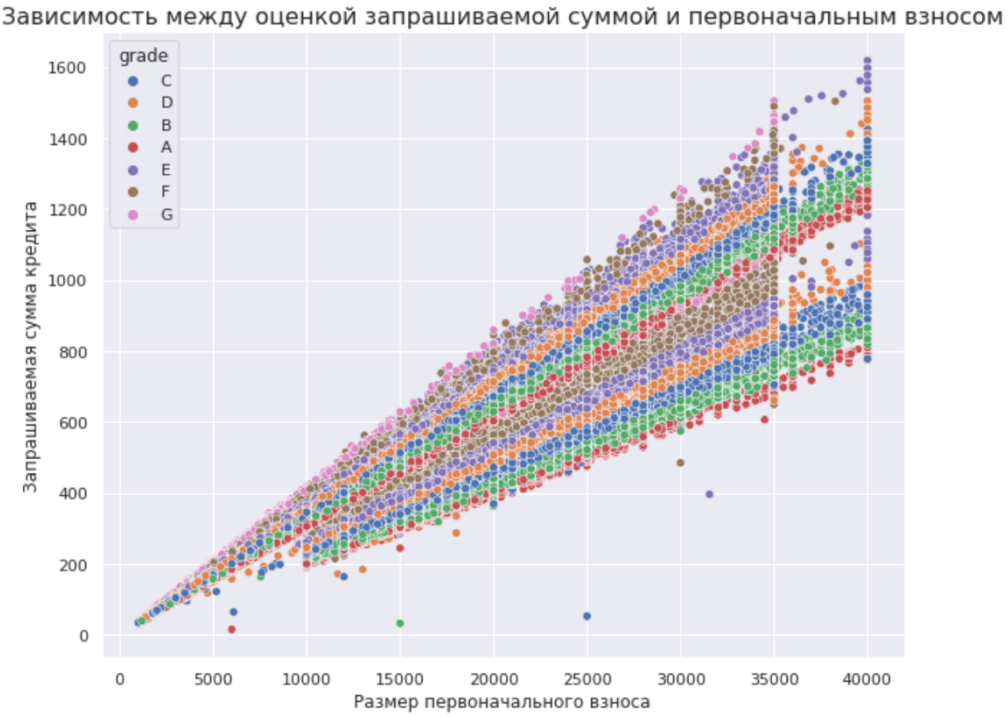


Рис 2.4 Диаграмма рассеяния первоначального взноса и запрашиваемой суммы кредита для разных оценок кредитного риска (A – наивысший уровень кредитоспособности).

Преобразование WOE и Information value

При решении задачи бинарной классификации мы часто имеем дело со множеством категориальных признаков, оценить вклад которых бывает сложно. Для решения этой проблемы разберем концепцию, которая широко применяется при решении задачи кредитного скоринга: «Weight of evidence» и «Information value».

$$WOE = \ln\left(\frac{\text{Event\%}}{\text{Non Event\%}}\right)$$

Рисунок 2.5 Вычисление WOE где Even – дефолт, Non Event – его отсутствие

Вычисление полезности какого-либо признака через WOE преобразование очень близко к вычислению обычной байесовской вероятности, мы смотрим логарифм от соотношения процента «хороших клиентов» к рассматриваемой характеристикой к количеству «плохих». Таким образом, каждое значение признака заменяется размером его «предсказательной силы». После вычисления значения *weight of evidence* у каждого признака, можно попробовать объединить столбцы с близким значением WOE, сильно коррелирующие между собой. Это связано с тем, что эти признаки могут попросту одинаково влиять на целевую переменную, а значит нет необходимости оставлять их обе сепарировано, чтобы не вызвать переобучения.

Некоторые плюсы использования WOE:

- Самостоятельно обрабатывает пропущенные значения
- Работает с выбросами
- Итоговое значение это логарифм распределения, что согласуется со значением логистической регрессии

$$IV = \sum(\text{Event\%} - \text{Non Event\%}) * \ln\left(\frac{\text{Event\%}}{\text{Non Event\%}}\right)$$

Рисунок 2.6 Вычисление IV, где Even – дефолт, Non Event – его отсутствие

Information value – это один из наиболее простых способов оценки значимости признака. После выполнения WOE преобразования можно вычислить «коэффициент полезности», который будет отражать разделительную способность признака, после чего принять решение полезен

ли признак в нашей ситуации. Можно заметить, что в формуле IV второй множитель – это как раз вычисленное ранее значение WOE.

Information Value	Variable Predictiveness
Less than 0.02	Not useful for prediction
0.02 to 0.1	Weak predictive Power
0.1 to 0.3	Medium predictive Power
0.3 to 0.5	Strong predictive Power
>0.5	Suspicious Predictive Power

Рисунок 2.7 Согласно «Credit Risk Scorecards»[6], оценка IV
в задаче кредитного скоринга может трактоваться следующими значениями.

Для вычисления обоих характеристик, важно чтобы значение признака было конечным, что не всегда выполняется для численных признаков. В этом используют следующий прием: численный признак разбивается на так называемые «бины», после чего они объединяются по 5–8 штук, а каждое значение внутри заменяется на соответствующий интервал.

