МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**«Южно–Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Главный специалист по данным
ООО «MoneyCare»

_____ Р.Д. Якупов
“___”_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.–м.н., профессор

_____ Л.Б. Соколинский
“___”_____ 2020 г.

Разработка программной системы для классификации заемщиков кредитным агрегатором

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.04.02.2020.308–566.ВКР

Научный руководитель,
к.ф.–м.н., доцент кафедры СП
_____ А.Т. Латипова

Автор работы,
студент группы КЭ–220
_____ Я.М. Юхновец

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
“___”_____ 2020 г.

Челябинск–2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно–Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
09.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистра
студенту группы КЭ–220
Юхновцу Яна Максимовичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»
(магистерская программа «Технологии разработки высоконагруженных систем»)

1. Тема работы (утверждена приказом ректора от 24.04.2020 № 627)

Разработка программной системы для классификации заемщиков кредитным агрегатором

2. Срок сдачи студентом законченной работы: 05.06.2020 г.

3. Исходные данные к работе

3.1. Математические методы обучения по прецедентам. [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron–ML–1 (дата обращения: 26.04.2020).

3.2. Analysis from 50+ papers on the Application of ML in Credit Lending. [Электронный ресурс] URL: https://towardsdatascience.com/my–analysis–from–50–papers–on–the–application–of–ml–in–credit–lending–b9b810a3f38 (дата обращения: 26.04.2020).

3.3. Кредитный скоринг. Не все так страшно. [Электронный ресурс] URL: https://www.webcitation.org/66A4ilFSO?url=http://factoringpro.ru/index.php/credit–scoring–statya/408–skoring–statya–kredit (дата обращения: 26.04.2020).

4. Перечень подлежащих разработке вопросов

4.1. Изучить существующие методы ранжирования клиентов в кредитном скоринге.

4.2. Разработать систему ранжирования клиентов.

4.3. Протестировать работоспособность разработанной системы.

5. Дата выдачи задания: 09.02.2020 г.

Научный руководитель
к.ф.–м.н., доцент кафедры СП А.Т. Латипова

Задание принял к исполнению Я.М. Юхновец

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Кредитный скоринг – это статистический анализ, осуществляемый кредиторами и финансовыми учреждениями для определения кредитоспособности человека. Чаще всего используется в потребительском кредитовании на небольшие суммы. Также возможно его использование в бизнесе сотовых операторов, страховых компаний и т.д. Сам скоринг осуществляется с помощью «скоринговой модели» – специальные веса, которые «взвешивают» математически выраженные характеристики заемщика, влияющие на его способность получить кредит.

Существует несколько видов скоринга [1], но в данном случае нас будет интересовать только один – application scoring – оценка кредитоспособности заемщиков при выдачи кредита. Основывается на обработке первичных данных заемщика.

Существуют несколько готовых решений программного обеспечения, но они подходят исключительно для банков. При этом стоит понимать, что не все клиенты кредитного агрегатора имеют возможности и/или средства для такого программного обеспечения. Поэтому при помощи собственного кредитного скоринга мы можем решить следующие задачи.

- Повысить качество клиентов для банков, отсекая явных дефолтников.
- Применять страховки для рискованных клиентов.
- Уменьшать размер выдаваемого кредита ненадежным клиентам.

Отсюда следует, что автоматизация классификации заемщиков для кредитного агрегатора является актуальной задачей.

Для достижения цели работы необходимо было решить следующие задачи.

- Произвести анализ предметной области, изучить существующие подходы кредитного скоринга.
- Произвести анализ требований к реализуемой системе.

- Спроектировать систему классификации в соответствии с требованиями.
- Реализовать спроектированную систему.
- Выполнить тестирование реализованной системы.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения и списка используемой литературы. Объем работы составляет 44 страницы, объем библиографии 20 наименований.

Содержание работы

Первая глава, «Анализ предметной области» содержит описание кредитного скоринга, а также существующих методов классификации.

Вторая глава, «Методология исследования», описывает подход к работе над данными.

Третья глава, «Проектирование и реализация системы», описывает проект новой системы и ее реализацию.

Четвертая глава, «Тестирование», содержит описание и результаты написанных и проведенных тестов новой системы.

Заключение резюмирует результаты, полученные в рамках проведенной работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Кредитный скоринг

Кредитная история – это информация о ваших кредитных обязательствах. Она показывает, в какие банки, микрофинансовые организации (МФО) или кредитные потребительские кооперативы (КПК) вы обращались за кредитами и займами. Когда это было, и какие суммы вы брали. Были ли вы созаемщиком или поручителем по чужим кредитам. Платили ли аккуратно или задерживали платежи.

Для финансовых организаций, выдающих кредит, существует две проблемы: во–первых, выдача кредита недобросовестному заемщику, который с большой долей вероятности, не вернет кредит, и, во–вторых, невыдача кредита заемщику, который с большой долей вероятности, вернет его.

Для того чтобы минимизировать количество ошибок в выдаче кредитов, применяется кредитный скоринг. Кредитный скоринг – система оценки кредитоспособности (кредитных рисков), базирующаяся на численных статистических методах. Чаще всего используется в потребительском кредитовании при маленьких суммах. Заемщик должен заполнить анкету, на основании которой, ему будут выставлены баллы. От количества набранных баллов зависит решение о выдаче (не выдаче) кредита [2].

Кредитный скоринг основывается на анализе данных заемщика, связанных с риском невозврата займа. Данные можно условно разделить на две категории.

1) Общеэкономические факторы: уровень инфляции, валовый внутренний продукт, курс валюты. Профессия заемщика, отрасль работы, средний уровень заработной платы и т.д.

2) Индивидуальные факторы: возраст, пол, семейное положение, доходы и расходы, кредитная история, контактная информация.

Также стоит понимать, что общеэкономические факторы служат дополнительным источником информации. Так как данные из этого источника общие и слабо характеризуют конкретного заемщика [3]. Один из лучших источников информации – это кредитная история. Она показывает финансовую дисциплину клиента, содержит информацию о способности погашения сумм, показывает заинтересованность в займе.

При этом банки могут использовать свой набор параметров, как, например, на рисунке 1.

