МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И. Сатпаева

Институт кибернетики и информационных технологий

УДК 004.942

На правах рукописи

Абдыкалыкова Асель Ермековна

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На соискание академической степени магистра

Название диссертации	Решение скоринговой проблемы с применением алгоритмов машинного обучения
Направление подготовки	6М070500 – «Математическое и компьютерное моделирование»

Научный руководитель
Профессор, д-р инж. наук
_____Р.И.Мухамедиев
«_8_» __июля_____2020 г.

Рецензент
Профессор, к.ф-м.н
_____ И.М.Уалиева
8 июля 2020 г.

Нормоконтроль
Ведущий инженер
_____ М.С.Амирбекова
8 июля 2020 г.

ДОПУЩЕН К ЗАЩИТЕ
Директор НОЦ МиК
_____Н.С.Даирбеков
8 июля 2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет
имени К.И. Сатпаева

Институт кибернетики и информационных технологий

Научно-образовательный центр математики и кибнернетики

6М070500 – Математическое и компьютерное моделирование

УТВЕРЖДАЮ
Директор НОЦ МиК
_____Н.С.Даирбеков
8 июня 2020 г.

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту *Абдыкалыковой Асель Ермековне*
Тема: *Решение скоринговой проблемы с применением алгоритмов машинного обучения*
Утверждена приказом Ректора Университета _____ г.
Срок сдачи законченной диссертации *“08”июля 2020 г.*
Исходные данные к магистерской диссертации: *Статьи с описанием методов построения стандартных скоринговых карт*
Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание диссертации:
а) Сбор и анализ источников информации по теме диссертации.
б) Анализ существующих скоринговых систем.
в) Сбор и подготовка данных для построения скоринговых карт при помощи алгоритмов машинного обучения.
г) Построение классификационных моделей, анализ и интерпретация результатов.
д) Заключение
е) Список использованной литературы
Перечень графического материала (с точным указанием обязательных чертежей): *Презентация диссертации на 40 слайдах*

Рекомендуемая основная литература:

1. Е.С.Волкова, В.Б.Гисин, В.И.Соловьев «Современные Подходы К Применению Методов Интеллектуального Анализа Данных В Задаче Кредитного Скоринга», 2017 // Электронная версия на сайте https://cyberleninka.ru/article/n/sovremennye-podhody-k-primeneniyu-metodov-intellektualnogo-analiza-dannyh-v-zadache-kreditnogo-skoringa/viewer
2. А.С.Сорокин «Построение скоринговых карт с использованием модели логистической регрессии», 2014 // Электронная версия на сайте https://cyberleninka.ru/article/n/postroenie-skoringovyh-kart-s-ispolzovaniem-modeli-logisticheskoy-regressii/viewer

ГРАФИК

Подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю**	**Примечание**
Сбор материалов для изучения.	23.12.2019	
Обзор и изучение научных статей, теории лежащей в основе применяемой методологии построения скоркарт.	27.01.2019	
Сбор данных необходимых для построения моделей машинного обучения.	30.03.2020	
Практическая работа над построением системы кредитного скоринга. Анализ полученных результатов.	27.04.2020	
Написание магистерской диссертации.	03.07.2020	

Подписи

консультантов и нормоконтролера на законченную магистерскую диссертацию с указанием относящихся к ним разделов диссертации

Наименование Разделов	**Консультанты, И.О.Ф. (уч.степень, звание)**	**Дата подписания**	**Подпись**
Сбор научных статей для изучения	Р.И.Мухамедиев, Профессор, д-р инж..наук	8.07.2020	
Сбор данных необходимых для построения моделей машинного обучения.	Р.И.Мухамедиев, Профессор, д-р инж..наук	8.07.2020	
Практическая работа и анализ полученных результатов	Р.И.Мухамедиев, Профессор, д-р инж..наук	8.07.2020	
Нормоконтроль	М.С.Амирбекова, ведущий инженер	8.07.2020	

Научный руководитель _____ Р.И.Мухамедиев

Задание принял к исполнению обучающийся _____ А.Е.Абдыкалыкова

Дата “08” июля 2020 г.

АНДАТПА

Магистрлік диссертацияның негізгі мақсаты - дәл нәтиже алу үшін машиналық оқыту алгоритмдері мен заманауи оңтайландыру әдістерін қолдану арқылы баллдық есепті шешу.

Дәстүрлі түрде, банктерде сапалы скоринг жүйесін құру міндеті логикалық регрессия, шешім ағаштары, сызықтық регрессия әдісімен шешіледі. Бұл әдістер банктік классификацияның ұқсас проблемаларында қатаң бекітілген.

Осы жұмыста скорингтік есепті келесі себептерге байланысты шешу үшін машиналық оқытудың негізгі жіктеу алгоритмдерін қарастыруды ұсынамыз:

Қазіргі уақытта банктерде өсіп келе жатқан деректерді жинау, сақтау және өңдеу мүмкіндігі бар. Машиналарды оқыту алгоритмдері жасырын заңдылықтар мен қатынастарды таба алады

Екінші себеп - шағын несиелік серіктестіктерден бастап, тез дамып келе жатқан жаңа банктермен аяқталатын көптеген несиелік ұйымдардың пайда болуы, бұл әлеуетті клиент үшін жоғары бәсекелестік тудырады. Осылайша, компанияда скоринг тез және дәл жұмыс істеуі керек.

Экономикалық жағдайдың жылдам өзгеруіне байланысты скорингтің сапасын үнемі қадағалап отыру және қажетті жаңартуларды жасау қажеттілігі туындайды, сондықтан процесті баптап, оңтайландыру қажет.

Бұл жұмыста несиелік скорингті құру процесінде қолданылатын деректерді талдаудың заманауи әдістері және әдістеменің сипаттамасы берілген. Мәселені шешу тірек вектор әдісі, кездейсоқ орман сияқты алгоритмдердің негізінде ұсынылады.

Диссертация тақырыбы бойынша диссертация 23-24 сәуірде ISMA университетінде өткен (Латвия, Рига) IT және менеджменттің 18-ші халықаралық ғылыми конференциясында жарияланды.

АННОТАЦИЯ

Основная цель данной магистерской диссертации – решение скоринговой проблемы с использованием алгоритмов машинного обучения и современных методов оптимизации для получения наиболее точного результата.

Традиционно в банках задача построения качественной скоринговой системы решается методами логистической регрессии, деревьев решений, реже линейной регрессии. Эти методы прочно обосновались в подобных банковских классификационных задачах.

В данной работе предлагается рассмотреть основные классификационные алгоритмы машинного обучения для решения скорингвой задачи по следующим причинам:

В настоящее время у банков есть возможность собирать, хранить и обрабатывать всё большее количество данных. Алгоритмы машинного обучения способны находить скрытые закономерности и связи

Вторая причина появление множества кредитных организаций, начиная с мелких микрофинансирующих компаний заканчивая новыми быстроразвивающимися банками, создает высокую конкуренцию за потенциального клиента. Тем самым скоринг в компании должен работать быстро и максимально точно.

В связи с быстрой сменой экономической ситуации возникает необходимость постоянно мониторить качество скоринга и вносить требуемые обновления, поэтому процесс должен быть настроен и оптимизирован.

В данной работе представлены современные методы анализа данных, используемые в процессе построения кредитного скоринга, и описание методологии. Предложено решение задачи на основе таких алгоритмов как метод опорных векторов, случайный лес.

По теме данной диссертации был опубликован тезис на 18ой международной научной конференции IT и менеджмента, проведенной 23-24 апреля университетом ISMA (Латвия, Рига).

ANNOTATION

The main goal of this master's thesis is to solve a scoring problem using machine learning algorithms and modern optimization methods to obtain the most accurate result.

Traditionally, the task of constructing a high-quality scoring system is solved by the methods of logistic regression, decision trees, less often linear regression. These methods are firmly established in similar banking classification problems.

In this paper, we propose to consider the basic classification algorithms of machine learning to solve the scoring problem for the following reasons:

Currently, banks have the opportunity to collect, store and process an increasing amount of data. Machine learning algorithms are able to find hidden patterns and relationships

The second reason is the emergence of many credit organizations, starting with small microfinancing companies and ending with new rapidly developing banks, which creates high competition for a potential client. Thus, scoring in the company should work quickly and accurately.

In connection with the rapid change in the economic situation, there is a need to constantly monitor the quality of scoring and make the required updates, so the process must be tuned and optimized.

In this paper, modern methods of data analysis, used in the process of building credit scoring, and a description of the methodology are presented. A solution to the problem is proposed on the basis of such algorithms as the support vector method, random forest.

On the topic of this dissertation, the thesis was published at the 18th international scientific conference of IT and management, held April 23-24 by ISMA University (Latvia, Riga).

Содержание:

ВВЕДЕНИЕ

Кредитные организации, в особенности банки, играют решающую роль в рыночной экономике. Одна из постоянных проблем кредитных организаций, требующая своевременного решения, выдача займа неблагонадежному заемщику и отказ в выдаче благонадежному. Неверно принятое решение в конечном результате приведет к убыткам и возможному банкротству. Эта проблема существует уже несколько тысячелетий, со времен появления заимствования денежных средств. Но в ходе истории эта задача не утратила актуальности и, более того, набрала еще большую значимость.

Кредитный риск включает в себя потерю основной суммы займа, процентов, которые фактически являются заработком банка. Все эти потери в конечном итоге могут привести не просто к нарушению потока денежных средств и дестабилизации финансового состояния банка, а даже к банкротству и неспособности возместить сбережения клентов.

Во избежание таких проблем или, по крайней мере, минимизации их влияния кредитные организации должны постоянно проводить мониторинг и улучшение существующих методов, изменять свои внутренние кредитные политики.

Скоринг – повсеместная оценка клиента на каждом этапе жизни, выданного ему кредита. Скоринг помогает определить кто получит кредит, особенности выдаваемого кредита, кто является мошенником, операционные действия для повышения эффективности возврата проблемных кредитов,

Кредитный скоринг является системой прогнозного моделирования, используемая для предсказания принесет ли кредит, потенциальному клиенту-заявителю, убыток компании, при этом минимизировав влияние человеческого фактора при принятии решения. Другими словами, скоринг помогает оценить кредитный риск. Эта система основана на реальных данных клиентов, поэтому она может максимально точно решать подобные задачи.

Одним из современных подходов к решению задачи скоринга являются алгоритмы машинного обучения.

Итак, настоящая работа посвящена изучению методов построения системы кредитного скоринга, а также лежащей в его основе методов интеллектуального анализа данных, с последующим выявлением самых эффективных алгоритмов.

1. ОБЩИЕ ХАРАКТЕРИСТИКИ.

Понятие «скоринг» означает присвоение балла параметрам, описывающим объект. В результате такого оценивания складывается общий балл из ранее присвоенных баллов известных параметров. Для обеспечения достоверности и высокой точности такой оценке необходимо использовать большое количество известных характеризующих параметров.

В банковской сфере скоринг является повсеместной оценкой клиента, использующей всевозможные известные и подтвержденные данные о заявителе. Поэтому банки тестируют на эффективность и пользу для дальнейшего использования внешние источники данных, которые могут дополнительно повлиять на точность оценки клиента. Чем больше параметров, тем точнее оценка.

Существуют следующие источники данных: кредитное бюро (подробная информация по всем историческим и текущим кредитам, просрочкам), пенсионный фонд (официальная заработная плата, стаж работы), внутренняя база банка (частота обращений и результат обращения отказ/одобрение, транзакции, сберегательные счета), анкетные данные (возраст, пол, семейное положение, сфера деятельности, недвижимость и т.д.). В последнее время многие компании (сотовые операторы, предоставляющие различные услуги, такие как такси, покупки в магазинах) также собирают и анализируют данные по клиенту и предоставляют их банку (с согласия клиента), тем самым пополняя базу данных по клиенту.

Скоринговый балл – непостоянная величина, он может меняться с течением времени. Поэтому при каждом обращении важно пересчитывать оценку и пересматривать решение.

Однако недостаточно просто посчитать скоринговый балл. Важно также отобрать порог, при котором клиент, набравший тот или иной балл, будет считаться благонадежным или нет. От этого решения зависит дальнейшая финансовая ситуация и риски банка.

1.1. Историческая сводка.

Самая первая скоркарта (1956) была создана на базе алгоритма, присваивающего определенный числовой балл и определяющего способность человека выплачивать кредит. Затем разработчики данного алгоритма основали компанию FICO, которая в настоящее время является крупнейшим кредитным бюро и поставщиком программного обеспечения для банковского кредитного скоринга.

Скоринговые баллы, на основе моделей компании носят название компании FICO. Компания не раскрывает четкого алгоритма и правил составления баллов, однако предоставляет информацию о том какая информация и в каком соотношении формирует данную оценку.

- 35%: кредитная история (в т. ч. история платежей по счетам);
- 30%: бремя задолженности (в т. ч. количество счетов с остатками, сумма задолженности по различным видам счетов, доля используемых средств от возможного кредита и др.);
- 15%: продолжительность кредитных историй (например, средний возраст аккаунта, возраст самого продолжительного аккаунта);
- 10%: типы использованных кредитов (например, возобновляемый кредит, ипотечное кредитование и др.);
- 10%: недавние заявки на получение кредита. [1]

Как упоминалось ранее, определение порогового значения по распределению скорингового балла среди клиентов очень важная часть анализа кредитного риска. Однако FICO предоставляет только балл, без флага хороший это клиент, по их мнению, или нет. То есть определение порога зависит только от целей банка, хочет ли он в первую очередь снизить риск невозвратных кредитов или увеличить продажи, выдавая как можно больше кредитов.

Индекс FICO и подобные основаны на кредитной истории заемщика, то есть возможность оценить заемщика с помощью этих рейтингов не охватывает людей, никогда ранее не бравших кредиты. Для последних выделяют так

называемый социодемографический рейтинг, учитывающий как следует из названия социальные, демографические и другие доступные данные.

1.2. Цель кредитного скоринга и его виды.

Способы выбора основных параметров клиента для определения платежеспособности меняются для соотвествия требованиям современных реалий и новейших технологий.