Посмотрим на значения, которые получились после WOE преобразования категориальных и численных признаков на 10 бинах, при этом данные предварительно были вручную обработаны:

[6] Книга «Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring», автор: Naeem Siddiqi

	categ_feature	IV
0	term	0.003421
1	grade	0.026464
2	emp_length	0.012042
3	home_ownership	0.000192
4	verification_status	0.113406
5	purpose	0.020983
6	addr_state	0.024436
7	earliest_cr_line	0.022572
8	issue_month	0.469447

	feature	IV
0	loan_amnt	0.008603
1	funded_amnt	0.008603
2	int_rate	0.398808
3	installment	0.017854
4	annual_inc	0.006154
5	dti	0.095786
6	open_acc	0.007700
7	revol_bal	0.010455
8	revol_util	0.078568
9	total_acc	0.001475
10	mean_payment	0.021894
11	earliest_cr_year	0.002613

Рисунок 2.8 Значения Information values на обработанных данных

Значения получились совсем низкие, при этом с предсказательной способностью осталось слишком мало признаков. Попробуем посмотреть на совсем «сырых» данных:

	categ_feature	IV
0	term	0.003234
1	grade	0.027093
2	emp_length	0.011360
3	home_ownership	0.000227
4	verification_status	0.111073
5	issue_d	0.000000
6	purpose	0.016257
7	addr_state	0.022347
8	earliest_cr_line	0.018013

	num_feature	IV
0	id	0.000025
1	loan_amnt	0.015740
2	funded_amnt	0.015740
3	funded_amnt_inv	0.015744
4	int_rate	0.161944
5	installment	0.013286
6	annual_inc	0.005194
7	dti	0.075124
8	delinq_2yrs	0.004326
9	inq_last_6mths	0.022072
10	open_acc	0.003928
11	pub_rec	0.002155
12	revol_bal	0.004939
13	revol_util	0.056900
14	total_acc	0.002927
15	out_prncp	0.000000
16	total_pymnt	1.034497
17	mean_payment	0.012848

Рисунок 2.9 Значения Information values на необработанных данных

Посмотрим, как оба преобразования данных повлияют на качество моделей.

3. Построение моделей

Логистическая регрессия – это статистическая модель, используемая для решения задачи бинарной классификации, в которой вместо бинарного значения целевой переменной, предсказываются вероятности попадания в положительный класс.

$$\langle w, x_i \rangle = \log\left(\frac{p}{1-p}\right)$$

$$e^{\langle w, x_i \rangle} = \frac{p}{1-p}$$

$$p = \frac{1}{1 + e^{-\langle w, x_i \rangle}}$$

Где соответственно x - вектор со значениями признаков у объекта, а w - вектор весов модели. Обучение модели происходит за счет подбора оптимальных весов w.

Логистическая регрессия на WOE

Наиболее часто применяемым решением в банках является использование логистической регрессии на WOE переменных. Для отбора признаков после преобразования использовались следующие действия:

- Значение IV для признака ≥ 0.015
- Для всех признаков с WOE была посчитала матрица корреляции Спирмина. Для всех пар признаков, сильно коррелирующих между собой (значение корреляции выше 0.8), был оставлен только один с большим значением IV.