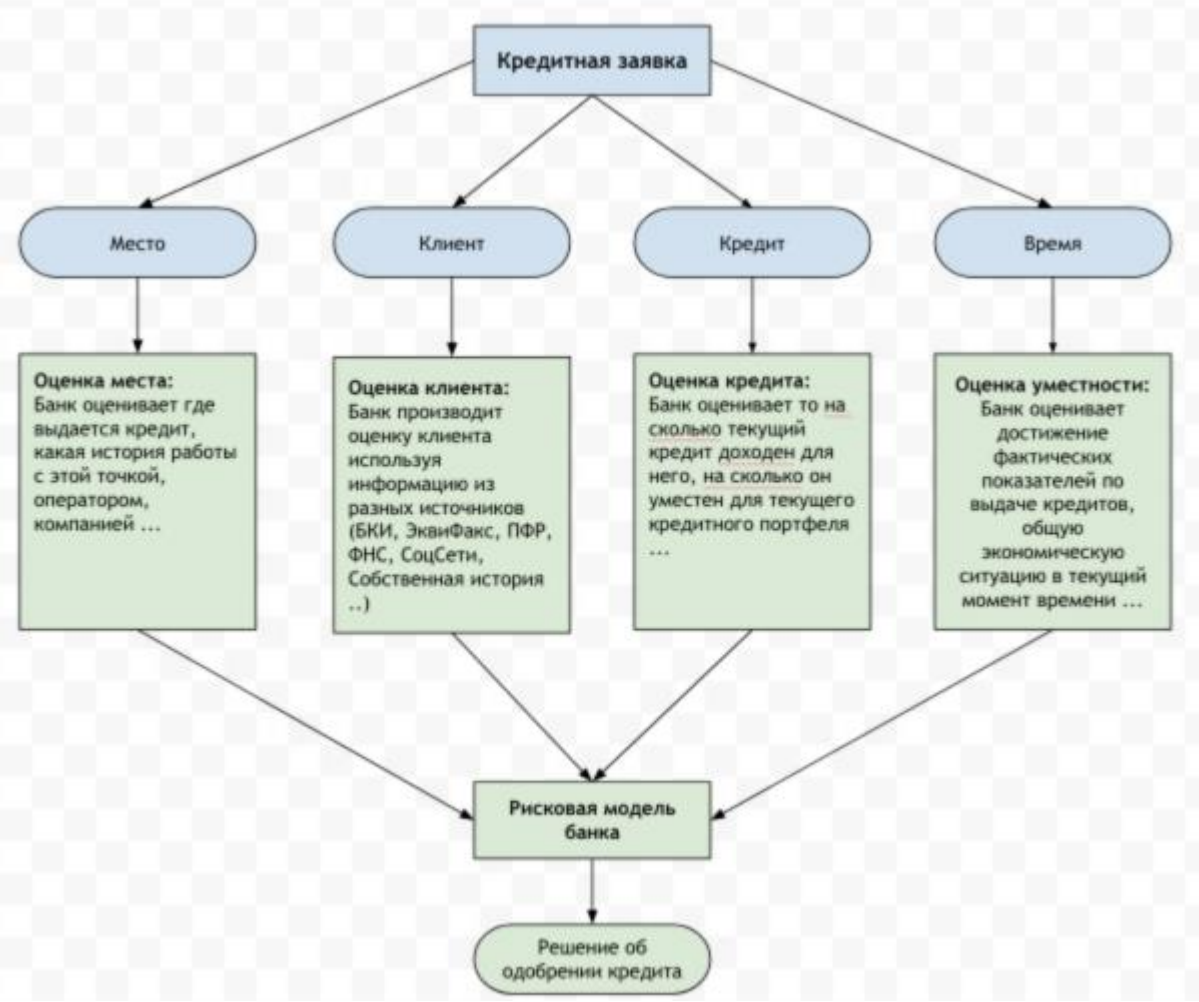

Рис. 1. Пример параметров

1.2. Способы оценивания кредитоспособности

Для оценки риска при кредитовании применяются два подхода. Они могут использоваться как по одиночке, так и вместе [4]:

1) Опираясь на личные заключения.

2) Используя программные системы кредитной оценки.

Необходимо учитывать, что, кроме компьютерной программы, оценивание потенциальной платежеспособности заёмщика производит и сотрудник банка во время личной встречи. Визуальный контроль – ещё один важный фактор одобрения или отказа в кредите.

В самом упрощенном виде скоринговая модель представляет собой взвешенную сумму определенных характеристик. В результате получается интегральный показатель (score); чем он выше, тем выше надежность клиента, и банк может упорядочить своих клиентов по степени возрастания кредитоспособности.

Интегральный показатель каждого клиента сравнивается с неким числовым порогом, или линией раздела, которая, по существу, является линией безубыточности и рассчитывается из отношения, сколько в среднем нужно клиентов, которые платят в срок, для того, чтобы компенсировать убытки от одного должника. Клиентам с интегральным показателем выше этой линии выдается кредит, клиентам с интегральным показателем ниже этой линии – нет.

Все это выглядит очень просто, однако сложность заключается в определении, какие характеристики следует включать в модель и какие весовые коэффициенты должны им соответствовать.

Существуют различные виды кредитного скоринга.

1) Application–scoring: определяет уровень кредитного риска потенциального заемщика на основании данных, доступных в момент подачи заявления.

2) Fraud–scoring: оценивает вероятность мошенничества потенциального заемщика. Нередко мошенники в попытке преодолеть application–скоринг создают образ идеального заемщика. Fraud–scoring призван бороться с ними.

3) Behavioral–scoring: вид скоринга, который позволяет оценить финансовое поведение заемщика и, тем самым, предсказать его поведение,

(наличие/отсутствие просрочек, будущая доходность, вероятность приобретения других банковских продуктов) в процессе обслуживания кредита.

4) Collection–scoring: данный вид скоринга выполняется, когда у клиента имеются просрочки по кредиту. Collection–scoring устанавливает, что следует сделать по отношению к задолжнику: ограничиться напоминанием по телефону или сразу передать дело коллекторам.

Таким образом, кредитный скоринг позволяет банкам снижать риски невозврата кредита, избегать мошенничеств, а также в зависимости от уровня риска определять процентную ставку по кредиту.

1.3. Типология задач классификации

В машинном обучении задачу классификации можно разделить на следующие типы.

1) Двухклассовая (бинарная) классификация (binary classification).

2) Многоклассовая классификация на непересекающиеся классы (multiclass classification).

3) Многоклассовая классификация на пересекающиеся классы (multilabel classification).

Самый простой и явный – решить ее как задачу типа бинарной классификации. В рамках задачи классификации необходимо на основе признакового описание заемщиков определить вероятность их принадлежности к одному из двух классов, тем самым решить задачу бинарной классификации. Первый класс – заемщики, которым кредит можно одобрить. Второй класс – это те заемщики, которым кредит не одобряется. В данном случае нам прежде всего интересует вероятность принадлежности потенциального заемщика к первому классу.

Этот подход имеет очевидную выгоду – упрощение процесса бинарной классификации. Помимо этого, он позволяет использовать более простые модели.

1.4. Обзор существующих методов классификации

Наиболее часто используемые алгоритмы классификации по проблеме кредитного скоринга варьируются в зависимости от конкретного набора данных, выбора функций и многого другого [5].

1) Метод опорных векторов.

2) Градиентный бустинг.

3) Обучение дерева решений.

4) Нейронные сети.

Интересно, что нейронные сети не дают значительно лучших результатов, что заставляет нас сомневаться в их использовании. Это связано с тем, что из–за регуляторных ограничений нейронные сети могут оказаться более сложными для применения на практике по сравнению с другими алгоритмами ML. Также, логику работы нейронной сети сложно интерпретировать как клиентам, так и сотрудникам банка. В связи с этим лучше рассмотреть более простые модели, обладающие большей интерпретируемостью.

1.4.1. Нейронные сети

Область нейронных сетей изначально была вдохновлена целью моделирования биологических нейронных систем, но с тех пор отклонилась от изначальной цели и сфокусировалась на достижении хороших результатов в задачах машинного обучения [9]. Искусственная нейронная сеть – упрощенная математическая модель, разработанная в соответствии с основными принципами функционирования нейронов живого организма.

Аппарат искусственных нейронных сетей (ИНС) представляет собой совокупность математических моделей (нейронных сетей), способных обучаться на массивах данных и с помощью этого решать задачи прикладного характера, будь то прогноз или классификация, с чем лучше всего справляется модель нейронной сети – многослойный персептрон.

К преимуществам нейронных сетей можно отнести.

- Возможность выучить сложные, нелинейные закономерности в данных.
- Скорость на этапе формирования предсказаний.
- Гибкие настройки параметров (кол–во слоев, нейронов).

Среди недостатков стоит выделить следующее.

- Сложно интерпретировать результаты (модель черного ящика).

Требуется много вычислительных ресурсов и данных на стадии обучения.

1.4.2. Логистическая регрессия

Логистическая регрессия является частью более широкого класса алгоритмов, известных как обобщенная линейная модель (glm). В 1972 году Nelder и Wedderburn предложили эту модель для обеспечения возможности использования линейной регрессии в решении проблем, которые нельзя было напрямую решить с помощью линейной регрессии. Фактически они предложили класс различных моделей (линейная регрессия, ANOVA, регрессия Пуассона и т. д.), которые включали логическую регрессию как особый случай.

Фундаментальное уравнение обобщенной линейной модели имеет вид (1):

$$g\big(E(y)\big) = \alpha + \beta x_1 + \gamma x_2, \quad (1)$$

Здесь g() – функция связи, E(y) – ожидаемое значение целевой переменной, а α + βx1 + γx2 – линейный предиктор (α, β, γ прогнозируются). Роль функции связи заключается в том, чтобы «связать» ожидаемые значения y с линейным предиктором.