Использование скоринга решается множество задач в банке, начиная с момента подачи заявки, оценки клиента на платежеспособность, определение на мошенничество, определение последующего возможного поведения клиентов. Существуют следующие разновидности кредитного скоринга:

1) Скоринг заявителя - *Application-scoring*: определяет платежеспособность клиента в момент подачи заявки.
2) Мошеннический скоринг - *Fraud-scoring*: определяет вероятность мошенничества со стороны клиента.
3) Поведенческий скоринг - *Behavioral-scoring*: определяет возможное будущее поведение клиента в процессе после одобрения ему кредита.
4) Скоринг по работе с просроченной задолженностью - *Collection-scoring*: определяет дальнейшие действия для эффективного сбора просроченной задолженности коллекторами. [7]

Кредитный скоринг позволяет банкам снижать риски невозврата кредита, избегать мошенничеств, а также в зависимости от уровня риска определять процентную ставку по кредиту. Таким образом, скоринг широко применяется в разных сегментах кредитования. Технологии построения скоринговой модели оценки заемщика и его поведения являются практически идентичными. В данной работе рассматривается скоринг заявителя.

В основном для решения всех вышеперечисленных проблем лучше всего использовать бинарные классификаторы, поскольку они являются наиболее качественными методами машинного обучения. В банковской сфере модели

бинарной классификации могут быть применены практически во всех ключевых направлениях деятельности. Бинарная классификация используется в задачах, где объекты выборки делятся на два класса — положительные и отрицательные:

— в скоринге это модели прогнозирования вероятности дефолта заемщика (1- кредит выйдет на просрочку, 0 – нет);

— во взыскании — модели прогнозирования вероятности возврата просроченного кредита (1 – клиент вернет просроченный кредит, 0 - не вернет);

— в антифроде — модели выявления мошеннических кредитов (1- кредитная заявка окажется мошеннической, 0 – нет);

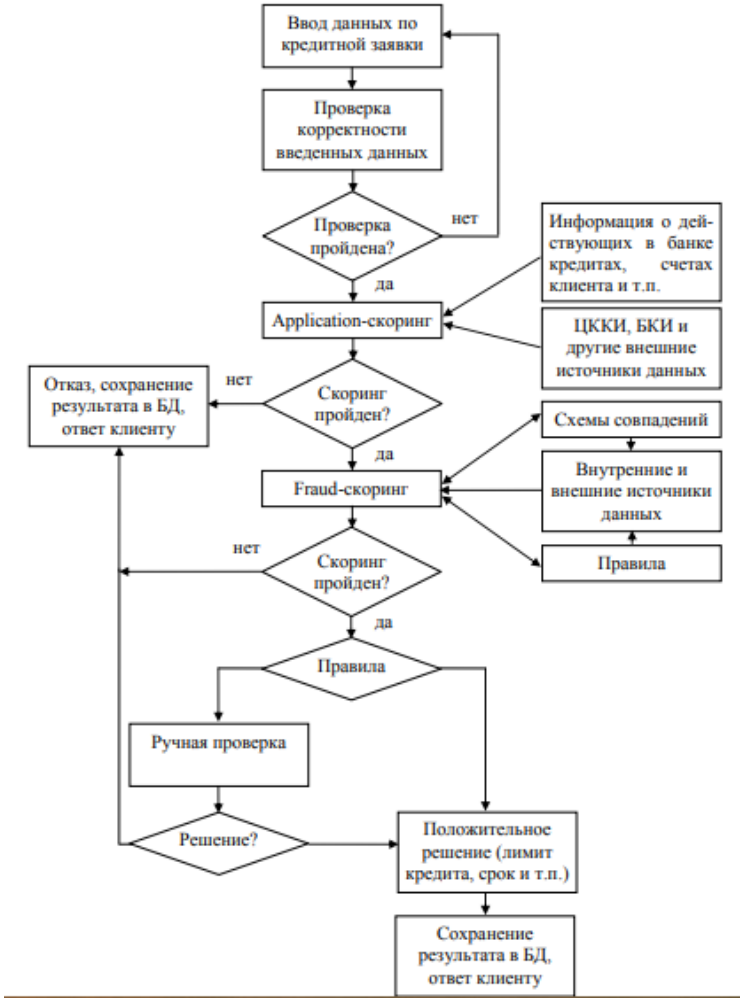


Схема 1. Процесс проверки кредитной заявки.[3]

Плюсы использования скоринговых систем в банковских процессах:

- Снижение уровня кредитного риска при отказе в кредите «плохому» клиенту.
- Обработка большого объема данных по клиенту в кратчайшие сроки.
- Исключение личностного фактора при принятии решения по одобрению кредита.
- Высокая точность.

Минусы использования скоринговых систем в банковских процессах:

- Требования к кредитной политике могут постоянно меняться в связи с изменением в экономической ситуации, что будет требовать постоянного изменения и пересмотра скоринговых систем.
- Для построения системы скоринга требуется возможность технической поддержки больших объемов данных, а также их обработки и анализа.

1.3. Актуальность исследования.

В настоящее время с ростом возможного объема собираемых данных, с появлением возможности получения одобрения на кредит без визита в банк между банками выросла конкуренция за тех самых «хороших» клиентов. С ростом конкуренции возникает необходимость в ускорении принятия решения банком, то есть процесс должен быть автоматизирован и оптимизирован. При всем этом возрастает вероятность мошенничества со стороны клиента в виде подделки документов и так далее. Поэтому новые обстоятельства требуют более точного подхода к предоставлению решения банка, а также требуют высокой скорости.

В основе прогнозной аналитики клиентов на выявление их кредитоспособности должны лежать различные алгоритмы машинного обучения, которые извлекают скрытые данные из массы данных.

Цель этого исследования сфокусирована на методах, которые используются для проверки вероятности заявителя на получение кредита, а также на методе, с помощью которого точность может быть увеличена.

Созданные скоринговые модели быстро устаревают, поэтому их необходимо регулярно проверять и обновлять. Любое изменение экономических условий требует обязательной коррекции системы расчетов. Скоринговый балл — величина непостоянная, которая меняется в зависимости от действий заемщика. Например, заемщик взял кредит — выросла кредитная нагрузка и скоринговый балл снизился. Просрочил платеж — балл упал еще ниже. Если заемщик аккуратно без просрочек выплатит кредит — балл увеличится.

1.4. Преимущество решения задачи скоринга методами машинного обучения.

Принципиальное отличие методов машинного обучения от методов статистического анализа заключается в постановке задачи, в подходе к ее решению и в принципе организации работы с данными.

Классическая задача скоринга — это задача классификации, то есть разработки модели, которая по вектору входных параметров X возвращает вектор вероятностей принадлежности объекта множеству возможных классов Y. Таким образом, задача классификации сводится к задаче поиска алгоритма a, который с вероятностной мерой переводит пространство X в вектор Y. На таком вероятностном подходе сходство между постановками задач машинного обучения и статистики заканчивается.

Различие же между подходами обусловлено методами формирования пространства факторов. В классических статистических подходах вектор правил модели четко структурирован и формируется на основе интеллектуальной работы аналитика, занимающегося построением системы. По

этой причине если смотреть на вектор наиболее значимых факторов, то он чаще всего состоит из данных кредитной истории и имеет следующий вид:

Factor_name	Feature_importance
CntOverdue	0,21
LastPaym_sameType	0,17
CurBalance	0,17
AvgLimitPaid	0,10
AvgDifferenceGoodBadAmount	0,08

Таблица 1. Пример приведения наиболее значимых факторов.[4]

Столь высокий вес достаточно простых в расчете факторов делает модели, основанные на простых подходах, легко интерпретируемыми.

В машинном обучении подход к анализу данных существенно отличается от стандартных эконометрических методов. Задача построения системы скоринга сводится к задаче поиска общих паттернов в массивах неструктурированной информации.

Эконометрика	Машинное обучение
Количество просрочек в КИ	Паттерн погашения
Статистики по суммам кредитов	Паттерн кредитов в КИ
Статистики по давности кредитов	
Статистики по срокам кредитов	
Статистики по суммам заявок	Паттерн поступления заявок
Статистики по давности заявок	
Статистики по типам заявок	
Количество посещений офисов (офлайн)/страниц сайта (онлайн)	Вектор идентификаторов посещенных офисов (или регионов)/страниц
Соответствие региональных признаков	Упорядоченный вектор регионов
Статистики по платежам	Паттерн платежей

Таблица 2. Отличия характера обрабатываемой информации.[4]

Таким образом, методы машинного обучения предполагают меньшие потери информативных признаков вследствие ручного перевода данных в

формат факторов, а также не привязаны к числовому характеру данных и могут оперировать массивами и объектами. В процессе обучения модель оперирует данными в контексте связи с другими образцами обучающего множества, поэтому при внутреннем формировании и самоорганизации факторов получаются более тонкие настройки калькуляции, вследствие чего увеличивается итоговая размерность входа и сокращается значимость каждого из полученных факторов в итоговом классификаторе.

Точность эконометрических моделей кредитного скоринга очень сильно зависит от набора факторов; эта зависимость гораздо сильнее выбранного метода построения классификатора

В случае построения системы скоринга на основе алгоритмов машинного обучения важность методов калькуляции факторов отходит на второй план; в первую очередь необходимо покрыть максимальное многообразие групп признаков в условиях ограниченности машинного ресурса. При дальнейшем покрытии максимального объема признаков необходимо поставить задачу нахождения схожих между собой подпоследовательностей и сочетания характеристик, имеющих связь с финальным значением признаков. Задача такого уровня схожа по своей сути с задачей формирования системы факторов на основе свертки известных данных.

2. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ПОСТРОЕНИЯ СИСТЕМЫ КРЕДИТНОГО СКОРНГА.

Современные скоринговые модели базируются на следующих методах статистики и исследования операций [6]:

- логистическая регрессия;
- линейная регрессия;
- деревья классификации;

Традиционными и наиболее распространенными являются линейные многофакторные регрессионные методы. Поскольку перед банком стоит задача отделить качественных заемщиков от рисковых можно использовать как методы решения задачи классификации, так и регрессии. Помимо построения кредитного скоринга, регрессионный анализ можно использовать на этапе выбора значимых переменных.

2.1. Логистическая регрессия.

В настоящее время метод логистической регрессии является наиболее часто используемым методом для построения скоринговых систем в банках. Логистическая регрессия позволяет подразделять клиентов на несколько групп риска. Все регрессионные методы чувствительны к корреляции между характеристиками, поэтому в модели не должно быть сильно коррелированных независимых переменных. Коэффициенты регрессии объясняют влияние параметров на вероятность отнесения клиента к рисковому.

Хотя логистическая регрессия и не является самым эффективным методом для классификации, она значительно менее чувствительна к размеру выборки и соотношению плохих и хороших клиентов в ней по сравнению со многими другими методами, применяемыми для классификации.

На основе полученных оценок коэффициентов логистической регрессии строится скоринговая карта, переводящая коэффициенты модели в скоринговые

баллы. Именно по набранному суммарному скоринговому баллу происходит ранжирование заемщиков и принимается решение о выдачи кредита.

$$\ln\left(\frac{p_i}{1-p_i}\right) = b_0 + b_1 x_i^{(1)} + b_2 x_i^{(2)} + \ldots + b_k x_i^{(k)},$$

Pi - вероятность наступления дефолта по кредиту для i-го заемщика;

Xj - значение j-ой независимой переменной;

b0 — независимая константа модели,

bj — параметры модели.

Уравнение, представленное выше, показывает линейную зависимость между вероятностью дефолта по кредиту и значениями независимых переменных. Независимая константа $b0$ отражает степень риска при условии, что все независимые переменные равны нулю. Значения параметра bi отражает уровень влияния переменной при этом параметре на шанс дефолта в логарифмической шкале.

Однако у такого классического метода есть минус-ограничение целевой переменной в числах 0 и 1 для обучения модели. На выходе мы можем получить вероятностную оценку принадлежности объекта к тому или иному классу, однако отсутствие возможности включения объектов с целевым классом в виде вероятностного прогноза является недостатком модели логистической регрессии.

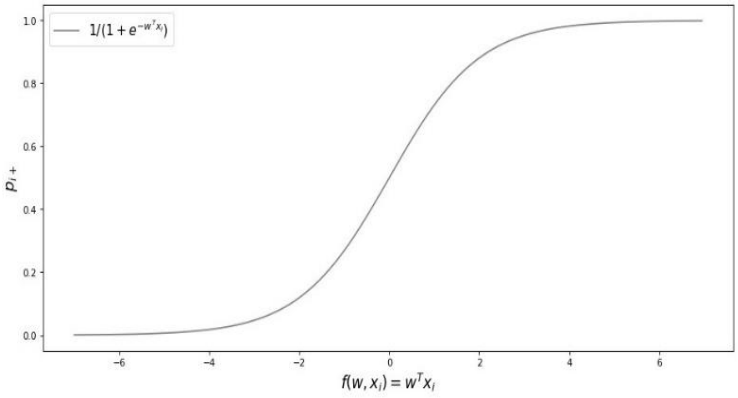


График 1. График уравнения логистической регресии.

Для интерпретации коэффициентов модели логистической регрессии обычно используют экспоненциальную форму записи модели:

$$p_i = \frac{1}{1+\exp(-(b_0 + b_1 x_i^{(1)} + b_2 x_i^{(2)} + \ldots + b_k x_i^{(k)} + \varepsilon_i))}.$$

- При включении в модель логистической регрессии непрерывных количественных переменных коэффициенты при них будут показывать, на сколько в среднем измениться логарифм шанса наступления просрочки по кредиту при изменении независимой переменной на единицу своего измерения при неизменности остальных переменных.
- В экспоненциальной форме коэффициенты будут показывать насколько в среднем изменятся шансы наступления дефолта при изменении независимой переменной на единицу своего измерения при неизменности остальных переменных. Если коэффициент регрессии будет положительный, то его экспонента будет больше единицы и шансы будут возрастать, если коэффициент окажется отрицательным — меньше, шансы будут убывать.
- При включении в модель бинарной независимой переменной, коэффициент регрессии в экспоненциальной форме при фиктивной переменной будет показывать соотношение шансов проявления дефолтов при наличии фактора, отражаемого бинарной независимой переменной, по сравнению с его отсутствием. [2]

2.2. Линейная регрессия.

Линейная регрессия связывает характеристики заемщика, представленные вектором $x \in Rn$ с целевой переменной $y \in \{-1;1\}$:

$$y = \beta_0 + \langle \beta, x \rangle + \varepsilon,$$

где ε – случайная ошибка с нулевым средним. При решении вопроса об отнесении y к тому или иному классу величина β0+⟨β,x⟩ трактуется как условное математическое ожидание E(y|x).