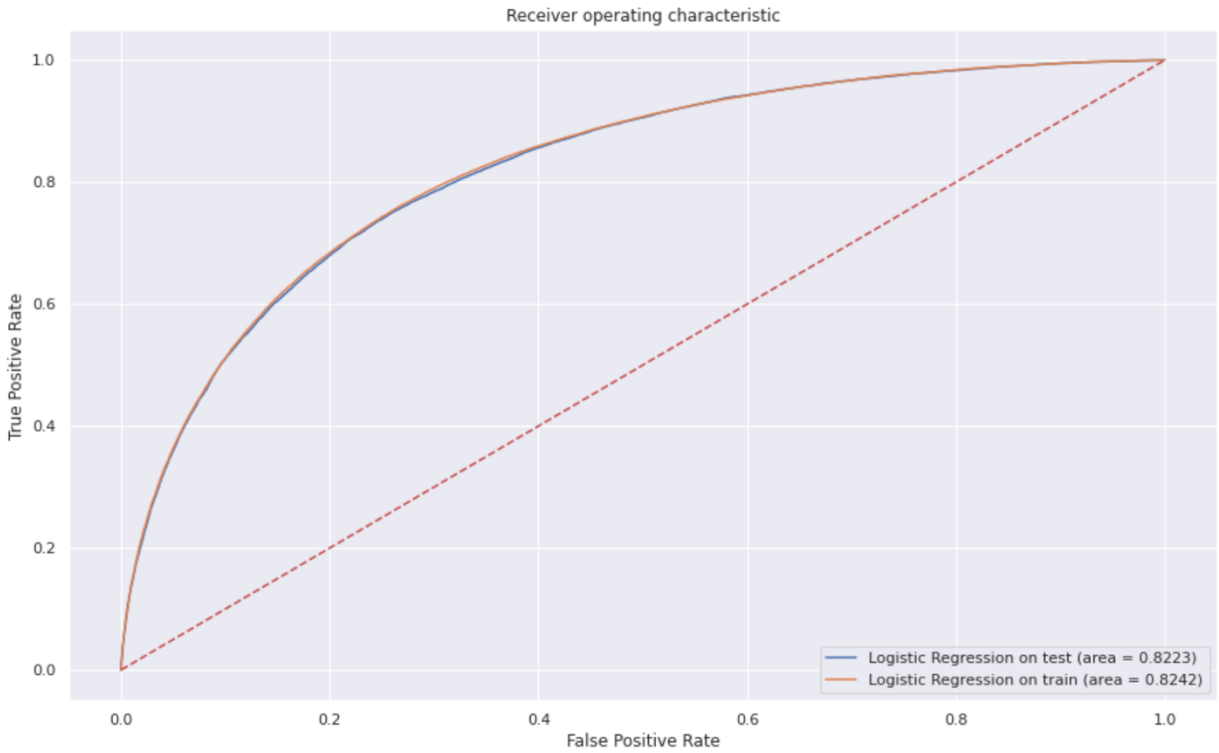


Рисунок 3.1 ROC-AUC кривая для логистической регрессии

После обучения модели можно попробовать улучшить ее качество за счет подбора параметра регуляризации. Для этого я взяла 7 различных значений, и обучив модель с каждым, выбрала наилучшее для обучения на тестовой выборке. Существенного улучшения это не дало: модель стала точнее на 0.01. Все эксперименты были зафиксированы с помощью MLflow и отражены в следующей таблице:

		Metrics		Parameters			
☐	Run Name	GINI	ROC-AUC	Bin number	Features num	Regularization	Solver
☐	Логрег на WOE с предобработкой	0.74	0.87	10	17	1	liblinear
☐	Логрег на WOE с предобработкой	0.74	0.87	10	17	1	-
☐	Логрег на WOE с предобработкой	0.74	0.87	10	17	-	-
☐	Логрег на WOE без предобработки	0.679	0.84	15	18	0.1	newton-cg
☐	Логрег на WOE без предобработки	0.679	0.84	15	18	0.1	saga
☐	Логрег на WOE без предобработки	0.679	0.84	15	18	0.1	sag
☐	Логрег на WOE без предобработки	0.679	0.84	15	18	0.1	liblinear
☐	Логрег на WOE без предобработки	0.679	0.84	15	18	0.1	-
☐	Логрег на WOE без предобработки	0.675	0.838	20	15	0.1	-
☐	Логрег на WOE без предобработки	0.675	0.838	20	15	-	-
☐	Логрег на WOE без предобработки	0.673	0.837	20	18	-	-

Рисунок 3.2 Промежуточные эксперименты логистической регрессии с WOE преобразованием признаков

Постепенно посмотрим на распространенные подходы к измерению качества моделей, и поговорим подробнее о том, почему были выбраны именно эти метрики для определения качества.

Глобально метрики качества бинарных классификаторов делятся на «пороговые» (single-threshold) и «не зависящие от порога» (threshold-free). Чтобы разделить значения на две категории, необходимо найти «порог», по которому это разделение будет проведено – именно это отличает два этих типа друг от друга. Не зависящие от порога метрики проводят разделение разными способами, которые мы рассмотрим ниже. Считается, что в задаче кредитного скоринга исторически наиболее часто используются именно threshold-free метрики.

Наиболее очевидной метрикой при обучении в задаче классификации является доля правильных ответов – *accuracy*. Она является пороговой, и сложно интерпретирует полезность полученной модели (особенно на несбалансированных наборах данных), потому что процент положительных ответов не всегда содержит в себе какую-то полезную информацию. Наш набор данных содержит уникальную сбалансированность: практически 50% дефолтных состояний, поэтому ориентироваться на эту метрику не стоит.

Более информативными являются две следующих метрики: *точность* (precision) и *полнота* (recall). Для того чтобы их рассчитать, нам необходимо обратиться к матрице ошибок. Матрица ошибок – это способ разделения всех результатов на четыре категории, в зависимости от предсказания модели и реального ответа: True Positive, False Positive, True Negative, False Negative. Из этих данных точность и полнота высчитываются следующим образом:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}; \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Полнота отражает долю верно выделенных классификатором положительных объектов. Точность отражает долю положительно выделенных объектов, которые действительно являются положительными.

Однако, обе метрики стоит особенно осторожно применять на несбалансированных данных. В задаче кредитного скоринга чаще используется комбинация этих метрик, которая создает новый способ оценки ошибки: *F меру*.

F-мера (или F1) является средним гармоническим точности и полноты. Эта метрика идеально подходит, если мы хотим сравнить точность предсказания используя результаты работы разных моделей. Что отличает эту метрику от трех предыдущих: она полезна даже на несбалансированных наборах данных. Однако, ее сложнее интерпретировать, чем, например, accuracy. Можно посмотреть на ее значение, но кроме этой метрики использовать и другие.