Логистическая регрессия – это тип множественной регрессии, предназначенный для классификации записей на основе значений поля

ввода. В этом случае выходная переменная является категориальной или двоичной (то есть она может принимать только два значения).

При использовании системы двоичной классификации, любой элемент или наблюдение должны быть разделены по двум категориям. Затем событие связывается с каждым результатом: элемент относится к категории A или элемент принадлежит категории B. Итогом будет расчет вероятности достижения необходимого результат.

Например, если рассматривается результат ссуды, переменная y устанавливается со значениями 1 и 0, где 1 означает, что соответствующий заемщик выплатил ссуду, а 0 – дефолт.

Преимущества.

- Низкая вычислительная стоимость.
- Простота реализации.
- Легко интерпретировать результаты – достаточно просто посмотреть на коэффициенты уравнения.

Недостатки.

- Склонность к недообучению.
- Необходимость в предобработке данных (масштабирование, исключение коррелированных признаков).

1.4.3. Метод опорных векторов

Метод опорных векторов (support vector machine, SVM) – один из широко используемых технологий обучения, применяемый для получения решения задач регрессии и классификации. Суть метода кроется в построении гиперплоскости, разграничивающей предметы выборки наиболее корректным способом. Чем больше расстояние, т.е. зазор между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора [6].

SVM ищет оптимальную гиперплоскость (границу решения) с максимальным запасом, чтобы разделить разные классы. В SVM

используется и метод наименьших квадратов (LS–SVM) для разработки модели кредитного скоринга, и их характеристики сравниваются с другими современными методами в восьми различных наборах данных [7]. Исследована проблема набора данных с дисбалансом, где SVM продемонстрировал стабильность в работе. Как правило, SVM имеет лучшую производительность и меньше вычислительных усилий в эксперименте [8]. Если за наиболее значимый фактор берется платежеспособность покупателя, то при каждом возврате средств похожим заемщиком, возможность его платежеспособности будет возрастать. При этом если условия и картина рынка будет меняться, то профессия заемщика может перестать быть решающим фактором при выдаче кредита. В таком случае компании может понести убытки в связи с потерей кредитоспособности заемщика в перспективе.

1.4.4. Градиентный бустинг

Градиентный бустинг – это метод, используемый для решения задач классификации и регрессии. Он состоит из ряда комбинаций аддитивных моделей (слабое обучение), которые оцениваются итеративно, что приводит к более сильной модель обучения. Обычно используется метод градиента дерева решений модели, однако в этом процессе может использоваться любая модель, например, логистическая регрессия. Градиентый бустинг является альтернативой традиционному методу логистической регрессии, ускоряющий разработку скоринговых решений. Это очень надежный метод, который имеет высокую гибкость в решении задач классификации. Градиентный бустинг дает большие или равные результаты, если сравнивать с логистической регрессией, с меньшими затратами на разработку, можно объяснить двумя характеристиками.

1) Взаимодействие между переменными: градиентный бустинг строит взаимодействие между переменными, предоставляя несколько комбинаций для дальнейшего объяснения события.

2) Интерпретация параметров: градиентный бустинг представляет собой процесс нескольких моделей, и интерпретация переменных не является прямой, поэтому неправильно удалять переменную, потому что она представляет обратную интерпретацию делового смысла. С другой стороны, это очень распространенно исключать переменные при подгонке логистической регрессии, потому что это причина, даже когда она уменьшает общую модель точность и предсказуемость [9].

1.4.5. Деревья решений

Деревья решений и связанные с ними ансамблевые методы, такие как случайный лес, являются базовыми инструментами в области машинного обучения для прогнозной регрессии и классификации. Однако, случайному лесу не хватает интерпретируемости, и, следовательно, данная модель может быть менее значимой в приложениях для кредитного скоринга, где лицам, принимающим решения, и органам регулирования требуется прозрачная функция линейной оценки, которая обычно соответствует функции связи в логистических регрессиях. Дерево же лишено такого недостатка, поскольку процесс принятия решения легко поддается интерпретации, но обладает большей склонностью к переобучению и ступенчатой границей решений.

В задаче кредитного скоринга лучше отталкиваться от более простых моделей, которые дают меньшую точность, но обладают лучшей интерпретируемостью. К таким моделям можно отнести дерево решений, логистическую регрессию и SVM. Если же качество работы таких моделей окажется недостаточным, то придется пожертвовать интерпретируемостью и использовать более сложные подходы, среди которых наиболее часто применяются случайные лес и градиентный бустинг. Нейронные сети же следует использовать, когда точность классификации у всех выше описанных моделей недостаточна, а также когда присутствует достаточное количество примеров для обучения нейронной сети.

1.5. Метрики качества модели

Выбор метрики свойства считается важным этапом в процессе разработки модели машинного изучения, так как как раз подобранная метрика разрешает осуждать о качестве модели. Если выбрать неправильно метрику качества, то последующие этапы разработки модели будут интерпретированы ложно и по этой причине есть вероятность большого количества ошибок, которое повлечет за собой время на их исправление. На выбор метрики влияет предметная область и поставленная цель, которую надо решить.

В данной работе решается задача бинарной классификации. Важно понимать, что нас интересует вероятность принадлежности к классу, так и 5метка класса. Основываясь на этой информации, мы можем попытаться сделать выбор в пользу метрики ROC AUC, отражающей площадь (Area Under Curve) под кривой ошибок (Receiver Operating Characteristic curve). Эта кривая отражает качество алгоритма классификации при всех порогах вероятности [10].

Количественную интерпретацию ROC даёт показатель AUC (area under ROC curve). AUC – это площадь, заключенная в рамки ROC-кривой и осью доли ложных положительных классификаций. Показатель AUC считается качественным в том случае, если его показатель >0,5. При этом показатель <0,5 считается недопустимым (некачественным) для такого метода классификации. Оно также означает, что классификатор действует в обратном направлении (наоборот): чтобы он работать корректно, нужно положительное назвать отрицательным [15].

На рисунке 2 можно увидеть пример графика ROC AUC.