При использовании линейной регрессии фактически делается попытка связать вероятность дефолта p со значениями ответов на вопросы линейной функцией:

p = w0 + w1X1 + . . . + wnXn,

где левая часть представляет собой вероятность и должна изменяться от 0 до 1, тогда как правая может принимать любые значения. Для решения этой проблемы необходима некоторая функция. Логистическая регрессия заменяет вероятность дефолта на логарифм шансов дефолта:

$$\log \frac{p}{1-p} = w_0 + w_1 X_1 + \ldots + w_n X_n = s(X).$$

Формула логарифм шансов дефолта

$$s(x) = \log \frac{p(B|x)}{p(G|x)}.$$

Формула отношения вероятности дефолта.

2.3. Дерево классификации.

Наряду с упомянутыми выше методами для классификации плохих и хороших кредитных рисков используются деревья решений. Метод деревьев решений является популярным алгоритмом не только в кредитном скоринге, но и во многих других банковских процессах.

Дерево решений классифицирует клиентов на категории близкие по качеству, однако сильно различающиеся друг от друга.Дерево решений выглядит в форме дерева с разветвлениями, ветви которого являются отдельным решением.

С помощью данное метода классификации также можно сформулировать правила по принятию решения заявки на кредите, выделить самые важные переменные, которые лучше всего влияют на точность классификации.

Метод деревьев решений обладает следующими преимуществами:

- наглядностью представления результатов и возможностью интерпретации;
- автоматический поиск предикторов - алгоритм сам выберет наиболее значимые и использует их для построения дерева;
- способностью выявлять сложные нелинейные взаимосвязи
- переменные могут быть как: количественными, порядковыми, так и номинальными;

К недостаткам метода деревьев решений можно отнести:

- отсутствие общего уравнения, выражающего модель;
- проблему переобучения: слишком детализированные деревья с большим числом узлов и ветвей;
- смещение выбора в пользу переменных, у которых большее количество категорий.

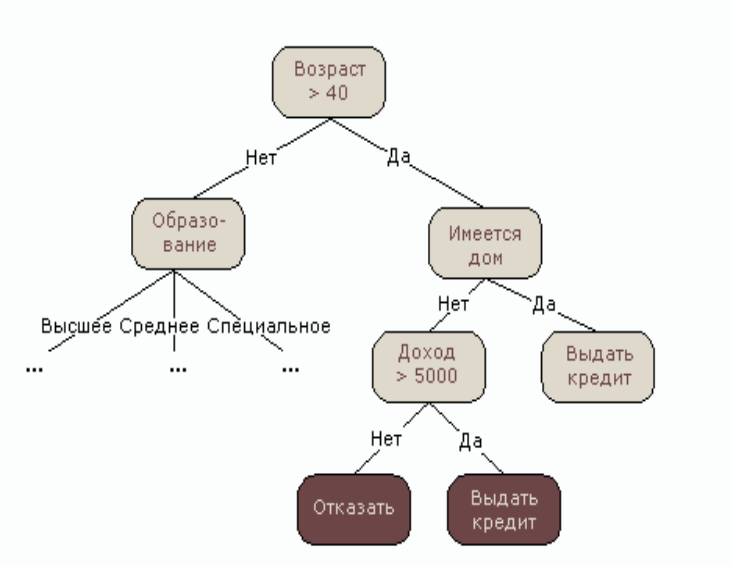


График 2. Пример схемы дерева решений.

Как и регрессионный анализ, деревья решений являются методом изучения статистической взаимосвязи между одной зависимой переменной и несколькими независимыми (предикторными) переменными. Базовое отличие метода деревьев решений от регрессионного анализа заключается в том, что взаимосвязь между значением зависимой переменной и значениями независимых переменных представлена не в виде общего прогнозного уравнения, а в виде древовидной структуры, которую получают с помощью иерархической сегментации данных.[5]

Энтропия Шеннона помогает осуществить эффективное разделение выборки на каждом этапе, при выборе переменной, по которой производить разделение. Энтропия указывает на степень хаотичного распределения предиктора (мера неоднородности множества), чем больше ее значение тем хаотичнее разброс.

Энтропия находит правила (предикаты), на основе которых разбивать тренировочный набор данных, таким образом, чтобы уменьшалось среднее

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i,$$

Формула энтропии Шэннона.

3. МЕТОДОЛОГИЯ ПОСТРОЕНИЯ КАРТЫ.

3.1. Необходимое программное обеспечение.

1. Программный продукт, используемый для автоматизации решения задачи интеллектуального анализа данных и машинного обучения – Python.
Язык Python стал самым популярным средством для анализа данных после выхода отлично документированной библиотеки scikit-learn, в которой реализовано большое количество алгоритмов машинного обучения. Кроме scikit-learn, популярны также библиотеки TensorFlow и Theano (эти библиотеки также реализуют различные методы анализа данных, но выигрывают у scikit-learn только в количестве реализованных техник работы с нейронными сетями).

Использованные библиотеки:

- Pandas для извлечения и подготовки данных: популярная библиотека, которая предоставляет высокоуровневые структуры данных, которые просты в использовании и интуитивно понятны. Он имеет много встроенных методов для группировки, объединения данных и фильтрации, а также выполнения анализа временных рядов. Pandas может легко получать данные из разных источников, таких как базы данных SQL, файлы CSV, Excel, JSON, и манипулировать данными для выполнения операций над ними.
- Matplotlib для визуализации данных: Это стандартная библиотека Python, используемая каждым ученым для создания 2D-графиков и диаграмм. Он довольно низкоуровневый, то есть требует больше команд для создания красивых графиков и диаграмм, чем с некоторыми продвинутыми библиотеками. Однако оборотной стороной этого является гибкость. С достаточным количеством команд вы можете создать практически любой вид графиков с Matplotlib. Может создавать разнообразные диаграммы, от гистограмм и диаграмм рассеяния до не декартовых графов координат.

- Scikit-Learn для работы с классическими алгоритмами ML: одна из самых популярных библиотек ML. Он поддерживает множество контролируемых и неконтролируемых алгоритмов обучения. Примеры включают в себя линейные и логистические регрессии, деревья решений, кластеризацию, k-средних и так далее. Он основан на двух основных библиотеках Python, NumPy и SciPy. Он добавляет набор алгоритмов для общих задач машинного обучения и интеллектуального анализа данных, включая кластеризацию, регрессию и классификацию. Даже такие задачи, как преобразование данных, выбор объектов и методы ансамбля, могут быть реализованы в несколько строк.
- NumPy: Это модуль расширения для Python, в основном написанный на C. Это гарантирует, что предварительно скомпилированные математические и числовые функции и функции Numpy гарантируют высокую скорость выполнения.

В задаче решения скоринга все вышеописанные библиотеки были использованы для:
- загрузки собранных данных,
- первичной предобработки,
- проведения анализа переменных,
- построения моделей машинного обучения,
- оценивания моделей,
- визуализации полученных результатов.

2. SQL — простыми словами, это язык программирования структурированных запросов (SQL, Structured Query Language), который используется в качестве эффективного способа сохранения данных, поиска их частей, обновления, извлечения из базы и удаления.

Главный инструмент оптимизации и обслуживания базы данных — вот, для чего нужен SQL, хотя он и не ограничен этими целями. Возможности обработки охватывают команды определения представлений, указания прав

доступа, схем отношений (в том числе, их удаления и изменения), взаимодействие с другими языками программирования, проверку целостности, задание начала и завершения транзакций.

В нашей задаче SQL был использован в следующих этапах:

- сбор и фильтрация необходимых данных из нескольких источников (база кредитного бюро, база пенсионных отчислений, база с анкетными данными и так далее),
- агрегация данных и соединение в одну общую выборку,
- расчеты переменных.

3.2. Сбор необходимых данных.

Для скорингового моделирования требуются огромные репрезентативные выборки, которые составляются из исторически выданных кредитов. Данные должны быть максимально полными и достоверными, то есть необходимо минимизировать вероятность потери данных или возникновения в них ошибок впоследствии технического сбоя.

Период выборки выбирается экспертно и зависит от цели и области использования скоринговой модели. Выбор короткого временного периода может привести к проблеме невозможности расчета многих переменных, слишком большой период может быть уже неактуальным для использования.

Определение целевой переменной опять-таки зависит от конечной цели модели. Зависимая переменная может быть, как количественной (сумма просрочки), так и качественной (плохой или хороший).В основном в качестве таргета банки используют флаг выхода на просрочку 90 дней. Это усредненный показатель для всех продуктов банка (как для краткосрочных кредитов, так и для долгосрочных). Статистически доказано, что именно выход на просрочку более 90 дней говорит о явном плохом качестве заемщика, тогда как выход на просрочку до 90 дней не всегда может быть намеренным, и считается вполне нормальным для кредиторов.

Разработка скоринговой карты начинается с анализа исторических данных о поведении прошлых и текущих заемщиков для предположения поведения будущих заемщиков.

Независимыми переменными может быть личная информация о заемщике:

- возраст, пол, семейное положение,
- количество иждивенцев, детей, членов семьи,
- социальный статус, образование,
- стаж на последнем/предыдущем месте работы, общий опыт работы,
- сфера деятельности, должность,
- тип регистрации,
- время проживания по текущему/предыдущему адресу,
- адрес регистрации совпадает с адресом фактического проживания,
- ежемесячный подтвержденный доход по основному месту работы и из других источников дохода (пенсия, аренда, алименты и т.п.),
- ежемесячный неподтвержденный доход,
- ежемесячный доход семьи,
- флаг наличия депозита в банке, сумма,
- предполагаемые ежемесячные расходы и т.д.

3.3. Предобработка данных.

В большинстве случаев на практике первично загруженные данные требуют предобработки, так как содержат пропуски, аномалии и выбросы, строчный формат (неподходящий для многих моделей машинного обучения). Такие помехи могут значительно повлиять на точность результатов предсказания моделей. По этой причине необходимо привести данные в нормальный вид, проведя анализ переменных, или другими словами, проведя предобработку данных.

1. Пропуски.

Как правило, данные содержат в себе пропуски. Это связано с различными причинами: ошибка при вводе данных сотрудником банка, клиент сам отказался давать ответ при заполнении анкеты.

На данном этапе необходимо понять какое действие стоит применить по отношению к пропускам. На практике считается, что если пропусков менее 5%, то их можно удалить из выборки, так как отсутствие строк с пропусками не повлияет на конечный результат. Если наличие пропущенных значений больше 5%, то данные следует подвергнуть подробному анализу: пропуск может означать отсутствие (например, иждивенцев, квартиры или мобильного телефона), клиент может сознательно не указать какую-то информацию. В подобных случаях имеет смысл пропуск заменить каким-то заведомо не встречающимся в данных значением и включить в анализ.

В данной работе в исходных данных почти для каждой переменной встречались строки с пропущенными значениями. В некоторых случаях строки с пропусками были удалены, так как их было незначительное количество. В остальных - были заменены на значения, не встречающиеся ранее в данных- на значение -999). Также удалены признаки, которые не содержали более 80% значений.

2. Выбросы.

При анализе распределения параметров часто встречаются аномальные показатели, которые называются выбросами. Выбросы так же как и пустые значения, могут возникнуть технически, при неправильном вводе данных. Но также может быть что данные истины и достоверны. Например, очень большая зарплата. Обычно, такие записи удаляются, поскольку они сильно влияют на производительность модели.

Принцип удаления выбросов стандартный: если значение выброса лежит за пределами 3 стандартных отклонений, за 25 или 75 перцентилями.

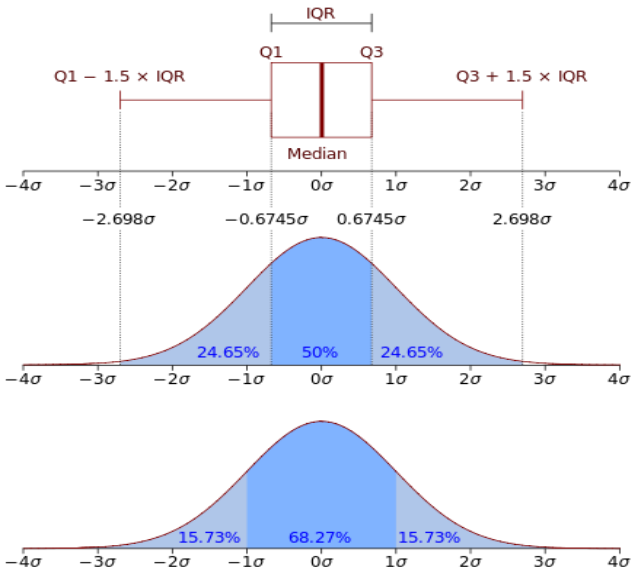


График 3. Пример нормального распределения значений параметра.

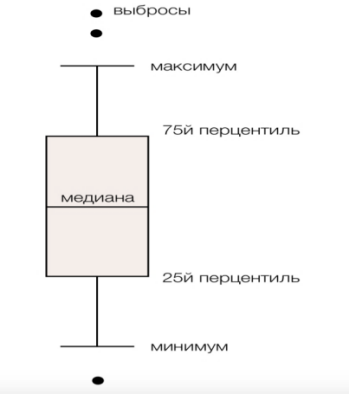


График 4. Пример выявления выбросов.

3. Обработка категориальных признаков.

Все значения входных переменных были переведены в числовой формат, атрибутивные признаки переведены в булево пространство при помощи кодировки One-Hot.

Основная идея такой кодировки — это представление категориального признака, как вектора в векторном пространстве размерностью, соответствующей количеству возможных категорий. При этом значение координаты этой категории берется за единицу, а все остальные координаты обнуляются. С булевыми значениями все совсем просто, они превращаются в вещественные единицы или нули.

4. Стандартизация и нормализация.

На последнем шаге была выполнена стандартизация значений числовых признаков, так как это является важным условием успешного обучения моделей, ведь многие алгоритмы машинного обучения чувствительны к масштабированию данных. В результате среднее значение каждого числового признака стало равно 0, а стандартное отклонение – 1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

нормализация по методу **минимакс**

$$z = \frac{x - \mu}{\sigma}$$

Z-масштабирование

Формула 1. Формулы нормализации и масшабирования значений.

Проведя процесс предобработки данных, и тем самым получив так называемую обучающую выборку, уже можно приступать к построению моделей, так как алгоритмы уже могут дать хорошие результаты на обработанных данных.Однако существуют следующие методы для обеспечения лучшей эффективности классификатора перед непосредственным их использованием.

3.4. Отбор переменных.

При отборе переменных учитывают такие моменты, как: независимость между объясняющими переменными, итоговую прогностическую способность модели, интерпретируемость получаемых коэффициентов при значимых переменных.