Чтобы перейти к следующей метрики, нужно ввести еще два понятия: *False Positive Rate* и *True Positive Rate*. Первая отражает долю неверно определенных дефолтных заемщиков, вторая – долю верно определенных. Обе метрики формируются из обозначенной ранее матрицы ошибок:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}; \qquad \mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

Рассмотрим координатную плоскость, где двумя координатами будут FPR и TPR. Значения координат будут соответстенно от 0 до 1. Площадь под кривой, полученной из всех комбинаций данных значений, и будет являться данной интегральной метрикой. Таким образом, через количественное значение этой метрики, можно оценить качество классификатора, зная, например что ROC-AUC случайной модели равно 1/2. Эта метрика является одной из самых популярных при решении задачи кредитного скоринга, кроме этого ее можно легко интерпретировать и она подходит даже для несбалансированных данных, поэтому в основном я буду использовать ее.

Коэффициент Джини (Gini Coefficient) — еще одна метрика, которая наиболее часто используется в задачах банковского кредитования. Она используется при оценке качества моделей при решении задач бинарной классификации в условиях сильной несбалансированности классов целевой переменной.

Существует несколько способов посчитать эту метрику, одна из формул напрямую даже линейно связанна с предыдущей метрикой ROC-AUC[7]:

$$GINI = 2 * ROC\_AUC - 1$$

В задачах кредитного скоринга коэффициент интерпретируется напрямую как качество модели: от 0 до 1, при наилучших результатах. Одним из главных недостатков этой метрики является то, что она не отражает чувствительность модели к разным уровням риска. Однако, даже несмотря на это, коэффициент Джини пока остается одной из самых широко применяемых метрик для оценки модели.

Логистическая регрессия с кодированием признаков

Кроме WOE преобразования, можно попробовать более «точечный» способ предобработки данных. Для того, чтобы вручную не анализировать каждый категориальный признак, я буду использовать различные типы кодирования. Это нужно делать из-за того, что базовые линейные модели часто не могут работать с категориальными переменными. Перед тем как перейти к построению модели, разберем несколько самых известных кодировщиков.

Label Encoder является одним из самых часто применяемых способов кодирования. Принцип очень простой: каждой категории ставится в соответствие уникальное значение. Таким образом, данный кодировщик отлично работает даже с признаками, у которых большое количество разных категорий. Однако у данного способа, есть один существенный недостаток, который стоит учитывать: мы создаем избыточные зависимости в данных. Иногда моделями признаки 1 и 4 могут восприниматься «ближе», чем 1 и 10,

[7] Доказана зависимость между этими двумя метриками в исследовании «The relationship between Gini terminology and the ROC curve», проведенным Edna Schechtman и Gideon Schechtman

хотя реальные значения категорий могут этому не соответствовать. Например, если мы работаем с оценками кредитного риска, и они закодированы как: 1 – «Inadequate», 4 – «Excellent», и 10 – «Adequate».

OneHotEncoding создает новые бинарные признаки в данных. Простыми словами, в колонке нового признака «Grade_ Excellent» будет 1 в случае, если объект принадлежит к этой категории, и 0 иначе. Существенный минус этого подхода в том, что если у нас большое количество категориальных признаков, при этом с большим количеством различных категорий внутри каждого, то полученный набор данных становится просто огромным.

Некой комбинацией двух предыдущих методов служит *Binary Encoder*. Отличает его от OneHot тот факт, что каждая категория кодируется бинарным числом, а не 1 или 0. Таким образом, вместо создания N новых признаков для N категорий, мы получаем Log(N), что существенно меньше при работе с большим количеством категориальных признаков.

			Metrics	Parameters		
	↓ Start Time	Duration	F1	Average of F1	Encoding type	Scaling type
	11 seconds ago	8.7s	0.869	weighted	OneHotEncoder	StandardScaler
	36 seconds ago	10.7s	0.589	weighted	OneHotEncoder	MinMaxScaler
	1 minute ago	8.9s	0.596	weighted	OneHotEncoder	StandardScaler
	4 minutes ago	12.0s	0.598	weighted	OneHotEncoder	StandardScaler
	6 minutes ago	10.8s	0.041	binary	OneHotEncoder	StandardScaler
	6 minutes ago	11.8s	0.434	macro	OneHotEncoder	StandardScaler
	7 minutes ago	9.5s	0.434	-	OneHotEncoder	StandardScaler

Рисунок 3.3 Промежуточные эксперименты логистической регрессии с различным наполнением Pipelines

В данном случае кодирование признаков происходило после непосредственной ручной предобработки каждого признака. Наилучшее качество модели по F-мере: 0.869, по ROC_AUC метрике: 0.838, что немного ниже качества, полученного при обучении на WOE переменных.

RidgeClassifier

Кроме классической логистической регрессии, может использоваться RidgeClassifier. Это одна из версий алгоритма Ridge Regression, но хорошо работающая с бинарной классификацией. Этот классификатор преобразует целевую переменную в {-1, 1}, а затем решает ту же задачу, что и классическая Ridge-регрессия. Кроме того, эта модель может быть намного быстрее при работе с большими данными, чем логистическая регрессия, которую мы рассмотрели выше.