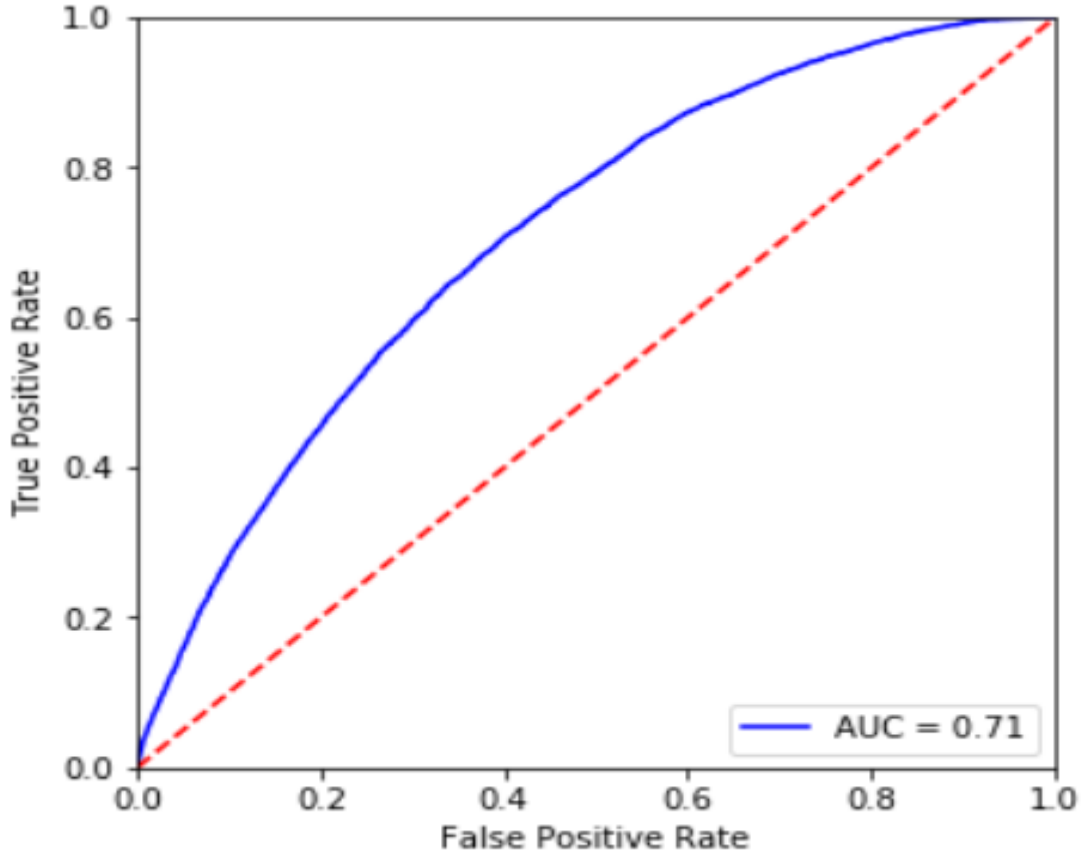

Рис. 2. AUC график

ROC кривая отображает два параметры: True Positive Rate и False Positive Rate.

True Positives (TP) – количество объектов, верно отнесенных моделью к положительному классу.

False Positives (FP) – количество объектов, неверно отнесенных моделью к положительному классу.

False Negatives (FN) – количество объектов, неверно отнесенных моделью к отрицательному классу.

True Negatives (FN) – количество объектов, верно отнесенных моделью к отрицательному классу. Параметр True Positive Rate рассчитывается по формуле (2). Параметр False Positive Rate рассчитывается по формуле (3).

$$TPR = TP / (TP + FN), \quad (2)$$

$$FPR = FP / (FP + TN), \quad (3)$$

Приведенные выше формулы (3) и (4) базируются на элементах матрицы ошибок (рисунок 3), которая является основой для метрик качества в задаче бинарной классификации. Стоит понимать, что критерий ROC AUC используется для сравнения моделей в целом. Для детального понимания ответов от модели, необходимо смотреть на значения матрицы ошибок, точности, охвата, f–меры.

		Предсказанные метки классов	
		Positive class (1)	Negative class (0)
Реальные метки классов	Positive class (1)	True Positive (TP)	False Positive (FP)
	Negative class (0)	False Negative (FN)	True Negative (TN)

Рис. 3. Матрица ошибок

Точность системы в пределах класса – это доля элементов, которые достоверно относятся к данному классу, относительно всей массы элементов, отнесенных к этому классу. Полнота системы – это доля найденных классификатором элементов, которые принадлежат классу, относительно всех элементов этого класса в тестовой выборке. Параметр точности рассчитывается по формуле (4). Параметр охвата рассчитывается по формуле (5).

$$Precision = TP / (TP + FP), \quad (4)$$

$$Recall = TP / (TP + FN), \quad (5)$$

Обычно эти две метрики противоборствуют друг с другом, то есть улучшая одну из них, вторая стремится вниз. В зависимости от поставленной задачи выбирается некоторая точка баланса. Для усреднения Precision и Recall применяется метрика F–score.

При необходимости использования формальных метрик для оценки качества классификатора уместно будет использовать F-меру. Она заключает в себе сразу два необходимых показателя: точность и полноту. Используя F-меру в работе, мы сможем с легкостью оценивать изменения в алгоритме и понимать их динамику (положительная или отрицательная).

1.6. Анализ существующих решений

1.6.1. ZestFinance

Один из передовых стартапов в инновационном скоринге – ZestFinance (США). Основанная на ИИ, его платформа собирает и анализи–рует данные из множества источников, в том числе сведения об онлайн активности выбранных лиц; правда, в отличие от многих конкурирующих платформ, в ZestFinance информацию из соцсетей не используют, считая её малопригодной для своих целей. Отталкиваясь от тысяч сигналов, алгоритмы сервиса высчитывают, какой степени доверия заслуживает человек. Помимо всего прочего, система помогает выявить перспективных заёмщиков среди тех, кому кредитные организации по формальным признакам обычно отказывают в ссудах (например, короткая кредитная история или вообще её отсутствие). Говоря о сферах применения решения, достаточно добавить, что в ZestFinance инвестировал китайский интернет–гигант Baidu, с тем чтобы построить скоринг платформу, которая использовала бы данные о поведении пользователей в поисковике.

1.6.2. Lenddo

В отличие от ZestFinance, гонконгская Lenddo делает ставку именно на оценку профилей заёмщика в социальных сетях и круга его друзей. При удачном прохождении такого «социального скоринга» одобряется микрокредит, обычно в несколько сотен долларов (стартап делает ставку на развивающиеся страны с низким средним доходом населения).

1.6.3. Mail.ru

Свою предсказательную систему оценки кредитных рисков разработала в 2016 году и Mail.ru Group совместно с бюро кредитных историй Equifax. С помощью машинного обучения платформа анализирует крупные массивы данных и, по утверждению создателей продукта, позволяет находить надёжных заёмщиков в том числе в высокорисковых сегментах аудитории.

Можно легко заметить, что ZestFinance и Lenddo, как и проект от Mail.ru подходят только банкам и большим корпорациям. К тому же, большая часть анализа происходит на информации из социальных сетей. Когда в нашем случае вся информация берется из анкеты заемщика.

2. МЕТОДОЛОГИЯ ИССЛЕДОВАНИЯ

2.1. Сбор данных

На текущий момент данные о заемщиках хранятся в облачной платформе Azure [11]. Они хранятся там, потому что есть инструмент Azure Machine Learning [12]. Этот инструмент позволяет быстро собирать и развертывать модели машинного обучения. Удобство в том что все вычисления можно производить, используя браузер и стандартный Jupyter Notebook.

Изначально данные о заемщиках попадают из серверной части кредитного агрегатора, написанного на языке Java. После агрегации и фильтрации они попадают в Azure.

Система сбора данных была реализована на языке Python версии 3.5. Для подключения к Azure используется библиотека azureml. В Azure хранятся подготовленные датасеты, выгружать их можно, обращаясь по имени. Для каждого банка имеется соответственное имя файла. Дальше данные с помощью библиотеки pandas приводятся к нужному виду. Набор данных содержит 66 признаков и 300 тысяч примеров. На рисунке 4 представлена часть данных.

	id_bank	id_mc	createstamp	cr_y	cr_m	cr_d	cr_dw	bank_id	insurance_life	insurance_unemployment	insurance_status_boolen
681023	69265574	69265245	2020-03-15 15:19:00.437	2020	3	15	0	35979754	0.0	0.0	0
681024	69281843	69281657	2020-03-15 18:54:15.813	2020	3	15	0	673465	0.0	0.0	0
681025	69281913	69281657	2020-03-15 18:56:04.807	2020	3	15	0	35979754	0.0	0.0	0
681026	69281915	69281657	2020-03-15 18:56:04.817	2020	3	15	0	219369	0.0	0.0	0
681027	69281928	69281657	2020-03-15 18:57:34.850	2020	3	15	0	8034500	0.0	0.0	0

Рис. 4. Пример датасета

2.2. Очистка данных

Ненужные признаки снижают скорость обучения модели, интерпретируемость и, главное, способность к обобщению. Поэтому необходимо удалить те признаки, которые не влияют на обучение модели. Например, такие признаки как id региона, серия паспорта и т.д. можно удалить. Очистка данных состоит из двух шагов – удаление