1. Корреляция. Оценка линейной зависимости между количественными переменными.

На данном этапе нужно исключить коррелирующие между собой предикторы, чтобы избежать искажения оценки параметров модели. Также удаляются предикторы, имеющие высокую корреляцию с целевым показателем.

2. Изменение размерности векторного пространства признаков с использованием Principal Components Analysis (PCA)

Если векторное пространство признаков слишком велико (миллионы признаков) или мало (менее десятка), то можно применить методы повышения или понижения размерности пространства:

• Для повышения размерности можно использовать часть обучающей выборки как опорные точки, добавив в вектор признаков расстояние до этих точек. Этот метод часто приводит к тому, что в пространствах более высокой размерности множества становятся линейно разделимыми, и это упрощает задачу классификации.

• Для понижения размерности чаще всего используют PCA. Основная задача метода главных компонент — поиск новых линейных комбинаций признаков, вдоль которых максимизируется дисперсия значений проекций элементов обучающей выборки.[8]

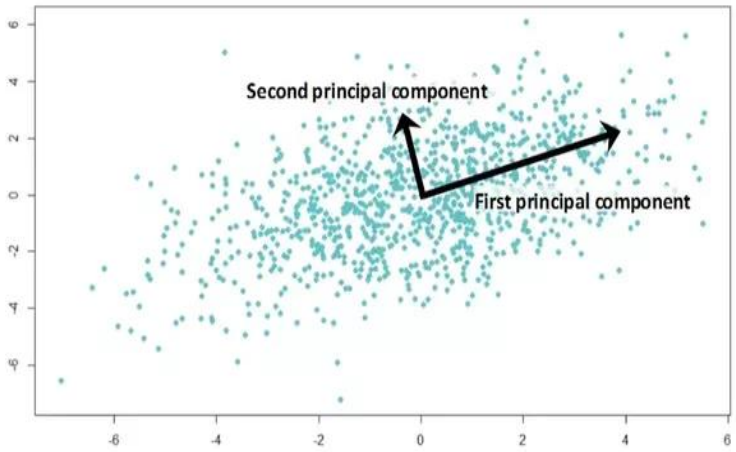


График 5. Пример работы PCA.

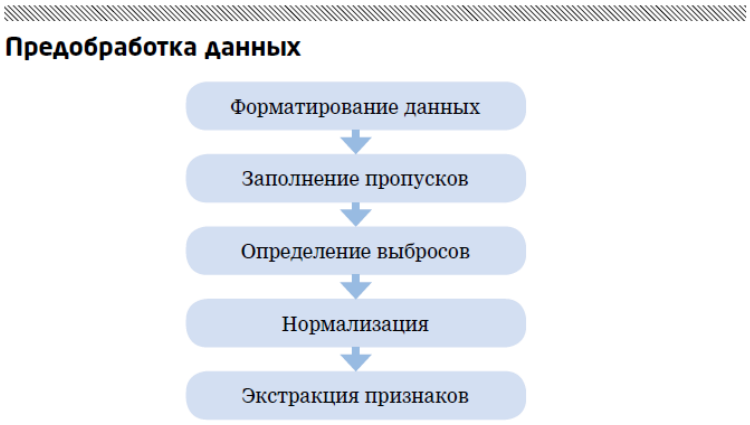


Схема 2. Процесс предобработки данных.[4]

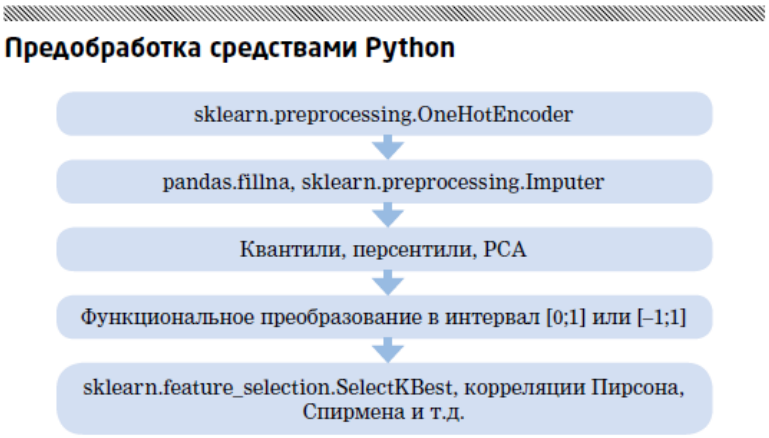


Схема 3. Схема предобработки данных средствами Python.[4]

2.5. Репрезентативность выборки.

После того как выборка готова к использованию необходимо проверить распределение целевой переменной в ней. Распределение плохих клиентов будем называть BadRate (BR). BR – доля плохих клиентов среди всех в выборке. Обычно нормальным распределением целевой переменной считается интервал от 10% до 30%.

Для больших объемов данных несбалансированность является нормой. Для изменения репрезентативности выборки используют два основных метода:

- дублирование миноритарного класса (oversampling)

Недостатком первого метода является тот факт, что простое дублирование прецедентов может не влиять никаким образом на одни методы обучения и вести к переобучению других.

• удаление мажоритарного класса (undersampling).

При удалении объектов, относящихся к мажоритарному классу, возможна потеря важной для классификации информации, что также нежелательно. [9]

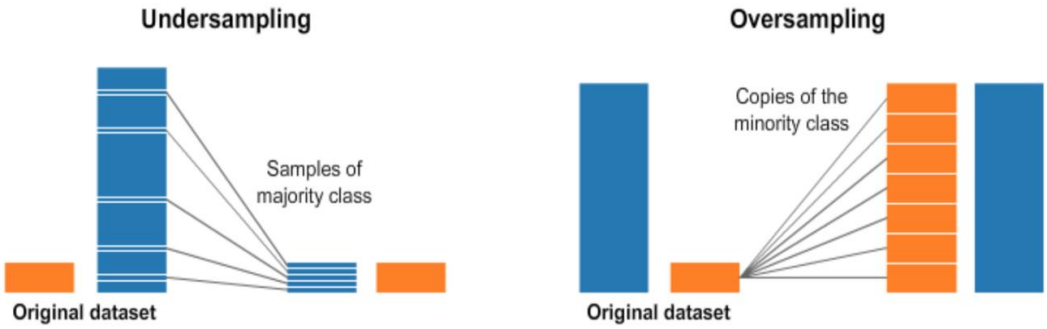


Схема 4. Пример проблемы несбалансированности данных.

Наиболее часто для решения проблемы несбалансированности выборки применяется метод синтетического размножения объектов миноритарного класса SMOTE (Synthetic Minority Oversampling Technique). Новый синтезированный объект по методу SMOTE строится следующим образом:

• вычисляется разность d = xb – xa между векторами xa, xb признаков соседних объектов a, b из миноритарного класса;

• формируется вектор признаков для нового синтезированного объекта x~d=xa+cd, где c~N(0,1). [9]

Существуют разные модификации метода SMOTE, в которых при генерации объектов миноритарного класса используются ближайшие соседи как из миноритарного, так и из мажоритарного класса, и генерируемые объекты располагаются ближе к границе разделения классов или дальше от нее. [9]

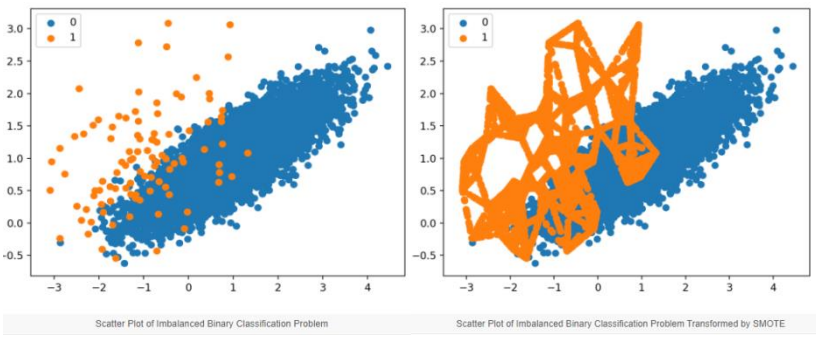


График 6. Пример применения SMOTE

3. ПОСТРОЕНИЕ МОДЕЛЕЙ.

При выборе алгоритма машинного обучения необходимо учитывать важность интерпретации полученных результатов, сложность внедрения полученной модели в процесс принятия решений.

4.1. Алгоритмы

- Логистическая регрессия.

Это статистический метод для анализа набора данных, в котором есть одна или несколько независимых переменных, которые определяют результат. Результат измеряется дихотомической переменной (в которой есть только два возможных результата). Цель логистической регрессии состоит в том, чтобы найти наиболее подходящую модель для описания взаимосвязи между дихотомической характеристикой, представляющей интерес (зависимая переменная = отклик или переменная результата) и набором независимых (предикторных или объяснительных) переменных.

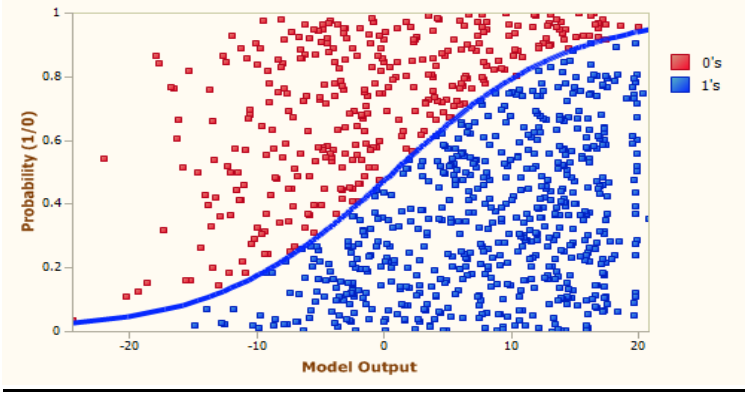


График 7. Пример классификации клиентов с помощью алгоритма логистической регрессии.

- Support Vector Machine

В настоящее время считается, что SVM является одним из самых мощных инструментов среди различных алгоритмов классификации. Исследования показывают, что метод опорных векторов (Support Vector Machine, далее SVM),

основная идея которого заключается в переходе от исходного пространства признаков в пространство более высокой размерности (или даже бесконечномерное) и поиск в нем гиперплоскости максимально разделяющей классы, зарекомендовал себя как весьма эффективный метод классификации. Два важных параметра ядра, это C и γ. Применяя метод поиска по сетке, мы можем найти лучшие значения C и гаммы для ядра.

Одним из главных недостатков этого метода является то, что он, как и нейронные сети, работает по принципу «черного ящика» и не поддается объяснению или интерпретации доступным человеку образом. Что не отказываться от преимуществ весьма точной классификации, которую может обеспечить SVM, применяют технику извлечения правил.

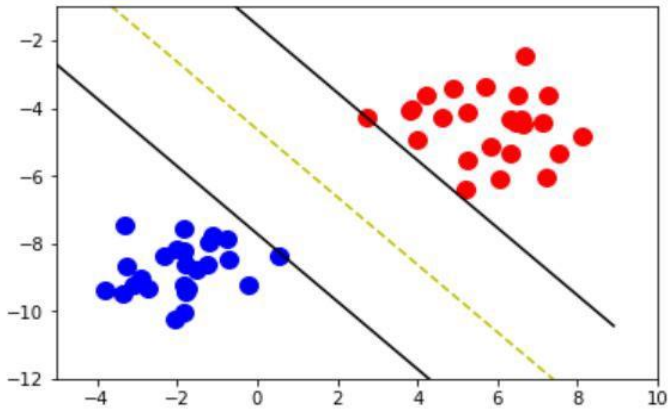


График 8. Пример классификации клиентов с помощью метода опорных векторов

- Случайный лес Random Forest

Случайные леса или леса случайных решений - это метод обучения ансамбля для классификации, регрессии и других задач, который работает путем построения множества деревьев решений во время обучения и вывода класса, который является режимом классов (классификация) или средним прогнозом (регрессия) отдельных деревьев. Случайные леса принятия

решений корректируют привычку деревьев решений чрезмерно соответствовать их обучающему набору.[10]

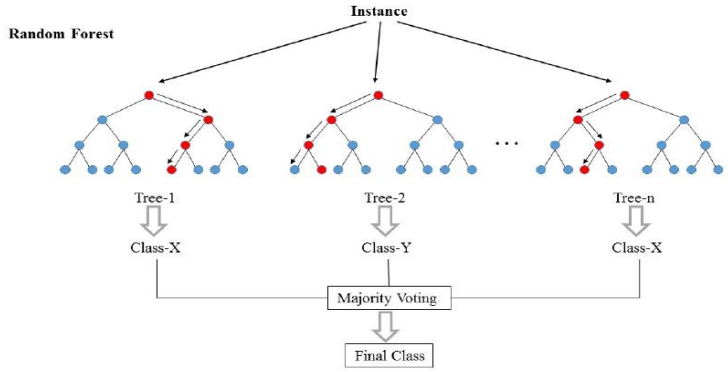


График 9. Пример классификации клиентов с помощью алгоритма случайного леса

- KNN

Алгоритм k-ближайшего соседа является алгоритмом классификации, и он контролируем: он берет группу помеченных точек и использует их, чтобы узнать, как пометить другие точки. Чтобы пометить новую точку, она смотрит на помеченные точки, которые находятся ближе всего к этой новой точке (это ее ближайшие соседи), и голосует за соседей, поэтому любой меткой, обозначенной большинством соседей, является метка новой точки (« k ”- количество соседей, которые он проверяет).

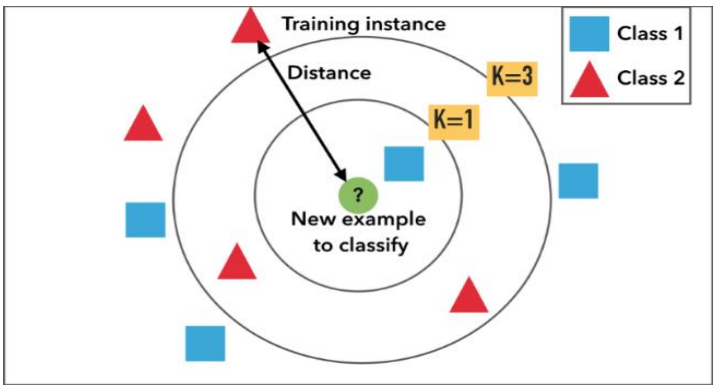


График 10. Пример классификации клиентов с помощью алгоритма KNN

4.2. Тестирование моделей.