Попробуем обучить эту модель аналогично логистической регрессии, а затем подобрать параметр регуляризации и различные типы решений.

		Metrics		Parameters		
☐	Run Name	F1	ROC-AUC	Encoding type	Scaling type	Solver
☐	RidgeClassifier	0.751	0.677	OneHotEncoder	StandardScaler	lbfgs
☐	RidgeClassifier	0.872	0.838	OneHotEncoder	StandardScaler	sag
☐	RidgeClassifier	0.872	0.838	OneHotEncoder	StandardScaler	lsqr
☐	RidgeClassifier	0.71	0.608	OrdinalEncoder	StandardScaler	svd
☐	RidgeClassifier	0.71	0.608	OrdinalEncoder	StandardScaler	-
☐	RidgeClassifier	0.872	0.838	OneHotEncoder	MinMaxScaler	-
☐	RidgeClassifier	0.872	0.838	OneHotEncoder	StandardScaler	-
☐	Логистическая регрессия с кодированием	0.869	0.838	OneHotEncoder	StandardScaler	-
☐	Логрег на WOE с предобработкой	-	0.87	-	-	-

Рисунок 3.4 Промежуточные эксперименты RidgeClassifier

Как видно по результатам, в сравнении с логистической регрессией, получить качество лучше чем у логистической регрессии на WOE переменных не удалось. Качество модели либо сохраняется, либо ведет себя хуже на некоторых видах параметров или типе кодирования.

Древесные алгоритмы

Линейные алгоритмы, которые мы рассмотрели выше, действительно имеют хорошее качество и умеют работать с большими данными. Однако, в задаче скоринга, кроме линейных зависимостей между параметрами (например, между размером кредита и первоначальным взносом, или между

оценкой кредитного риска и процентной ставкой) могут быть и другие, более неочевидные зависимости. Для того, чтобы повысить качество предсказаний еще больше, попробуем обучить несколько древесных алгоритмов.

Древесные алгоритмы, или решающие деревья – это новый вид моделей, который может устанавливать нелинейные зависимости между признаками произвольной сложности. Кроме этого, благодаря функциям визуализации, деревья можно будет легко интерпретировать.

DecisionTreeClassifier

Одним из самых базовых алгоритмов решения классификации является DecisionTreeClassifier. В качестве входных данных он использует два набора: массив X с обучающими выборками, и массив Y целочисленных значений целевой переменной. Посмотрим на некоторые параметры:

- *criterion* – это критерий выбора. По умолчанию используется описанный ранее коэффициент GINI.
- *splitter* – это критерий выбора точки разделения. Может быть двух типов: «случайный» или «лучший». «Случайный» выбор оптимальнее использовать, когда у нас очень большой объем данных. В нашем случае подойдет «лучший».
- *max_depth* – это максимальная глубина дерева. Параметр можно оставить по умолчанию, если набор данных небольшой, в противном случае глубину лучше ограничить во избежание переобучения.
- *max_features* – это количество признаков, учитываемых при делении. Если количество признаков небольшое, то можно оставить значение по умолчанию «None», что будет значить, что при разделении учитываются все признаки.
- *class_weight* – этот параметр помогает вручную корректировать веса признаков, в случае если при обучении случается очевидный «перекос».

В нашем случае можно попробовать оставить по умолчанию «None» или попробовать «balanced», что автоматически скорректирует веса.

			Metrics		Parameters			
	Duration	Run Name	F1	ROC-AUC	Criterion	Encoding type	Max_depth	max_features
	11.6s	DecisionTreeClassifier	0.919	0.914	-	OneHotEncoder	10	18
	6.3s	DecisionTreeClassifier	0.919	0.914	-	LabelEncoder	10	18
	5.9s	DecisionTreeClassifier	0.919	0.914	-	LabelEncoder	10	18
	6.0s	DecisionTreeClassifier	0.919	0.914	gini	LabelEncoder	10	18
	5.0s	DecisionTreeClassifier	0.815	0.798	gini	LabelEncoder	5	-
	6.1s	DecisionTreeClassifier	0.999	0.999	gini	LabelEncoder	20	-
	5.8s	DecisionTreeClassifier	0.919	0.914	-	LabelEncoder	10	-
	4.7s	DecisionTreeClassifier	0.776	0.769	-	LabelEncoder	-	-

Рисунок 3.5 Промежуточные эксперименты DecisionTreeClassifier

Нам удалось значительно превзойти качество линейной модели, кроме этого, благодаря визуализации дерева можно наглядно интерпретировать результаты (Приложение 1).

XGBoost

Посмотрим на одну из самых успешных реализаций алгоритма градиентного бустинга. В отличии от класического дерева решений, XGBoost использует алгоритмы, строящие деревья связанно и постоянно минимизируя функцию потерь, в то время как алгоритмы как DecisionTreeClassifier одно дерево с заданными параметрами. Таким образом, градиентный бустинг позволяет строить более точные модели.

Посмотрим аналогично на основные параметры модели:

- *booster* – это тип нашей модели. Она принимает два значения: «gbtree», для использования алгоритмов, основанных на деревьях, и «gblinear» для линейных моделей.
- *nthread* – это количество используемых ядер при параллельном обучении. Полезно при большом наборе данных, так как увеличивает скорость модели, но в нашем случае не необходимо.
- *eta* – это скорость обучения, делает модель более устойчивой, с каждым шагом уменьшая вес.