неиспользуемых признаков и удаление строк с пустыми значениями. На рисунке 5 представлен код, реализующий очистку данных.

```
      target = ['hcb', 'revo', 'mts', 'mig', 'brs', 'otp', 'rfb', 'post',
'ren', 'ab', 'keb', 'tnb']
      df = {}
      pd_df = {}
      for t in target:
          # Удаление строк с пустыми значениями и неиспользуемых признаков
          not_needed     =     ['is_approve_req_pos','area_id','sms_status',
'smsinforming_calc','passportseries',            'cr_y',            'cr_m',
'cr_d','createstamp','reason_for_cancel','product_code','bank_id','hcb','re
vo','abcard','mig','brs','rfb','ren','ab','keb','svcard','time_refresh_data
','product_id','eir','product_req_id',    'id_mc',    'priority',    'id',
'mobilephone','post','mts', 'otp','tnb']
          df[t] = data[data[t].isin([1,2])].copy()
          target_map = {1: 0, 2: 1}
          df[t][t] = df[t][t].map(target_map)
          pd_df[t] = df[t].dropna()
          not_needed.remove(t)
         pd_df[t].drop(not_needed, axis = 1, inplace = True)
```

Рис. 5. Листинг функции, осуществляющей очистку данных

2.3. Сравнение методов

Оценка эффективности моделей производится с помощью скользящего контроля (кросс–валидации) [13] – это методика, используемая для сравнения и тестирования алгоритмов классификации, прогнозирования и регрессии. Она строится на обучении и последующей оценке модели на имеющихся подвыборках главной (обучающей) выборки. Для этой оценки необходимо кластеризовать исходных ряд разбиений на две группы: обучающую и контрольную. Результатом (оценкой) скользящего контроля при этом станет средний показатель ошибки для всей контрольных подвыборках (для их суммы). Общая схема кросс–валидации представлена на рисунке 6.

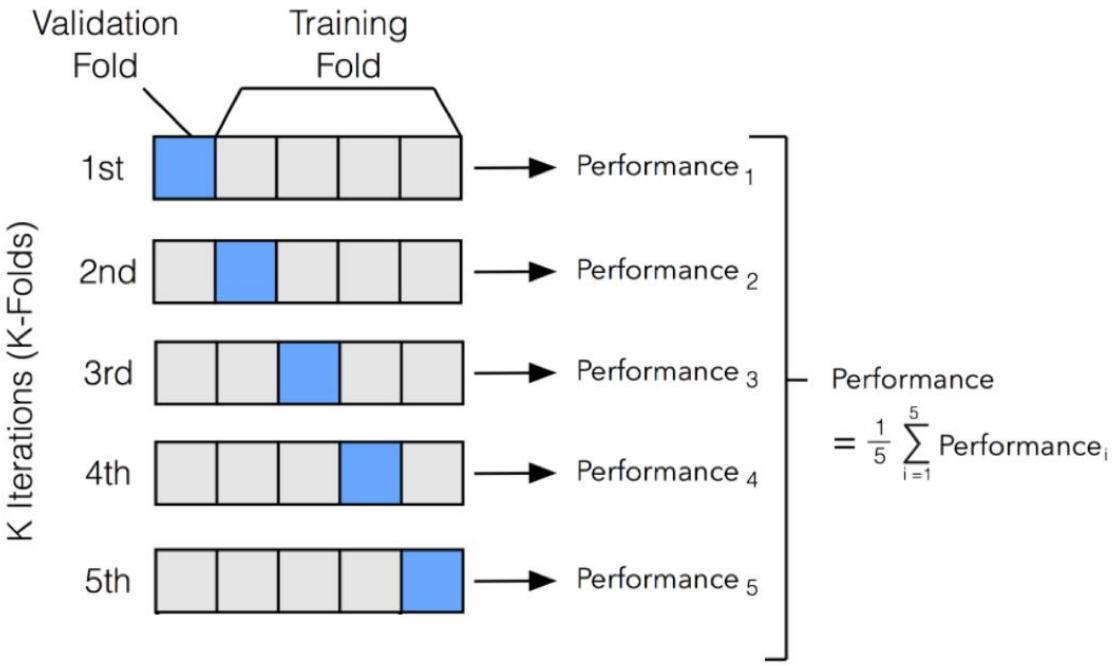

Рис. 6. Схема кросс–валидации

Общий алгоритм проведения экспериментов выглядит следующим образом.

1) Выбор модели машинного обучения.

2) Обучение модели при некоторых начальных параметрах и архитектуре, получение начального результата.

3) Изменение ее параметров с целью получения лучшего результата.

4) Сравнение моделей.

При обучении модели часто возникают две противоположные проблемы – переобучение и недообучение.

Недообучением считаются те случаи, когда заданная модель не предоставляет необходимое качество метрики на обучающей подвыборке и, в конечном итоге, неудовлетворительно описывает обучающую подвыборку. Этой проблемы можно избежать, если использовать при работе только сложные модели.

Переобучением считаются те случаи, когда модели на тестовой подвыборке существенно выше ошибки на обучающей подвыборке. Такая

проблема возникает в следствии использования чересчур сложных моделей. Таким образом, чтобы избежать недо- и переобучения необходимо отказаться от использования слишком простых или, наоборот, слишком сложных моделей.

Имея достоверные данные о плюсах и минусах нескольких типов классификации, были выдвинуты следующие эксперименты.

1) Градиентный бустинг с подбором гиперпараметров.

2) Логистическая регрессия с подбором гиперпараметров.

3. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ СИСТЕМЫ

3.1. Архитектура системы

На рисунке 7 представлена архитектура системы. Система состоит из 3 основных пакетов, объединяющих компоненты. Рассмотрим их подробнее.

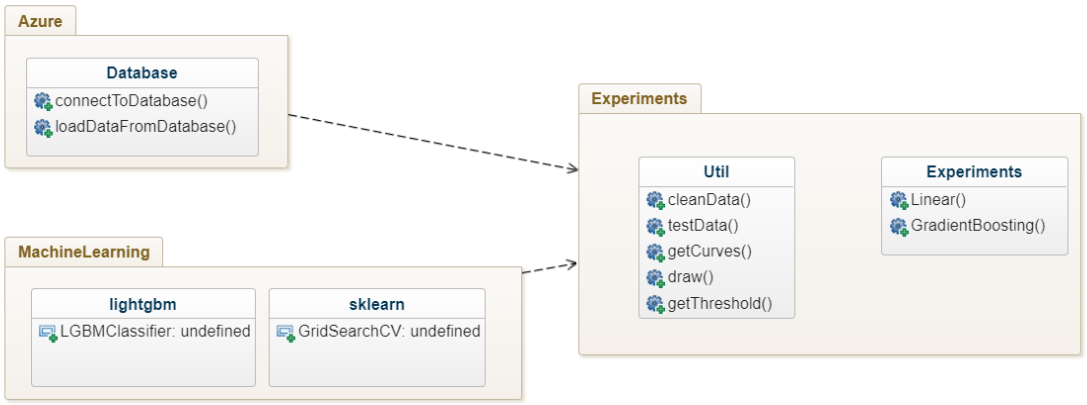

Рис. 7. Диаграмма компонентов системы

Пакет Azure объединяет функционал, осуществляющий подключение к рабочему пространству Azure, где хранятся данные. Пакет machine learning предоставляет реализацию алгоритмов машинного обучения и включает в себя сторонние библиотеки lightgmb и Sci–kit Learn. Пакет Experiments содержит компоненты, отвечающие за обучение, тестирование, сохранение моделей и непосредственно классификацию: класс Util предоставляет второстепенный функционал: очищение данных, отрисовку графиков. Класс Expirements реализуется обучение модели посредством градиентного бустинга, логистической регрессии.

В качестве реализации градиентного бустинга была выбрана библиотека LightGBM [14]. Для особенно больших наборов данных Легкий Градиентный Бустинг лучше остальных, так как требует меньше времени. Этот алгоритм основан на росте дерева по листьям, в отличие от других, которые работают по схеме поэтапного подхода. LightGBM добавляет

дерево вертикально, в то время как другой алгоритм добавляет деревья горизонтально. Это означает, что LightGBM добавляет дерево по листьям, в то время как другой алгоритм добавляет по уровням. Он выберет лист с максимальной потерей дельты для роста. При добавлении одного и того же листа, листовой алгоритм может уменьшить больше потерь, чем уровневый алгоритм.