Обычно в процессе машинного обучения данные делятся на обучающие и тестовые наборы; обучающий набор затем используется для обучения модели, а тестовый набор используется для оценки производительности модели.

Использование тестовую выборку для настройки гиперпараметров, и для оценки качества модели лишает возможности использовать ее для оценки качества модели. То есть для оценки качества модели необходим независимый набор данных, который не использовался для построения модели и настройки ее гиперпараметров и применяется лишь однократно для оценки качества модели.

Поэтому случайное разбиение на обучающую и тестовую выборки можно применять только тогда, когда задача заключается в том, чтобы построить базовую модель машинного обучения, не прибегая к оптимизации гиперпараметров.

Также такой подход может привести к проблемам с дисперсией. Проще говоря, проблема дисперсии относится к сценарию, в котором наша точность, полученная в одном тесте, сильно отличается от точности, полученной в другом тестовом наборе с использованием того же алгоритма. Чтобы избежать подгонки, выполняется 10-кратная перекрестная проверка.

1. Перекрестная проверка K Fold Cross Validation.

Итак, решением этой проблемы является использование перекрестной проверки K-Fold для оценки производительности, где K - любое число.

Процесс перекрестной проверки K-Fold:

- данные делятся на k подмножеств,
- метод удержания повторяется k раз, так что каждый раз одно из k поднаборов используется в качестве набора тестов / набора проверки, а

другие k-1 поднаборы объединяются для формирования обучающего набора,

- оценка ошибки усредняется по всем k испытаниям.

Как можно видеть, каждая точка данных попадает в проверочный набор ровно один раз и попадает в обучающий набор k-1 раз. Это значительно уменьшает смещение, так как мы используем большую часть данных для подгонки, а также значительно уменьшает дисперсию, так как большая часть данных также используется в наборе проверки. Обмен обучающими и тестовыми наборами также повышает эффективность этого метода. Как общее правило и эмпирические данные, K = 5 или 10, как правило, предпочтительнее, но ничего не установлено, и оно может принимать любое значение.

```
                                    ①     ②  ③    ④           ⑤
cv_mean = cross_val_score(ml_pipe, data, y, scoring=None, cv=5).mean()
```

① Модель, используемая для обучения
② Массив признаков
③ Массив меток
④ Метрика, оцениваемая в ходе перекрестной проверки
- ⑤ Стратегия перекрестной проверки

Код 1. Пример кода для метода перекрестной кросс валидации

Схема 5. Пример работы перекрестной проверки K Fold Cross Validation

4.3. **Настройка гиперпараметров моделей.**

Модели машинного обучения имеют свои настраиваемые гиперпараметры, которые могут улучшить точность предсказания. Примеры гиперпараметров:

- веса переменных,
- скорости обучения,
- кол-во разделяющих линий и так далее.

Часто значения гиперпараметров подбираются случайным перебором для определения наилучшей производительности, но это не рационально и может привести к ошибочному выбору модели.

Grid Search - алгоритм, который автоматически находит лучшие параметры для конкретной модели. Задача Grid Search найти такой вектор гиперпараметров, который дает наилучшую модель, оптимизирую заданную функцию потерь. Grid search принимает на вход модель и различные значения гиперпараметров (сетку гиперпараметров). Далее, для каждого возможного сочетания значений гиперпараметров, метод считает ошибку и в конце выбирает сочетание, при котором ошибка минимальна. [11]

Наиболее часто используемой является квадратичная функция потерь:

$\mathcal{L}(y, y') = (y' - y)^2$. где y - истинное значение выхода модели (которое должно быть получено в идеальном случае), y′- фактический выход модели.

4.4. Методы оценивания моделей.

Поскольку целью применения алгоритмов классификации в кредитном скоринге является разделение клиентов на «хорошие» и «плохие», эффективность алгоритмов оценивается путем сравнения прогноза с реальным показателем.

Задача кредитного скоринга имеет две особенности. Во-первых, классификация плохого кредита как хорошего обходится дороже, чем классификация хорошего кредита как плохого, а во-вторых, в обучающей выборке хороших клиентов всегда больше чем плохих.[5]

В связи с первой особенностью в задаче кредитного скоринга применяются следующие метрики качества алгоритмов. Для этого введем следующие понятия:

- TP (True Positive) — истиноположительный.
- FP (False Positive) — ложноположительный. Ошибка первого рода.
- FN (False Negative) — ложноотрицательный. Ошибка второго рода.
- TN (True Negative) — истиноотрицательный.

Матрица ошибок классификации

Спрогнозировано моделью	Фактически	
	Дефолт	Не-дефолт
Дефолт	**TP** истинно положительные («дефолт» классифицирован как «дефолт»)	**FP** ложно положительные («не-дефолт» классифицирован как «дефолт») **ОШИБКА II РОДА**
Не-дефолт	**FN** ложно отрицательные («дефолт» классифицирован как «не-дефолт») **ОШИБКА I РОДА**	**TN** истинно отрицательные (не-дефолт» классифицирован как «не-дефолт»)

Схема 6. Матрица ошибок классификации в кредитном скоринге.

Метрики качества алгоритмов:

- Accuracy=(TP+TN)/(TP+TN+FP+FN) – доля правильно классифицированных кредитов; Самая простая метрика, но не должна быть единственной метрикой модели, особенно в тех случаях, когда

представители разных классов встречаются с разной вероятностью (несбалансированная выборка).

- Precision=TP/(TP+FP) – точность, то есть доля правильно классифицированных плохих кредитов среди всех наблюдений, отнесенных алгоритмом к плохим кредитам;
- Recall=TP/(TP+FN) – полнота, то есть оценка способности алгоритма распознавать плохие кредиты;
- F1Score=2(Precision·Recall)/(Precision+Recall) – среднее гармоническое точности и полноты;
- NegativePredictiveValue=TN/(TN+FN) – доля правильно классифицированных хороших кредитов среди всех наблюдений, отнесенных алгоритмом к хорошим кредитам;
- Specificity=TN/(TN+FP) – специфичность, то есть оценка способности алгоритма распознавать хорошие кредиты;
- FalseNegativeRate=FN/(TP+FN) – доля плохих кредитов, неправильно отнесенных к хорошим;
- FalsePositiveRate=FP/(TN+FP) – доля хороших кредитов, неправильно отнесенных к плохим.

Чувствительность модели Sensitivity - Доля положительных результатов у «плохих» заемщиков (истинно положительных результатов). Это способность модели правильно определять «плохих» заемщиков и минимизировать убытки, связанные с выдачей кредита недобросовестному клиенту. Модель с высокой чувствительностью характеризуется большей вероятностью «ложных срабатываний».[12]

Специфичность модели Specifity - Доля отрицательных результатов у «хороших» заемщиков (истинно отрицательных результатов). Это способность модели правильно определять «хороших» заемщиков и минимизировать убытки, связанные с отказом в выдаче кредита добросовестному клиенту.

Модель с высокой специфичностью характеризуется большей вероятностью «ложных пропусков». [12]

Для проверки качества скоринговой модели и её предикативной силы используются стандартные статистические коэффициенты:

- Статистика Колмогорова-Смирнова;

Тест Колмогорова-Смирнова используется для предсказания прогнозной силы построенной модели скоринговой карты. Для этого проводят анализ распределения скоринговых баллов для «хороших» и «плохих» заемщиков, находят максимальную разницу между кумулятивными функциями этих распределений. Чем выше значение статистики Колмогорова–Смирнова, тем лучше работает модель, то есть правильнее разделяет заемщиков на «хороших» и «плохих».

$KS = \max_x \left| F_m(x) - G_n(x) \right| \cdot 100$, где $F_m(x)$ и $G_n(x)$ - эмпирические кумулятивные распределения скорингового балла для «плохих» и «хороших» заемщиков; n, m — количество «плохих» и «хороших» заемщиков.

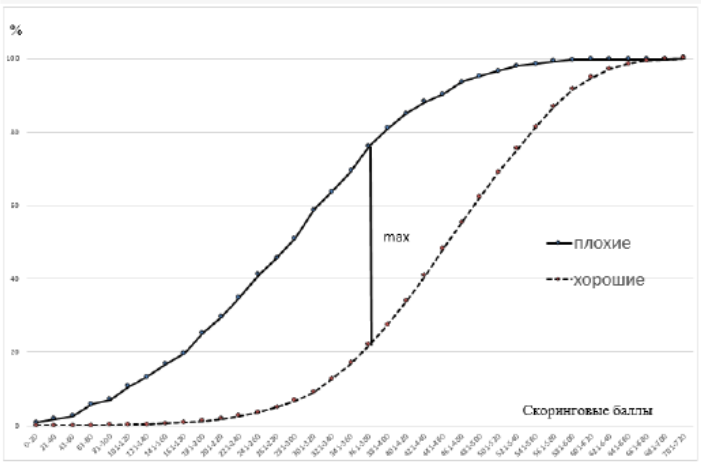


График 11. Пример отображения значения статистики Колмогорова-Смирнова.

- Площадь под ROC-кривой;

ROC-кривая (Receiver Operator Characteristic) – кривая, которая наиболее часто используется для представления результатов бинарной классификации в машинном обучении. ROC-кривая позволяет оценить зависимость доли верно классифицированных положительных исходов от доли неверно классифицированных отрицательных исходов. Чем кривая вогнутее (ближе к верхнему левому углу), тем выше предсказательная способность модели. Наоборот, чем ближе она расположена к диагональной прямой (бесполезному классификатору), тем менее эффективна модель. [13]]

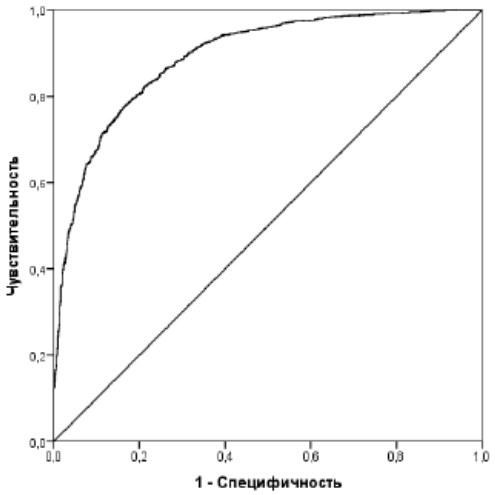


График 11. Пример отображения ROC кривой

● Коэффициент Gini;

Коэффициент Джини используется для анализа качества модели. Он позволяет судить о дискриминирующей способности модели, то есть о способности отличать “плохих” заемщиков от “хороших”. Чем ближе его к 1, тем выше данная способность. [14]]

G=2 (AUC−0.5), где AUC – площадь под кривой ROC.

Основным станадртным методом оценки скоринговых карт являются последние 3 приведенные статистики: Колмогорова-Смирнова, коэффициент GINI и AUC-ROC.

4.4. Результаты.

Выборка собиралась рандомно на базе выданных кредитов за 2019 год из банковских баз данных. У всех кредитов уже прошло 90 дней использования кредита, то есть целевая переменная у них уже «созрела». Собранная выборка содержит 950 записей. Каждая запись имеет 21 признак-предиктор. Признаки являются как числовыми, так и атрибутивными. В качестве зависимой переменной выбран флаг, который принимает категориальную шкалу измерения с двумя категориями. К категории «плохой» относят клиентов, имеющих просроченную задолженность 90 дней и более.

Во избежание нарушения правил конфиденциальности данных показатели параметров были зашифрованы и сгруппированы в программе SQL. Таким образом, исходная выборка выглядит следующим образом:

Цель кредита	Срок кредита	Процентная ставка	Сумма	Пол и Семейное положение	Кол-во иждивенцев	Возраст	Наличие телефона	Кредитная история	Кол-во кредитов	Кредитная нагрузка семьи	Другие траты
3	0.205882	0.333333	0.161770	2	0.0	0.089286	1	2	0.0	1	3
3	0.294118	1.000000	0.058380	4	0.0	0.125000	2	2	0.0	1	3
3	0.205882	1.000000	0.069055	3	0.0	0.232143	2	2	0.0	1	3
3	0.647059	0.666667	0.358094	2	0.0	0.214286	2	2	0.0	1	3
3	0.029412	0.333333	0.023825	2	0.0	0.178571	1	3	0.0	1	1

Депозит	Статус депозитного счета	Недвижимость	Время проживания по текущему адресу	Трудоустройство	Работник из другой страны	Сфера деятельности	Качество	Таргет
1	1	2	0.333333	3	1	3	1	1
2	4	2	0.666667	4	1	3	2	0
1	4	3	0.333333	3	1	4	4	0
1	1	2	0.333333	3	1	3	3	1
1	3	2	0.000000	2	1	3	2	0

Таблица 3. Отображение исходной выборки.

Проведена предобработка данных, включающая в себя группировку значений параметра (Сумма кредита), обработку выбросов (Возраст, Срок кредита), а так же стандартизация всех числовых переменных.

Пустых значений в выборке не было, так как в базах данных банка пустым значениям уже присваиваются определенные значения.

Категориальные переменные также предварительно были зашифрованы в SQL.

Затем обработанная выборка использовалась для построения моделей машинного обучения, таких как линейный дискриминант, логистическая регрессия, SVM, KNN, случайный лес для определения лучшей модели среди выбранных и дальнейшего ее улучшения.

```
Logistic Regression Accuracy: 0.7622377622377622
Linear Discriminant Accuracy: 0.7762237762237763
Naive Bayes Accuracy: 0.6153846153846154
Decision Tree Accuracy: 0.6643356643356644
RF Accuracy: 0.8321678321678322
SVM Accuracy: 0.7412587412587412
KNN Accuracy: 0.7622377622377622
Cross Validation RF: 84.04%
Smote RF: 87.50%
```

Точности алгоритмов при построении моделей после предобработки данных.

Далее была проведена оптимизация входных параметров,а точнее их количества. На всех входных параметрах алгоритмы показывали следующую производительность:

```
Все параметры RF ROC AUC=0.921971
Все параметры LR ROC AUC=0.850469
Все параметры DTC ROC AUC=0.795900
Все параметры kNN ROC AUC=0.836801
Все параметры SVM ROC AUC=0.858221
```

ROC-AUC алгоритмов.

Самым сильным алгоритмом является сллучайный лес, поэтому дальнейшие расчеты были произведены на его базе.