- *max_depth* – это максимальная глубина дерева, с аналогичными свойствами, что и в DecisionTreeClassifier.
- objective – выбор функции потерь для минимизации. Значение, которое подойдет нам лучше всего: «*binary:logistic*» – логистическая регрессия для бинарной классификации, возвращает предсказанную вероятность.
- eval_metric – метрика для проверки данных. Зависит от предыдущего параметра. Возможные значения: rmse (среднеквадратическая ошибка), mae (средняя абсолютная ошибка), auc (площадь под ROC-кривой), error (частота ошибки для бинарной классификации).

К сожалению, один из самых главный минусов использования градиентного бустинга является невозможность напрямую интерпретировать результаты модели. В главе с выбором лучшей модели мы посмотрим, как можно применить различные методы интерпретации и посмотреть на значимость отдельных признаков.

Наилучшее качество, которое получилось у этой модели по метрике качества ROC_AUC: 0.9453, что значительно превосходит предыдущие алгоритмы.

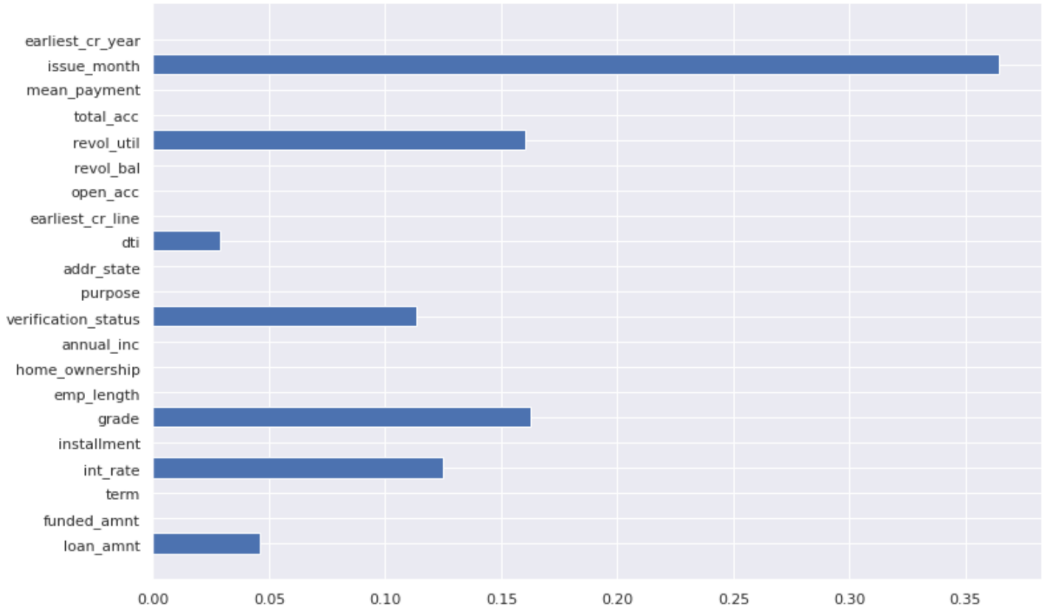


Рисунок 3.6 Индекс важности каждого признака после обучения XGBoost

CatBoost

Одним из наиболее эффективных решений является библиотека алгоритмов градиентного бустинга от Яндекса. Главное преимущество перед предыдущими – возможность работать с категориальными переменными, без необходимости предварительно их обрабатывать. CatBoost строить несколько деревьев и пытается для каждой новой повысить качество по сравнению с предыдущей.

Для нашей задачи лучше всего подойдет CatBoostClassifier.

Посмотрим на параметры для обучения, которые я буду использовать:

- *loss_function* – показатель, используемый для обучения. Нам подойдет logloss, использующийся для классификации.
- *Iterations* – максимальное количество деревьев, которое можно построить. По умолчанию 1000.
- *learning_rate*– скорость обучения, которую мы уже тестировали в предыдущих моделях.
- *l2_leaf_reg* – коэффициент для L2 функции потерь.
- *depth* – глубина дерева. Рассмотрю варианты 5 и 10.
- *cat_features* – категориальные признаки в нашем наборе данных.
- *custom_loss* – метрики, по которым будет проверяться качество модели. В нашем случае оставим AUC и добавим Accuracy.
- early_stopping_rounds —параметр, называемый детектор переобучения. В случае, если указанная метрика перестает улучшаться в течение определенного количества итераций (и метрику и итерации мы указываем), то обучение останавливается.

С данными в этом случае интереснее всего будет поработать совсем без предварительной обработки и посмотреть на результат, а затем попробовать ручную предобработку или WOE преобразование, и сравнить качество.