Плюс LightGBM.

1) Более быстрая скорость обучения и высокая эффективность.

2) Более низкое использование памяти.

3) Лучшая точность.

Для каждого банка подбираются гиперпараметры, которые дают лучший precision. Для каждого банка будет обучаться модель с разным набором параметров. n_estimators – число деревьев, чем больше деревьев, тем лучше качество, но время настройки и работы RF также пропорционально увеличиваются. Часто при увеличении n_estimators качество на обучающей выборке повышается (может даже доходить до 100%), а качество на тесте выходит на асимптоту. Чем меньше глубина, тем быстрее строится и работает RF. При увеличении глубины резко возрастает качество на обучении, но и на контроле оно, как правило, увеличивается. Рекомендуется использовать максимальную глубину (кроме случаев, когда объектов слишком много и получаются очень глубокие деревья, построение которых занимает значительное время). При использовании неглубоких деревьев изменение параметров, связанных с ограничением числа объектов в листе и для деления, не приводит к значимому эффекту.

Максимальная глубина деревьев – max_depth, чем меньше глубина, тем быстрее строится и работает RF [15]. При увеличении глубины резко возрастает качество на обучении, но и на контроле оно, как правило, увеличивается. Рекомендуется использовать максимальную глубину (кроме случаев, когда объектов слишком много и получаются очень

глубокие деревья, построение которых занимает значительное время). При использовании неглубоких деревьев изменение параметров, связанных с ограничением числа объектов в листе и для деления, не приводит к значимому эффекту [16].

3.2. Анализ предиктивной способности признаков

Во многих (бизнес) случаях одинаково важно иметь не только точную, но и интерпретируемую модель. Зачастую, помимо желания узнать, каков прогноз нашей модели, мы также задаемся вопросом, почему именно этот максимум/минимум и какие характеристики наиболее важны при определении прогноза.

Знание важности функций, определяемой моделями машинного обучения, может принести вам пользу, например, несколькими способами.

1) Лучше понимая логику модели, вы можете не только проверить ее правильность, но и улучшить ее, сосредоточившись только на важных переменных.

2) Вышесказанное можно использовать для выбора переменных – вы можете удалить x переменных, которые не столь значительны и имеют схожие или лучшие показатели за гораздо более короткое время обучения.

3) В некоторых бизнес–случаях имеет смысл пожертвовать некоторой точностью ради интерпретируемости. Например, когда банк отклоняет заявку на кредит, он также должен иметь обоснование решения, которое также может быть представлено клиенту.

Давайте начнем с деревьев решений, чтобы построить некоторую интуицию. В деревьях решений каждый узел является условием того, как разделить значения в одном объекте, чтобы после разделения аналогичные значения зависимой переменной оказались в одном наборе. Условие основано на примесях, которые в случае задач классификации представляют собой примеси Джини/прирост информации (энтропия), а для деревьев регрессии – ее дисперсию. Поэтому при обучении дерева мы

можем вычислить, насколько каждая особенность способствует уменьшению взвешенной примеси. Feature_importances_ в Scikit–Learn основан на этой логике, но в случае Random Forest мы говорим об усреднении уменьшения примеси по деревьям.

Для интерпретирования предсказания моделей использовалась биб–лиотека SHAP [17]. С помощью этой библиотеки мы можем построить график важности параметров. В библиотеке SHAP для оценки важности признаков рассчитываются значения Шэпли. Для оценки важности признаков происходит оценка предсказаний модели с и без данного признака. Рассчитывается важность признака по формуле (6):

$$\varphi_i(p) = \sum_{S \subseteq \frac{N}{\{i\}}} \frac{|S|!(n-|S|-1)!}{n!} (p(S \cup \{i\}) - p(S), \quad (6)$$

Значение Шэпли для i–той фичи рассчитывается для каждого сэмпла данных на всех возможных комбинациях значений, затем полученные значения суммируются по модулю, и получается итоговая важность i–того значения.

Данные вычисления чрезвычайно затратны, поэтому используются различные алгоритмы оптимизации вычислений. На рисунке 8 можно увидеть результат работы алгоритма.

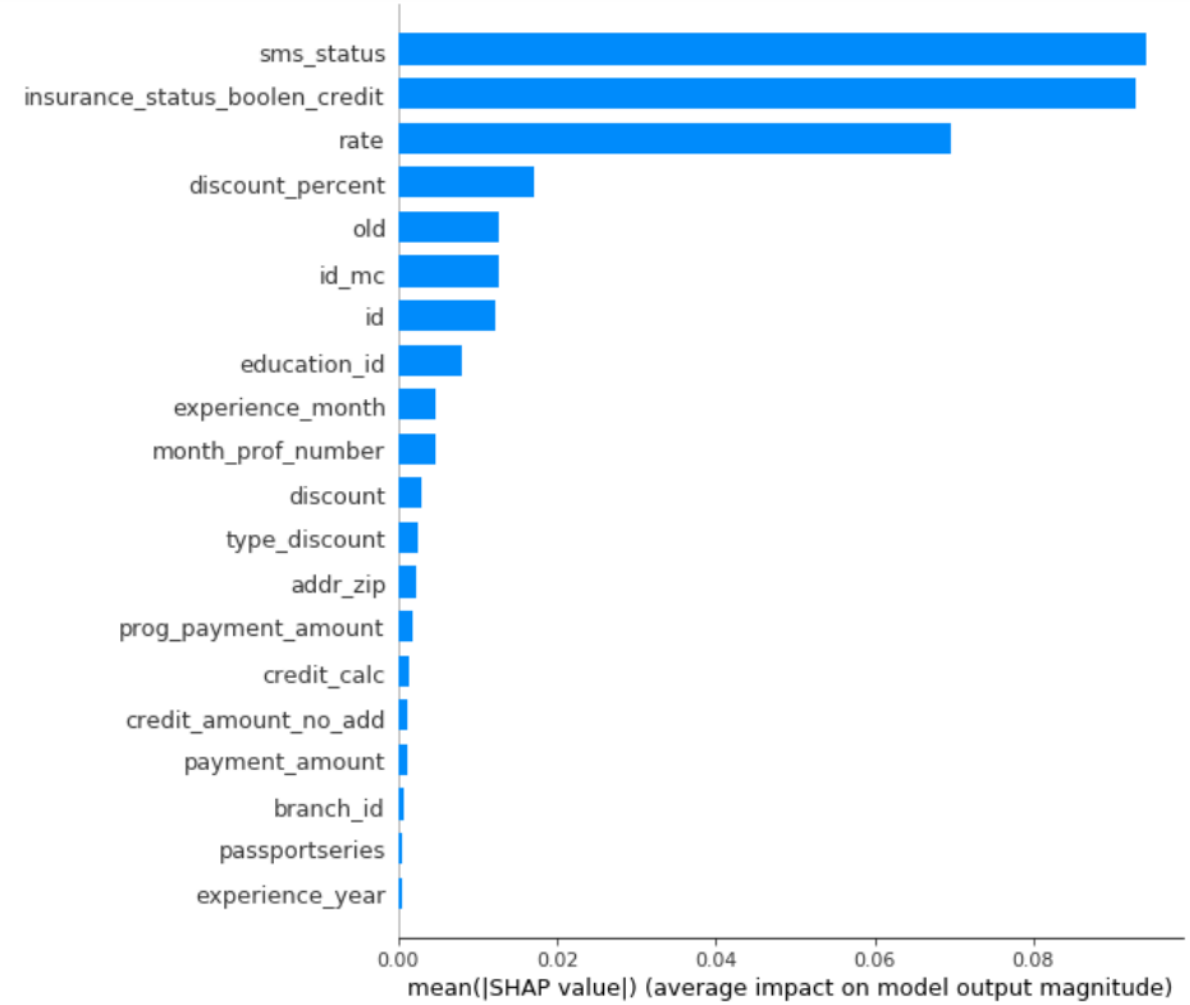

Рис. 8. Важность признаков по SHAP

Признаки, расположенные на графике вверху, вносят больший вклад в модель, чем нижние и, таким образом, обладают высокой прогнозирующей способностью. Это означает, что в модели нет необходимости использовать все 66 признаков, а только 6–12 признаков.