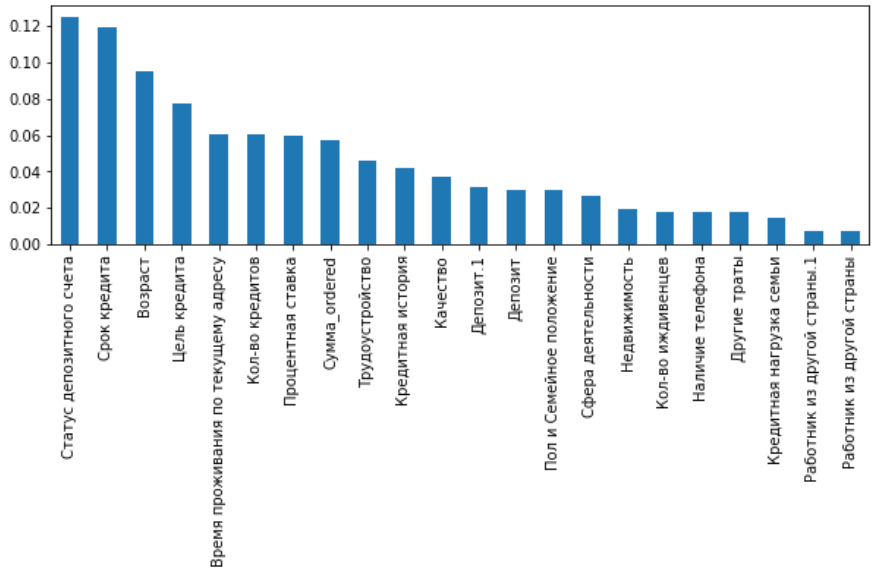


График 12. Предикторы по степени значимости.

Переменные отсортированы по степени влияния. Среди них путем перебора и наблюдения за изменением показателя ROC_AUC были оставлены 4 параметра: срок кредита, статус депозитного счета, флаг того является ли работник гражданином другой страны, Другие траты клиента. При добавлении остальных параметров метрика изменялась несильно.

Далее, была проделана настройка параметров алгоритма случайный лес, и найдены оптимальные гиперпараметры. Выведена матрица ошибок и изображена ROC-кривая:

	Confusion matrix	
	Pred Default	Pred Not Default
Default	47	10
Not default	12	74

Accuracy 0.85

	Default	Not Default
Кол-во	57	86
Precision	0.80	0.88
Recall	0.82	0.86
F1	0.81	0.87

Матрица ошибок полученного алгоритма.

Убытком от неправильной классификации может являться упущенная выгода, если в кредите отказано клиенту с хорошими кредитными рисками

(ошибка первого рода), или потеря невыплаченных заёмщиком средств,в случае одобрения плохой кредитной операции (ошибка второго рода).

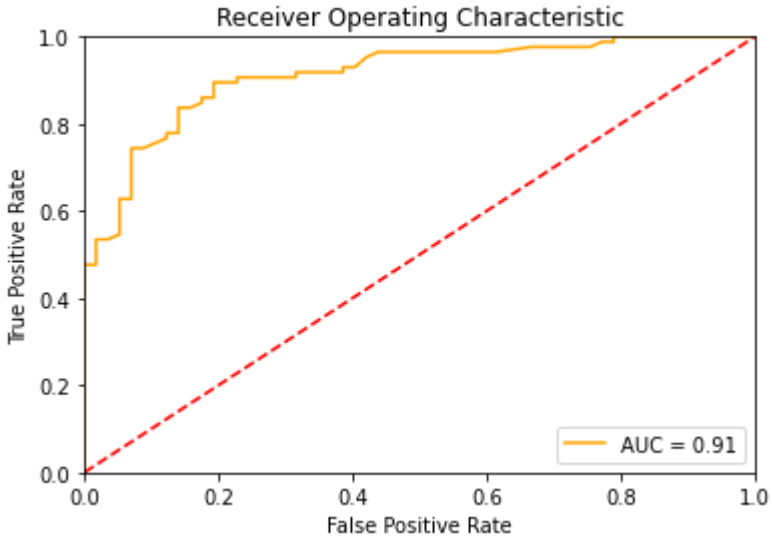


График 13. ROC- кривая

ЗАКЛЮЧЕНИЕ

Банки сталкиваются с широким спектром рисков в их повседневной деятельности. Основной деятельностью банков является привлечение средств с помощью выдачи различных кредитов физическим и юридическим лицам. Следовательно, оценивание кредитного риска является одной из главных задач в банковской сфере.

В ходе проделанной работы была изучена система построения скоринговых карт. Проанализированы существующие методы, используемые банками для создания скоринговых моделей. Рассмотрены разные статистические подходы к анализу качества модели.

Высокий показательнь эффективности классификации клиентов в задаче кредитного скоринга могут показать модели построенные на базе алгоритмов машинного обучения. Для практической реализации был выбран, изучен и использован в работе один из самых востребованных инструментов Python. С его помощью на основе реальных данных были построены модели машинного обучения для создания скоринговых карт. Для работы были использованы базовые библиотеки NumPy и Pandas. При этом были проведены все этапы процесса разработки скоринговой карты – от предварительного анализа данных до оценки построенной модели. Стандартизация распределения признаков, разбиение набора данных на обучающее и тестовое множества, построение матрицы классификации осуществлялись с помощью библиотеки Scikit-learn

Полученные результаты позволяют сделать вывод, что классификация заемщиков по рейтингу кредитоспособности может быть эффективно решена методом случайного леса, а затем SVM. Следом идет модель логистической регрессии, knn,и завершает список алгоритм деревья решений. То есть при качественно предобработанной выборке модели машинного обучения показывают очень высокое качество в разделении клиентов на тех кто выйдет на просрочку и нет. Результаты моделей машинного обучения были лучше

стандартных моделей регресии и дерева решений даже до настройки гиперпараметров. Это говорит нам о том, что алгоритмы машинного обучения также могут отлично справляться с задачей выдачи кредита клиенту, минимизируя денежные потери.

Автоматизация такой рутинной процедуры, как скоринг, позволяет банкам сократить затраты на соответствующие операции, а освободившиеся трудовые и финансовые ресурсы направить на решение иных задач. Поэтому использование современных алгоритмов машинного обучения могут помочь в решении реальных бизнес-задач.

ССЫЛКИ

[1] https://cyberleninka.ru/article/n/obzor-metodov-kreditnogo-skoringa/viewer

[2] https://cyberleninka.ru/article/n/postroenie-skoringovyh-kart-s-ispolzovaniem-modeli-logisticheskoy-regressii

[3] https://creditkin.guru/terms/skoring.html

[4] «Скоринг кредитных заявок: как перейти от статистических методов к моделям машинного обучения?» в сборнике профессиональных материалов для 7-й межотраслевой конференции Scoring Day 2019

[5] https://dmkpress.com/files/PDF/978-5-97060-456-4.pdf

[6] https://www.elibrary.ru/item.asp?id=23817008

[7] https://mnogo-kreditov.ru/kredity/chto-takoe-skoring.html

[8] https://netpeak.net/ru/blog/kratkiy-kurs-mashinnogo-obucheniya-ili-kak-sozdat-neyronnuyu-set-dlya-resheniya-zadachi-po-skoringu/

[9] https://cyberleninka.ru/article/n/sovremennye-podhody-k-primeneniyu-metodov-intellektualnogo-analiza-dannyh-v-zadache-kreditnogo-skoringa

[10] https://ru.qwe.wiki/wiki/Random_forest

[11]http://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

[12] https://edu.kpfu.ru/mod/resource/view.php?id=35193

[13] https://loginom.ru/blog/logistic-regression-roc-auc

[14] https://studfile.net/preview/8069285/page:3/

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Е.С.Волкова, В.Б.Гисин, В.И.Соловьев «Современные Подходы К Применению Методов Интеллектуального Анализа Данных В Задаче Кредитного Скоринга», 2017 // Электронная версия на сайте https://cyberleninka.ru/article/n/sovremennye-podhody-k-primeneniyu-metodov-intellektualnogo-analiza-dannyh-v-zadache-kreditnogo-skoringa/viewer
2. А.С.Сорокин «Построение скоринговых карт с использованием модели логистической регрессии», 2014 // Электронная версия на сайте https://cyberleninka.ru/article/n/postroenie-skoringovyh-kart-s-ispolzovaniem-modeli-logisticheskoy-regressii/viewer
3. А.В.Панюков, Е.С.Будина «Применение Системы Кредитного Скоринга Для Организации Процесса Розничного Кредитования», 2009 // Электронная версия на сайте https://cyberleninka.ru/article/n/primenenie-sistemy-kreditnogo-skoringa-dlya-organizatsii-protsessa-roznichnogo-kreditovaniya/viewer
4. В. В. Кочеткова, К. Д. Ефремова «Обзор Методов Кредитного Скоринга», 2017 // Электронная версия на сайте https://cyberleninka.ru/article/n/obzor-metodov-kreditnogo-skoringa/viewer
5. М.В.Шепелева «Модели кредитного и поведенческого скоринга», 2006 // Электронная версия на сайте http://masters.donntu.org/2006/kita/shepeleva/library/metod%20scoring.pdf
6. А.И. Якупов «Применение деревьев решений для моделирования кредитоспособности клиентов коммерческого банка», 2008 // Электронная версия на сайте http://dspace.nbuv.gov.ua/bitstream/handle/123456789/7387/026-Yakupov.pdf?sequence=1

ПРИЛОЖЕНИЯ

Импорт необходимых библиотек

```
import pandas as pd
import numpy as np
import numpy.random as nr
from numpy import loadtxt
import matplotlib.pyplot as plt
import seaborn as sns
import math
from math import sqrt
%matplotlib inline

from imblearn.over_sampling import SMOTE
from collections import Counter

from sklearn.model_selection import train_test_split,cross_val_score
import sklearn.model_selection as ms
from sklearn import metrics
import sklearn.metrics as sklm
from sklearn.metrics import roc_auc_score
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler,RobustScaler,StandardScaler

from sklearn import linear_model
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVR
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import xgboost as xgb
```

```
[111] df=pd.read_excel('CreditScoring2.xlsx')
      X_train, X_test, y_train, y_test = train_test_split(df, df['Таргет'], test_size=0.15,random_state=0)
      print('Shape of dataset: ', df.shape)
      df.head()
```

```
ıe of dataset:  (950, 21)
```

Цель кредита	Срок кредита	Процентная ставка	Сумма	Пол и Семейное положение	Кол-во иждивенцев	Возраст	Наличие телефона	Кредитная история	Кол-во кредитов	Кредитная нагрузка семьи	Другие траты
3	0.205882	0.333333	0.161770	2	0.0	0.089286	1	2	0.0	1	3
3	0.294118	1.000000	0.058380	4	0.0	0.125000	2	2	0.0	1	3
3	0.205882	1.000000	0.069055	3	0.0	0.232143	2	2	0.0	1	3
3	0.647059	0.666667	0.358094	2	0.0	0.214286	2	2	0.0	1	3
3	0.029412	0.333333	0.023825	2	0.0	0.178571	1	3	0.0	1	1

Депозит	Статус депозитного счета	Недвижимость	Время проживания по текущему адресу	Трудоустройство	Работник из другой страны	Сфера деятельности	Качество	Таргет
1	1	2	0.333333	3	1	3	1	1
2	4	2	0.666667	4	1	3	2	0
1	4	3	0.333333	3	1	4	4	0
1	1	2	0.333333	3	1	3	3	1
1	3	2	0.000000	2	1	3	2	0

Описательная статистика входящих переменных

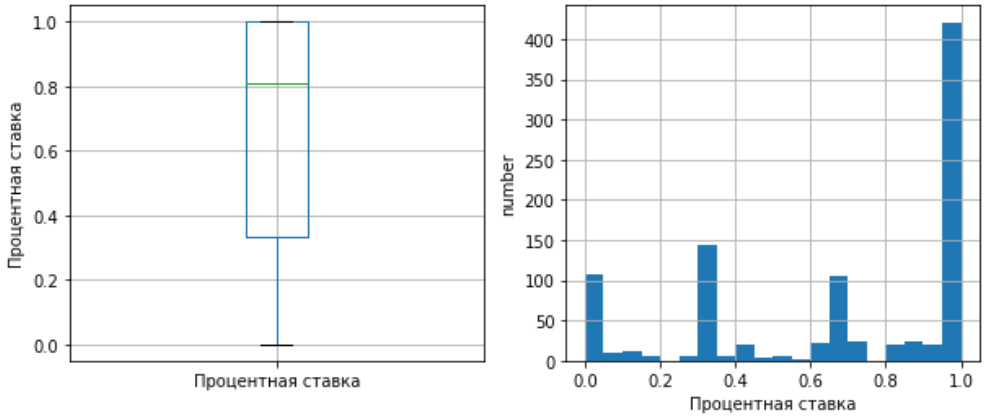
```
df.describe()
```

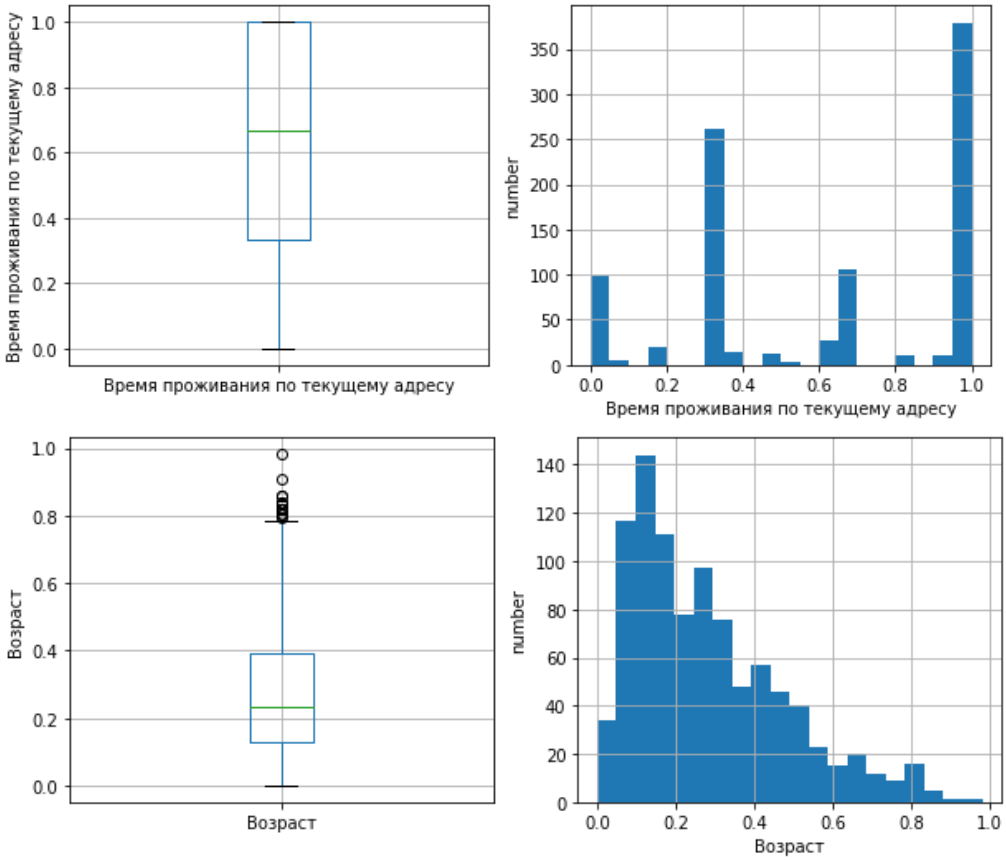
	Цель кредита	Срок кредита	Процентная ставка	Пол и Семейное положение	Кол-во иждивенцев	Возраст	Наличие телефона	Кредитная история	Кол-во кредитов	Кредитная нагрузка семьи	Другие траты
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000
mean	0.532009	0.266080	0.698553	0.548658	0.551649	0.261901	0.543946	0.545207	0.554123	0.545847	0.544702
std	0.124371	0.167555	0.358610	0.041402	0.094266	0.175265	0.003495	0.136740	0.170992	0.031156	0.028153
min	0.314286	0.029412	0.000000	0.505855	0.514706	0.026491	0.541152	0.364103	0.384236	0.428571	0.531532
25%	0.444976	0.117647	0.333333	0.505855	0.533230	0.125000	0.541152	0.364103	0.384236	0.543329	0.531532
50%	0.503759	0.221992	0.894426	0.583942	0.533230	0.223680	0.541152	0.567929	0.529774	0.543329	0.531532
75%	0.640000	0.372452	1.000000	0.583942	0.533230	0.357403	0.548287	0.567929	0.529774	0.543329	0.531532
max	0.698630	0.823529	1.000000	0.612903	1.000000	0.806921	0.548287	0.800000	1.000000	0.684211	0.612903