Во время обучения выяснилось, что переменная issue_d, которая хранит значение месяца, в которое получено финансирование, напрямую делит

дефолтные и нет значения. Во время обучения предыдущих моделей выявить эту зависимость не удалось. Итоговое качество на данных без предварительной обработки и с удалением issue_d составило: ROC AUC 0.95

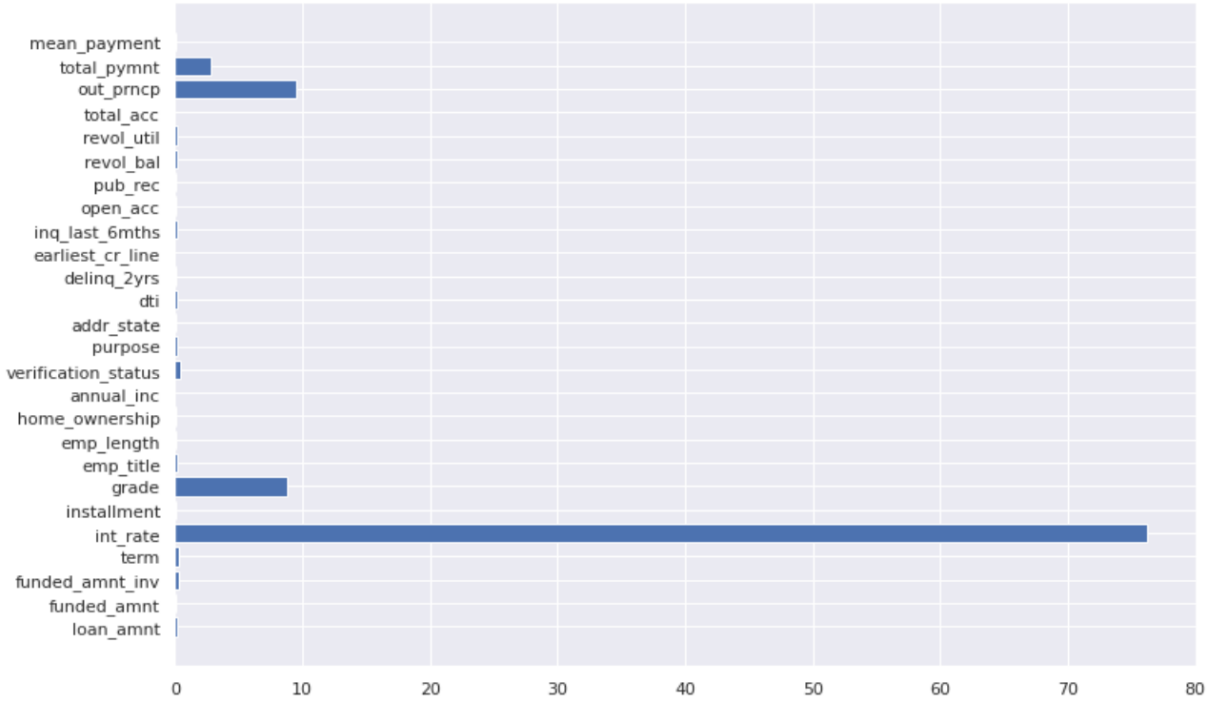


Рисунок 3.7 Индекс важности каждого признака после обучения CatBoost

Выбор лучшей модели

Посмотрим на качество, полученное на каждой модели на тестовой выборке с метрикой ROC-AUC:

Duration	Run Name	ROC-AUC
3.2min	CatBoostClassifier	0.95
4.7s	XGBClassifier	0.945
3.8s	DecisionTreeClassifier	0.914
8.0s	RidgeClassifier	0.838
4.2s	Логрегрессия на WOE	0.821

С небольшим отрывом из всех наших моделей побеждает CatBoost, а самой слабой моделью оказалась Логистическая регрессия с WOE преобразованием. Но почему тогда ее используют чаще всего? Как мы уже упоминали ранее, много зависит от трактуемости модели.

Большинство бустинговый алгоритмов похожи на черный ящик: несмотря на отличное качество работы, результаты довольно сложно интерпретировать. Для таких моделей существуют различные методы, и один из самых известных является SHAP (Приложение 2).

SAHP (Shapley additive explanations) – это инструмент, который позволяет визуализировать вклад каждого признака в качество модели. Работает это следующим образом: вычисляется качество модели на обучении с и без определенного признака, а затем вычисляется значение влияния:

$$\phi_i(p) = \sum_{S \subseteq N/\{i\}} \frac{|S|!(n-|S|-1)!}{n!} (p(S \cup \{i\}) - p(S))$$

Здесь p(S ∪ {i}) – это предсказание модели с признаком (где i – признак), p(S) - предсказание без i-го признака, n – это количество всех признаков. Посмотрим SHAP-значения для каждого признака из лучшей модели и определим, какие признак оказали наибольшее влияние:

Заключение

После построения пяти различных моделей для решения задачи кредитного скоринга, проверки разных типов кодирования признаков в качестве наилучшей модели была выбрана CatBoost, с наилучшим значением AUC-ROC метрики на тестовой выборке. Однако, стоит учитывать, что рекомендовать эту модель к внедрению по-прежнему может быть спорно: интерпретируемость признаков не так высока, как у классических методов решения, что может повысить экономический ущерб.