3.3. Выбор модели

Для получения несмещенной оценки качества алгоритма, был использован метод cross–validation [18]. Все алгоритмы были использованы с параметрами по умолчанию. Для логистической регрессии использовалась нормализация признаков. Для алгоритмов использовались признаки, полученные на предыдущем этапе.

Результат проведенных тестов приведен в таблице 1.

Табл. 1. Результаты тестирования алгоритмов классификации

Банк	**Градиентный бустинг**		**Логистическая регрессия**	
	Precision	**Recall**	**Precision**	**Recall**
Почта банк	0.84	0.69	0.72	0.52
Хоум Кредит	0.87	0.67	0.64	0.56
ОТП Банк	0.85	0.75	0.64	0.53
Альфа	0.77	0.7	0.62	0.60
РЕВО–ПЛЮС	0.86	0.73	0.54	0.61
МИГ	0.76	0.71	0.55	0.62

На основании результатов анализа алгоритмов классификации, в качестве основной модели системы классификации использовать градиентный бустинг над решающими деревьями.

3.4. Подбор гиперпараметров

В данной работе был проведен подход с обучением модели с разным набором гиперпараметров, для выявления лучшего результата. На рисунке 9 представлена функция, подбора гиперпараметров.

```
clf = LGBMClassifier()
params = {
    'n_estimators': [400, 500, 600, 800, 1000],
    'max_depth': [-1, 2,3,5],
    'learning_rate': [0.05,0.1,0.2]}
models = {}
for id, bank in id_to_bank.items():
    print(20*'=' + bank + 20*'=')
    models[id] = LGBMClassifier()
    test_model(models[id], sample[id], params)
```

Рис. 9. Код, отвечающий за подбор гиперпараметров

На рисунке 10 можно увидеть результаты лучших гиперпараметров для банка. Соответственно это проделывалось для каждого банка.

```
====================Банк Хоум Кредит====================
best params: {'learning_rate': 0.05, 'max_depth': 2, 'n_estimators': 400}
              precision    recall  f1-score   support

           0       0.80      0.95      0.87      1711
           1       0.66      0.29      0.40       576

    accuracy                           0.78      2287
   macro avg       0.73      0.62      0.64      2287
weighted avg       0.76      0.78      0.75      2287
```

Рис. 10. Пример подбора гиперпараметров для банка

3.5. Подбор пороговых значений

Подбор порогового значения для градиентного бустинга выполнялся с помощью функции. Суть задачи заключалась в обучении модели, которая способна предсказывать одобрение по информации о заказе клиенте с точностью 0,8–0,9, при охвате не менее 0,6. На рисунке 11 можно увидеть функцию, подбирающую пороговые значения.

```python
def get_curves(bank, clf, cols=None):
    precision_curve = []
    recall_curve = []
    thresholds = []
    preds = 0
    if cols is None:
        preds = clf.predict_proba(data[bank]['test']['X'])
    else:
        preds = clf.predict_proba(data[bank]['test']['X'][cols])
    for threshold in range(0,100,2):
        t = threshold/100
        thresholds.append(threshold)
        precision_curve.append(
            precision_score(data[bank]['test']['y'], (preds[:,1]>t).astype('int'))
        )
        recall_curve.append(
            recall_score(data[bank]['test']['y'], (preds[:,1]>t).astype('int'))
        )
    return thresholds, precision_curve, recall_curve
```

Рис. 11. Пример подбора гиперпараметров для банка

Функция возвращает 3 списка – пороговые значения с шагом в два, точность и полноту при этих пороговых значениях. На вход она получает название банка и модель.

Нам хочется отрегулировать модель таким способом, так чтобы точность была 0,8 – 0,9, при разумном снижении полноты. Для этого проведем эксперимент, обучив модели с подбором порогового значения и со стандартным значением. По умолчанию если значение порогового значение 0,5 и выше, то объект относится к первому классу (одобрен).

Это проделывается для каждого банка. Так же это позволяет поменять пороговое значение в скрипте, нежели загружать новую модель. Что сокращает время. Потому что менеджеры не смогут подобрать адекватные значения.

Но на основе их пожеланий (повысить точность или охват) можно внести корректировки. Чтобы определить качество модели для каждого банка необходимо обучить модель, с подобранным раннем набором гиперпараметров. Результат для одного банка можно увидеть на рисунке 12.

```
===Градиентный бустинг===

Используемые признаки:
['payment_amount_req', 'territory_id', 'old', 'rate', 'point_id', 'martial_status_id', 'operator_id', 'doc_status_simple_id',
'month_prof_number_year', 'insurance_life_req', 'discount']

С подбором уставок:
        post(thres=0.75)
                class=1         0.67    0.06
                class=0         0.73    0.99
        tnb(thres=0.76)
                class=1         0.73    0.18
                class=0         0.73    0.97
        mts(thres=0.5)
                class=1         0.84    0.78
                class=0         0.8     0.85

Без подбора уставок:
        post(thres=0.5)
                class=1         0.58    0.6
                class=0         0.84    0.83
        tnb(thres=0.5)
                class=1         0.65    0.71
                class=0         0.87    0.83
        mts(thres=0.5)
                class=1         0.84    0.78
                class=0         0.8     0.85
CPU times: user 2.12 s, sys: 221 ms, total: 2.35 s
Wall time: 4.02 s
```

Рис. 12. Результат пороговых значений

Полученный результат для всех банков приведен в таблице 2.

Табл. 2. Результаты подборов параметров

Банк	**Threshold**	**Precision**	**Recall**
Почта банк	0.58	0.87	0.7
Хоум Кредит	0.6	0.81	0.68
ОТП Банк	0.585	0.8	0.69
Альфа	0.59	0.82	0.67
РЕВО–ПЛЮС	0.596	0.79	0.7
МИГ	0.58	0.84	0.67

4. ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ

4.1. Регистрирование модели

После того как мы обучили нужную нам модель – ее необходимо зарегистрировать. Общая схема работы загрузки модели машинного обучения представлена на рисунке 13. Зарегистрированная модель – это логический контейнер для одного или нескольких файлов, составляющих модель. После регистрации файлов можно скачать или развернуть зарегистрированную модель и получить все зарегистрированные файлы.

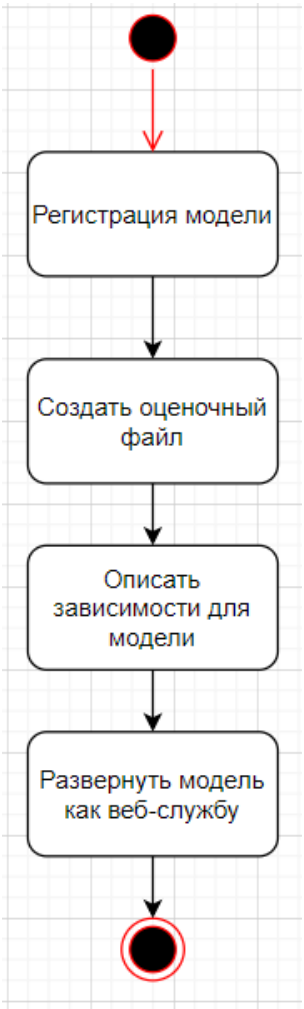

Рис. 13 Общая схема обновления модели

Поскольку файл модели может быть практически любым, нужно предоставить скрипт оценки, который может загрузить модель, а затем применить модель к новым данным. На рисунке 14 представлена функция, регистрации модели.

```
      model = Model.register(workspace=ws,
model_name='mts_insurance_add_rfc'
      model_path='./mts_insurance_add_rfc.pkl'
      model_framework=Model.Framework.SCIKITLEARN
      model_framework_version='0.20.3'
      resource_configuration=ResourceConfiguration(cpu=1,
memory_in_gb=1),
      description='Random Forest Classifier model to predict MTS bank
insurance add.',
      tags={'area': 'MTS insurance', 'type': 'Random Forest
Classifier'})
```

Рис. 14. Листинг функции, осуществляющей регистрацию модели

Этот файл скоринга представляет собой программу на Python, содержащую как минимум два метода init () и run (). Метод init () вызывается один раз при запуске развертывания, чтобы могли загрузить свою модель и любые другие необходимые объекты. Этот метод использует функцию get_model_path, чтобы найти зарегистрированную модель внутри контейнера Docker [19]. Метод run () вызывается интерактивно, когда веб–служба вызывается с одной или несколькими выборками данных для прогнозирования. Теперь, когда мы обучили набор моделей и определили прогон, содержащий лучшую модель, мы хотим развернуть модель для вывода в реальном времени. Процесс развертывания модели включает в себя.