Депозит	Статус депозитного счета	Недвижимость	Время проживания по текущему адресу	Трудоустройство	Работник из другой страны	Сфера деятельности	Качество	Таргет
143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000
0.547982	0.533067	0.545911	0.662896	0.547066	0.542211	0.545213	0.543269	0.601399
0.080801	0.200869	0.050302	0.353446	0.064045	0.092303	0.017807	0.079661	0.491331
0.400000	0.272374	0.517730	0.000000	0.438849	0.000000	0.512500	0.445000	0.000000
0.588123	0.272374	0.517730	0.333333	0.516129	0.557814	0.546332	0.445000	0.000000
0.588123	0.645455	0.517730	0.666667	0.540441	0.557814	0.546332	0.555957	1.000000
0.588123	0.735395	0.560811	1.000000	0.540441	0.557814	0.546332	0.555957	1.000000
0.617284	0.735395	0.673684	1.000000	0.646667	0.557814	0.573913	0.686131	1.000000

Анализ распределения числовых

```
numerical = ['Возраст','Сумма','Процентная ставка','Время проживания по текущему адресу']
for var in numerical:
    plt.figure(figsize=(10,4))
    plt.subplot(1, 2, 1)
    fig = df.boxplot(column=var)
    fig.set_title('')
    fig.set_ylabel(var)

    plt.subplot(1, 2, 2)
    fig = df[var].hist(bins=20)
    fig.set_ylabel('number')
    fig.set_xlabel(var)

    plt.show()
```





Группировка значений переменной «Сумма кредита»

```
labels = ['Q'+str(i+1) for i in range(0,10)]
print(labels)
X_train['Сумма_label'], bins = pd.qcut(x=X_train['Сумма'], q=10, labels=labels, retbins=True, precision=3, duplicates='raise')
X_train['Сумма_disc'], bins = pd.qcut(x=X_train['Сумма'], q=10, retbins=True, precision=3, duplicates='raise')

X_test['Сумма_label'] = pd.cut(x = X_test['Сумма'], bins=bins, labels=labels)
X_test['Сумма_disc'] = pd.cut(x = X_test['Сумма'], bins=bins, labels=labels)
X_train.head()

X_train.groupby(['Сумма_label'])['Сумма_label'].count().sort_values(ascending=False).index[0]
X_train.groupby(['Сумма_disc'])['Сумма_disc'].count().sort_values(ascending=False).index[0]
```

```
['Q1', 'Q2', 'Q3', 'Q4', 'Q5', 'Q6', 'Q7', 'Q8', 'Q9', 'Q10']
Interval(0.0396, 0.0571, closed='right')
```

```
def impute_na(df_train, df_test, variable):
    most_frequent_category = df_train.groupby([variable])[variable].count().sort_values(ascending=False).index[0]
    df_train[variable].fillna(most_frequent_category, inplace=True)
    df_test[variable].fillna(most_frequent_category, inplace=True)
for variable in ['Сумма_label']:
    impute_na(X_train, X_test, variable)
X_train = X_train.drop('Сумма_disc', axis = 1)
X_test = X_test.drop('Сумма_disc', axis = 1)
```

```
ordered_labels = X_train.groupby(['Сумма_label'])['Таргет'].mean().to_dict()
print(ordered_labels)
X_train['Сумма_ordered'] = X_train['Сумма_label'].map(ordered_labels)
X_test['Сумма_ordered'] = X_test['Сумма_label'].map(ordered_labels)

X_train = X_train.drop('Сумма_label', axis = 1)
X_test = X_test.drop('Сумма_label', axis = 1)

X_train = X_train.drop('Сумма', axis = 1)
X_test = X_test.drop('Сумма', axis = 1)
```

```
{'Q1': 0.49382716049382713, 'Q2': 0.5121951219512195, 'Q3': 0.3670886075949367, 'Q4': 0.4691358024691358, 'Q5': 0.5061728395061729,
```

Обработка выбросов

```
[ ] outlier = ['Срок кредита','Возраст']
    for var in outlier:
        IQR = df[var].quantile(0.75) - df[var].quantile(0.25)
        Lower_fence = df[var].quantile(0.25) - (IQR * 3)
        Upper_fence = df[var].quantile(0.75) + (IQR * 3)
        print([var],Lower_fence,'&', Upper_fence)
```

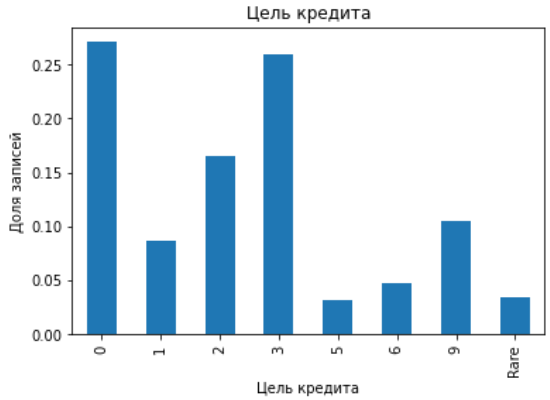
```
['Срок кредита'] -0.588235292 & 1.0588235270000002
['Возраст'] -0.6724138320000002 & 1.1918103742500001
```

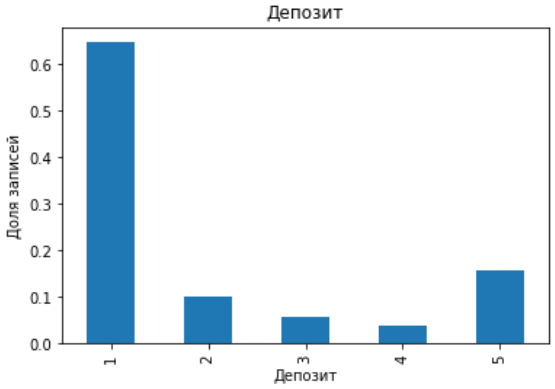
```
[ ] def top_code(df, variable, top):
        return np.where(df[variable]>=top, top, df[variable])

    for df in [X_train, X_test]:
        df['Срок кредита'] = top_code(df, 'Срок кредита', 1.0588)
        df['Возраст'] = top_code(df, 'Возраст', 1.1918)

    df['Возраст'].describe()
```

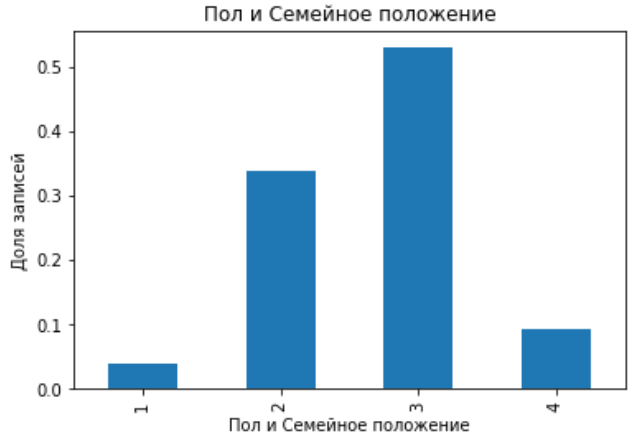
```
count    143.000000
mean       0.261901
std        0.175265
min        0.026491
25%        0.125000
50%        0.223680
75%        0.357403
max        0.806921
Name: Возраст, dtype: float64
```

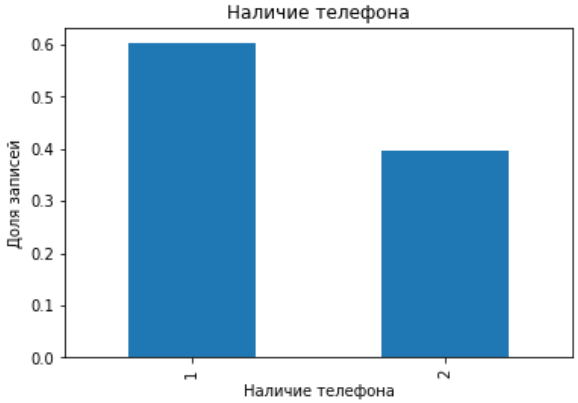
```
[ ] X_train['Кол-во кредитов'] = X_train['Кол-во кредитов'].astype('category')
    X_test['Кол-во кредитов'] = X_test['Кол-во кредитов'].astype('category')
    X_train['Кол-во иждивенцев'] = X_train['Кол-во иждивенцев'].astype('category')
    X_test['Кол-во иждивенцев'] = X_test['Кол-во иждивенцев'].astype('category')

    discrete = ['Кол-во кредитов','Кол-во иждивенцев']
    categorical = ['Статус депозитного счета','Кредитная история','Трудоустройство','Работник из другой
    def rare_imputation(variable):
        # find frequent labels / discrete numbers
        temp = X_train.groupby([variable])[variable].count()/np.float(len(X_train))
        frequent_cat = [x for x in temp.loc[temp>0.03].index.values]

        X_train[variable] = np.where(X_train[variable].isin(frequent_cat), X_train[variable], 'Rare')
        X_test[variable] = np.where(X_test[variable].isin(frequent_cat), X_test[variable], 'Rare')
    for var in categorical:
        rare_imputation(var)
    for var in discrete:
        rare_imputation(var)
```

```
for var in categorical:
    (X_train.groupby(var)[var].count() / np.float(len(X_train))).plot.bar()
    plt.ylabel('Доля записей')
    plt.title(var)
    plt.show()
```

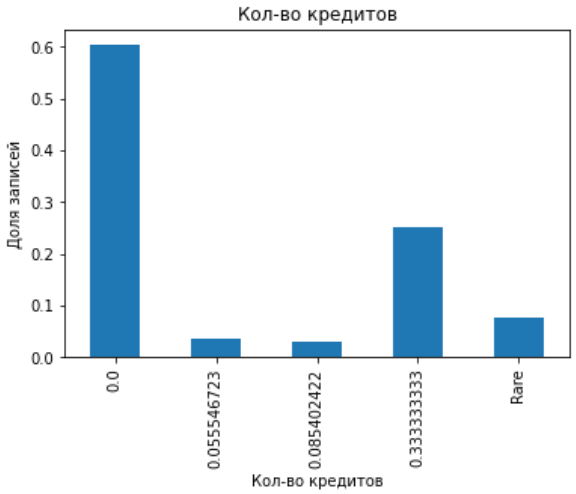


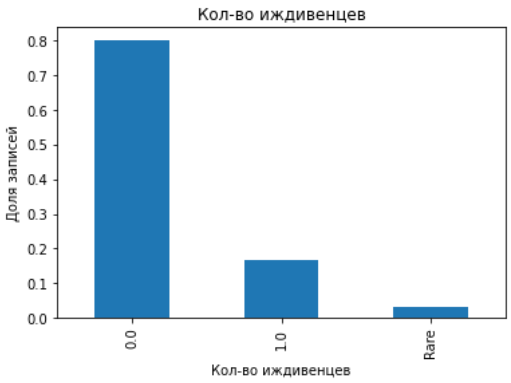







```
for var in discrete:
    (X_train.groupby(var)[var].count() / np.float(len(X_train))).plot.bar()
    plt.ylabel('Доля записей')
    plt.title(var)
    plt.show()
```






```
def encode_categorical_variables(var, target):
        ordered_labels = X_train.groupby([var])[target].mean().to_dict()
        X_train[var] = X_train[var].map(ordered_labels)
        X_test[var] = X_test[var].map(ordered_labels)
for var in categorical:
    encode_categorical_variables(var, 'Таргет')
for var in discrete:
    encode_categorical_variables(var, 'Таргет')
```