У данной работы есть множество перспектив развития: кроме самых известных моделей, можно проанализировать нейронные сети на табличных данных или другие древесные алгоритмы. В частности, для построения эффективной модели может потребоваться больший набор данных: с каждым годом признаков, влияющих на решение, становится все больше, и в нашем наборе не была отражена даже половина от самых используемых, что затрудняет построение жизнеспособной модели. Кроме того, для оценки эффективности полезно не только использовать метрики, но и считать экономический эффект: прибыль и потери банка, при условии выдачи кредита всем клиентам на основе предсказаний модели. В дальнейшем для лучшей модели можно разработать веб-сервис, который позволит интегрировать это решение в банковскую сферу, и возможно преумножить эффективность решения задачи кредитного скоринга, которое существует сейчас.

Список литературы

1. Machine learning algorithms can help us to estimate the risk of a financial decision. // Towards Data Science

   URL: https://towardsdatascience.com/financial-data-analysis-80ba39149126

2. Построение скоринговых карт с использованием модели логистической регрессии // «НАУКОВЕДЕНИЕ»

   URL: https://naukovedenie.ru/PDF/180EVN214.pdf

3. Machine learning algorithms can help us to estimate the risk of a financial decision // Towards Data Science

   URL: https://towardsdatascience.com/financial-data-analysis-2f86b1341e6e

4. Big Data for Credit Scoring: towards the End of Discrimination on the Credit Market? Evidence from Lending Club // Professions Financiers

   URL:https://www.professionsfinancieres.com/sites/professionsfinancieres.com/files/24-MNA-Big%20Data%20for%20Credit%20Scoring.pdf

5. The World Bank Group CREDIT SCORING APPROACHES GUIDELINES.

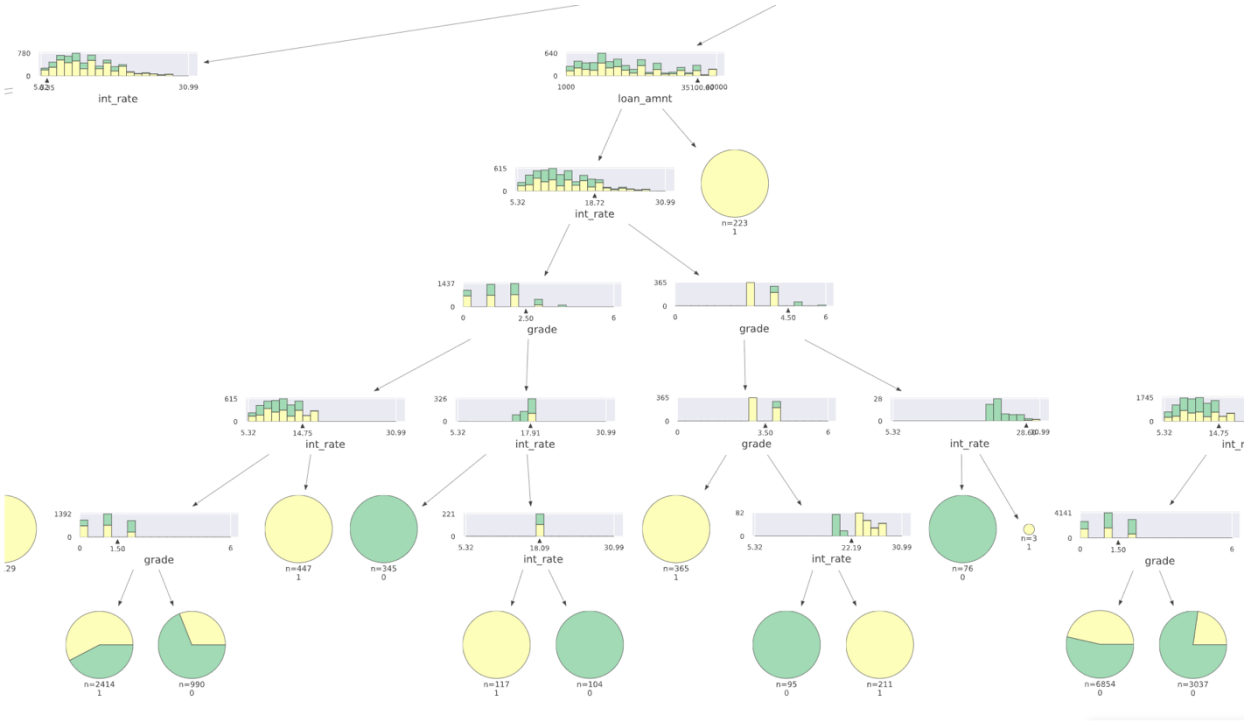
   URL:https://thedocs.worldbank.org/en/doc/935891585869698451013002202020/original/CREDITSCORINGAPPROACHESGUIDELINESFINALWEB

6. Weight of evidence and Information Value using Python // Medium

   URL:https://sundarstyles89.medium.com/weight-of-evidence-and-information-value-using-python

Приложения

Приложение 1. Визуализация построенного DecisionTreeClassifier дерева.



Приложение 2. Интерпретация признаков в CatBoost с помощью SHAP.