Стандартизация значений переменных

```
training_vars = ['Возраст', 'Статус депозитного счета', 'Кредитная история', 'Срок кредита','Трудоустройство', 'Кол-во кредитов', 'Работник из другой
'Процентная ставка', 'Сфера деятельности', 'Кол-во иждивенцев','Кредитная нагрузка семьи', 'Другие траты', 'Время проживания по текущему адресу','Каче
for var in training_vars:
    '''print(var, df[var].skew() )'''
```

```
for var in training_vars:
    max_abs_value = 2
    if abs(df[var].skew()) >= max_abs_value:
        '''print (var)'''
scalerR = RobustScaler()
X_train_scaledR = scalerR.fit_transform(X_train[['Работник из другой страны','Кол-во иждивенцев']])
X_test_scaledR= scalerR.transform(X_test[['Работник из другой страны','Кол-во иждивенцев']])

for var in training_vars:
    max_abs_value = 2
    if (1 <= abs(df[var].skew()) <=2):
        '''print (var)'''
scalerMM = MinMaxScaler()

X_train_scaledMM= scalerMM.fit_transform(X_train[['Кол-во кредитов','Недвижимость','Кредитная нагрузка семьи','Другие траты', 'Депозит']])
X_test_scaledMM= scalerMM.transform(X_test[['Кол-во кредитов','Недвижимость','Кредитная нагрузка семьи','Другие траты', 'Депозит']])

for var in training_vars:
    max_abs_value = 2
    if ((df[var].skew()) <=1):
        '''print (var)'''
scalerN = StandardScaler()
X_train_scaledN = scalerN.fit_transform(X_train[['Возраст', 'Статус депозитного счета','Кредитная история','Срок кредита','Трудоустройство',
'Работник из другой страны', 'Процентная ставка','Сфера деятельности','Время проживания по текущему адресу','Качество','Цель кредита','Депозит',
'Пол и Семейное положение','Наличие телефона', 'Сумма_ordered']])
X_test_scaledN = scalerN.transform(X_test[['Возраст', 'Статус депозитного счета','Кредитная история','Срок кредита','Трудоустройство',
'Работник из другой страны', 'Процентная ставка','Сфера деятельности','Время проживания по текущему адресу','Качество','Цель кредита','Депозит',
'Пол и Семейное положение','Наличие телефона', 'Сумма_ordered']])
```

```
smt = SMOTE(random_state=20)
train_input_new, train_output_new = smt.fit_sample(train_input, train_output)
X_train, X_t, Y_train, Y_t = train_test_split(train_input_new, train_output_new, test_size=0.20, random_state=0)
```

```
logreg = LogisticRegression(fit_intercept=True)
logreg.fit(X_train, Y_train)
logregprediction=logreg.predict(X_test)

lda = LinearDiscriminantAnalysis()
lda.fit(X_train, Y_train)
lda.fit(X_train, Y_train)
lda_prediction=lda.predict(X_test)

gnb = GaussianNB()
gnb.fit(X_train, Y_train)
gnbprediction=gnb.predict(X_test)

dtc= DecisionTreeClassifier()
dtc.fit(X_train,Y_train)
dtcprediction=dtc.predict(X_test)

rfc=RandomForestClassifier(n_jobs=-1,random_state=123)
rfc.fit(X_train,Y_train)
rfcprediction=rfc.predict(X_test)

svc = svm.SVC()
svc.fit(X_train,Y_train)
svcprediction=svc.predict(X_test)

knn = KNeighborsClassifier()
knn.fit(X_train,Y_train)
knnprediction=knn.predict(X_test)

print("Logistic Regression Accuracy:",metrics.accuracy_score(logregprediction,y_test))
print("Linear Discriminant Accuracy:",metrics.accuracy_score(lda_prediction,y_test))
print("Naive Bayes Accuracy:",metrics.accuracy_score(gnbprediction,y_test))
print("Decision Tree Accuracy:",metrics.accuracy_score(dtcprediction,y_test))
print("RF Accuracy:",metrics.accuracy_score(rfcprediction,y_test))
```

```
smt = SMOTE(random_state=20)
train_input_new, train_output_new = smt.fit_sample(train_input, train_output)
train_input_new = X_train
train_ouput_new = y_train
```

```
seed_val = 1000000000
np.random.seed(seed_val)

model_all_features = RandomForestClassifier()
model_all_features.fit(X_train, y_train)

y_pred_test = model_all_features.predict_proba(X_test)[:, 1]
auc_score_all = roc_auc_score(y_test, y_pred_test)
print('Все параметры RFC ROC AUC=%f' % (auc_score_all))
```

```
Все параметры RFC ROC AUC=0.921971
```

```
features = pd.Series(model_all_features.feature_importances_)
features.index = X_train.columns
features.sort_values(ascending=False, inplace=True)
features.plot.bar(figsize=(10,3))
```

```
seed_val = 1000000000
np.random.seed(seed_val)
model_one_feature = RandomForestClassifier()
model_one_feature.fit(X_train[features[0]].to_frame(), y_train)
y_pred_test = model_one_feature.predict_proba(X_test[features[0]].to_frame())[:, 1]
auc_score_first = roc_auc_score(y_test, y_pred_test)
print('Один параметр ROC AUC=%f' % (auc_score_first))
```

```
Один параметр ROC AUC=0.721236
```

```
] tol = 0.001
  print('doing recursive feature addition')
  features_to_keep = [features[0]]
  count = 1
  for feature in features[1:]:
      print()
      print('testing feature: ', feature, ' which is feature ', count,
            ' out of ', len(features))
      count = count + 1
      model_int = RandomForestClassifier()
      model_int.fit(X_train[features_to_keep + [feature] ], y_train)
      y_pred_test = model_int.predict_proba(X_test[features_to_keep + [feature] ])[:, 1]
      auc_score_int = roc_auc_score(y_test, y_pred_test)
      print('New Test ROC AUC={}'.format((auc_score_int)))
      print('All features Test ROC AUC={}'.format((auc_score_first)))
      diff_auc = auc_score_int - auc_score_first

      if diff_auc >= tol:
          print('Increase in ROC AUC={}'.format(diff_auc))
          print('keep: ', feature)
          print
          auc_score_first = auc_score_int
          features_to_keep.append(feature)
      else:
          print('Increase in ROC AUC={}'.format(diff_auc))
          print('remove: ', feature)
          print
  print('total features to keep: ', len(features_to_keep))
```

```
seed_val = 1000000000
np.random.seed(seed_val)
final = RandomForestClassifier()
final.fit(X_train[features_to_keep], y_train)
y_pred_test = final.predict_proba(X_test[features_to_keep])[:, 1]
auc_score_final = roc_auc_score(y_test, y_pred_test)
print('Выбранные 4 переменные ROC AUC=%f' % (auc_score_final))
```

```
Выбранные 4 переменные ROC AUC=0.912995
```

```
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(100, 500, num = 11)]
max_depth.append(None)
random_grid = {
 'n_estimators': n_estimators,
 'max_features': max_features,
 'max_depth': max_depth
 }
rfc=RandomForestClassifier()
rfc_model = RandomizedSearchCV(estimator = rfc, param_distributions = random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
rfc_model.fit(X_train[features_to_keep], y_train)
print(rfc_model.best_params_)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed:  4.3min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:  8.0min finished
{'n_estimators': 400, 'max_features': 'sqrt', 'max_depth': 300}
```

```
seed_val = 1000
np.random.seed(seed_val)
rfc2 = RandomForestClassifier(n_estimators=400, max_depth=300, max_features='sqrt')
rfc2.fit(X_train[features_to_keep],y_train)
rfc2_predict = rfc2.predict_proba(X_test[features_to_keep])[:, 1]
auc_score_final2 = roc_auc_score(y_test, rfc2_predict)
print('Выбранные параметры ROC AUC=%f' % (auc_score_final2))
```

```
Выбранные параметры ROC AUC=0.912383
```

```
df = pd.concat([X_train.append(X_test), y_train.append(y_test)], axis = 1)
labels = np.array(df['CreditStatus'])
probabilities = rfc2.predict_proba(X_test[features_to_keep])

def score_model(probs, threshold):
    return np.array([1 if x > threshold else 0 for x in probs[:,1]])
scores = score_model(probabilities, 0.5)

def print_metrics(labels, scores):
    metrics = sklm.precision_recall_fscore_support(labels, scores)
    conf = sklm.confusion_matrix(labels, scores)
    print('                 Confusion matrix')
    print('                 Pred Default               Pred Not Default')
    print('Default      %6d' % conf[0,0] + '             %5d' % conf[0,1])
    print('Not default  %6d' % conf[1,0] + '             %5d' % conf[1,1])
    print('')
    print('Accuracy  %0.2f' % sklm.accuracy_score(labels, scores))
    print(' ')
    print('           Default      Not Default')
    print('Кол-во   %6d' % metrics[3][0] + '        %6d' % metrics[3][1])
    print('Precision  %6.2f' % metrics[0][0] + '        %6.2f' % metrics[0][1])
    print('Recall     %6.2f' % metrics[1][0] + '        %6.2f' % metrics[1][1])
    print('F1         %6.2f' % metrics[2][0] + '        %6.2f' % metrics[2][1])
print_metrics(y_test, scores)
```

```
                 Confusion matrix
                 Pred Default               Pred Not Default
Default          47                10
Not default      12                74

Accuracy  0.85

           Default      Not Default
Кол-во        57           86
Precision   0.80         0.88
Recall      0.82         0.86
F1          0.81         0.87
```

https://www.ismaitm.lv/images/Files/Theses/2020/01_NN/19_ITM2020_Abdykalykova.pdf

The 18th INTERNATIONAL
SCIENTIFIC CONFERENCE
**INFORMATION
TECHNOLOGIES AND
MANAGEMENT 2020** ***April 23-24,
2020, ISMA, Riga, Latvia***

Credit scoring using machine learning algorithms

Abdykalykova A.E.

Kazakh National Technical University after K.I.Satpaeva, Almaty

Corresponding author's e-mail: *asselya081296@gmail.com*

Abstract

Credit scoring it is a way to predict borrower's behavior or future possibility of delay using input and historical information. It affects government's economy, thus financial companies should give loans only to responsible and solvent a part of population. Every financial institution has its own scorecard models. Usually, those models are based on logistic regression and decision tree, because of their simple interpretability. Since the data volume grows, variety and types of modern predictive methods develop the possibility of increasing the predictive power of models is growing too. This thesis is going to be about process of building qualitative model and modern optimization methods.

Keywords: Scorecard; Credit; Machine learning.

Introduction

Recently, consumer spending has become one of the key factors in macroeconomic conditions worldwide. Therefore, it is important to focus on credit scoring in order to better predict consumer behavior.

Credit scoring is a method that helps to decide whether to provide loans to consumers, it is a probability of person's debt repay in a timely manner, based on person's credit history. People are considered financially reliable when their score is higher. Credit scoring eliminates the human factor and uses only reliable data.

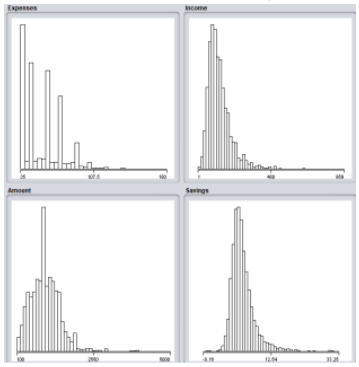
There are two main problems in credit scoring: giving a loan to a bad borrower and refusing to a good one. However, there are many ways to accomplish this problem, and some of them are more effective than others.

Advanced statistical and mathematical methods provide fast and automatic tools that help to make effective decisions. Models based on machine learning algorithms and artificial intelligence are believed to be more effective to support approval process in finance companies. The combination of machine learning methods can make a big contribution to the lending system and will be much more complicated in terms of use. Because machine learning forecasts are more adaptive and flexible to change, they can produce more accurate results. Therefore, the purpose of this thesis is to identify the most reliable and effective method.

Methods

The dataset consist of data from personal information, credit bureau, transactions and so on. Because of consumer privacy protection laws, all individual identification data were encrypted. The aim of any machine-learning model is the identification of statistically reliable relationships between input data features and the target variable. The target variable is a binary value, indicating whether an account is delinquent by 90 days or more within 12 months.

The dataset was preprocessed before the final feature selection using several classical methods like Chi-squared, Information Gain and new methods such as Lime/Shad (understanding of parameter's influence on model prediction), PCA (reducing data dimension). The selected features contain information about type of job, experience, number of credits, incomes & expenses, age, and marital status and so on.



Example of features' distribution

Then data was trained and tested with multiple number of supervised machine learning models, such as Logistic Regression, Decision Tree, Random Forest, Naïve Bayes, XGBoost, Support Vector Machine. Cross-validation technique was performed and parameters of algorithms were correctly tuned for models' better work. The performance was evaluated using several performance measurements such as accuracy, AUC-score, ROC-curve, Gini, confusion matrix, recall, precision, F-score.

Results

The result of the research should propose the best model to predict Credit Defaults.

Models were compared by several metrics and after all of the comparisons was selected the best model.

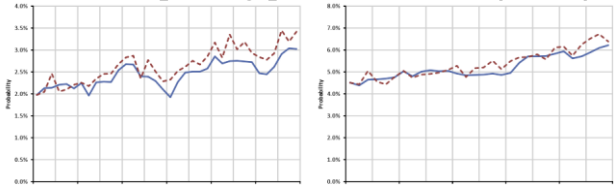
There are given main metrics' results:

Accuracy: XGBoost – 81,2%, SVM – 77%, Random Forest – 75%, Logistic Regression – 74.6%,Naïve Bayes - 69% and Decision Tree - 70%.

AUC: XGBoost – 0.854,SVM – 0.751, Random Forest – 0.774, Logistic Regression – 0.698,Naïve Bayes – 0.685 and Decision Tree – 0.704.

F-score: XGBoost – 87,2%, SVM – 86.3%, Random Forest – 84.9%, Logistic Regression – 82%,Naïve Bayes 76% and Decision Tree - 80%.

SVM *Comparing prediction and fact of target* XGBoost



Conclusion

XGBoost and SVM have shown the best results comparing with such a popular machine learning models used in credit scoring as Decision tree and Logistic Regression. Applying such methods as preprocessing dataset to avoid imbalance (and as a consequence incorrect result of the models), new optimizing methods in feature selection (reducing over-fitting, improving accuracy, reducing training time) helped to achieve such a good results, when most of the models have a high values of metrics.

From risk management perspective, the aggregation of machine-learning forecasts may have much to contribute to the management of systemic risk.

References

1. Stein, R.M., 2005. The relationship between default prediction and lending profits: Integrating ROC analysis and loan pricing. Journal of Banking & Finance
2. Baesens, B., T. V. Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen. 2003.Benchmarking state-of-the-art classification algorithms for credit scoring.
3. Hand, D., Henley, W., 1997. Statistical classification methods in consumer credit scoring: A review. Journal of the Royal Statistical Society.
4. Galindo, J., Tamayo, P., 2000. Credit risk assessment using statistical and machine learning: Basic methodology and risk modeling applications.
5. Anne Kraus, 2014. Recent Methods from Statistics and Machine Learning for Credit Scoring.
6. Foster, D., Stine, R., 2004. Variable selection in data mining: Building a predictive model for bankruptcy.
7. Loh W.-Y. Fifty Years of Classification and Regression Tree