1) Регистрация модели в вашем рабочем пространстве.

2) Создание файла оценки, содержащего методы init и run.

3) Создание файла зависимости среды с описанием пакетов, необходимых для вашего файла скоринга.

4) Развертывание модели и пакетов в качестве веб–службы.

Последний шаг к развертыванию вашего веб–сервиса – вызов Model.deploy(). Эта функция использует конфигурации развертывания и логического вывода, созданные выше, для выполнения следующих действий.

1) Создайте образ докера.

2) Развернуть в образ докера в экземпляр контейнера Azure.

3) Скопируйте файлы модели в экземпляр контейнера Azure.

4) Вызовите функцию init () в вашем файле скоринга.

5) Предоставить конечную точку HTTP для оценки вызовов.

Процесс развертывания модели включает в себя.

1) Рабочая область – рабочая область, содержащая сервис.

2) Name – уникальное имя, используемое для идентификации сервиса в рабочей области.

3) Models – массив моделей для развертывания в контейнере.

4) Inference_config – объект конфигурации, описывающий среду изображения.

5) Deploy_config – объект конфигурации, описывающий тип вычислений.

Теперь, когда ваш веб–сервис запущен, вы можете отправлять данные JSON непосредственно в сервис, используя метод run. Эта ячейка извлекает первый тестовый образец из исходного набора данных в JSON и затем отправляет его в службу. Представлен код отправки данных по разным банкам (рисунок 15). Необходимые параметры для модели мы задали раньше, поэтому достаточно отправить массив чисел.

```
input_data =
'{"data,"otp":[2399,5970,56936,216,39,25000,78,836584,836554,835740,1
7145147,17145147],"mts":[2399,5970,56936,216,39,25000,78,836584,83655
4,835740,17145147,17145147]}}'
      headers = {'Content-Type':'application/json'}
      api_key = service.get_keys()[0]
      headers = {'Content-Type':'application/json',
'Authorization':('Bearer '+ api_key)}
      resp = requests.post(service.scoring_uri, input_data,
headers=headers)
      print("POST to url", service.scoring_uri)
      print("prediction:", resp.text)
```

Рис. 15. Листинг функции, осуществляющей отправку запроса модели

В ответ от сервиса мы получим строку с оценкой возможности выдачи кредита для клиента в каждом банке. Результат можно увидеть на рисунке 16.

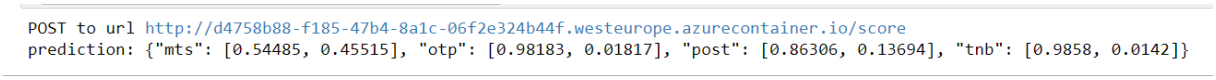

Рис. 16. Оценка вероятности выдачи кредита клиенту

4.2. Тестирование

Тестирование является неотъемлемой частью разработки современного программного обеспечения. Автоматические тесты позволяют разработчику быть уверенным, что программа работает как задумано. Успешное выполнение тестов является индикатором того, что внесенные в код изменения не повлияли на работоспособность программы. Провалившиеся тесты позволяют обнаружить, что в коде сделаны изменения, которые меняют его поведение, а исследование ошибки, которую выдает провалившийся тест, и сравнение ожидаемого результата с полученным даст возможность понять, где возникла ошибка [20].

С помощью стандартной библиотеки Python unittest были реализованы тесты, покрывающие базовый функционал системы. Данная библиотека позволяет создавать тесты в ООП стиле. Использование библиотеки подразумевает наследование от класса TestCase с переопределением методов.

Суть эксперимента заключается в том, что сначала данные по клиентам попадают в нашу модель, а затем только отправляться в банк. Результаты модели и ответы банков будут копиться в течение определенного периода времени (неделя/2 недели/месяц) и только после этого можно будет сделать вывод о том насколько точно модель предсказывает новые данные в реальных условиях. Только после такого тестирования и возможных правок и доработок модели можно будет использовать ее для предсказания уникальности одобрения до запроса в

банки, что сократит время ответа, а также позволит оператору/системе принимать решение о дополнительных действиях: продажа страховки, рекомендация выбора менее дорогих товаров, повышение процентной ставки.

Для того чтобы протестировать работу загруженной модели в сервисе Azure достаточно сделать веб запрос. Когда веб–сервис запущен, мы можем отправлять данные JSON непосредственно в сервис, используя метод run. Для мы будем сравнивать ответы банков с ответами нашей модели. На рисунке 17 показ пример запроса по 3–ем клиента из тестовой выборки.

```
import json

X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(x_df, y_df, test_size=0.3, random_state=68)

X_train_new = X_train_new.loc[:, ['prog_payment_amount', 'client_age', 'overpayment_amount', 'credit_calc']]
X_test_new = X_test_new.loc[:, ['prog_payment_amount', 'client_age', 'overpayment_amount', 'credit_calc']]

service = ws.webservices['ml-aci-svc-2']

# scrape the first row from the test set.
test_samples = json.dumps({"data": X_test_new[0:3].values.tolist()})
print(test_samples)
print(y_test_new[0:3])

#score on our service
service.run(input_data = test_samples)
```

```
{"data": [[1700, 38, 2339, 18060], [5721, 38, 22298, 115000], [1125, 40, 0, 26991]]}
426     1
1003    0
1981    0
Name: mts_insurance_add, dtype: int64

[1, 0, 0]
```

Рис. 17. Оценка вероятности выдачи кредита клиенту

Результаты тестирования системы на реальных данных представлено на таблице 3 и таблице 4.

Табл. 3. Результаты тестирования системы

Банк	**Всего заявок**	**Одобрено банком**	**Одобрено моделью**		**Одобрено моделью**	
			верно	**неверно**	**верно**	**неверно**
Почта банк	3777	2455	1473	323	957	1024

Продолжение табл. 3

Банк	Всего заявок	Одобрено банком	Одобрено моделью		Одобрено моделью	
			верно	неверно	верно	неверно
ОТП	3278	1967	1141	234	1367	537
Альфа	2688	1478	872	218	1016	582
Хоум Кредит	2415	1208	652	89	1202	472
РЕВО–ПЛЮС	450	203	115	22	242	71
МИГ	432	173	97	26	239	70

Табл. 4. Результаты тестирования системы.

Банк	Precision	Recall
Почта банк	0.82	0.59
ОТП Банк	0.83	0.68
Альфа	0.80	0.60
Хоум Кредит	0.88	0.58
РЕВО–ПЛЮС	0.84	0.62
МИГ	0.79	0.58

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы были решены следующие задачи.

1) Проведен анализ предметной области и обзор существующих решений.

2) Выбрано несколько подходов к решению проблемы.

3) Проведен сбор и очистка данных.

4) Проведены эксперименты по обучению различных моделей машинного обучения.

5) Реализована система классификации заемщиков кредитным агрегатором.

Целью данной работы было создание системы, позволяющей классифицировать заемщиков банков на основе методов машинного обучения. Для достижения этой цели был проведен анализ исходного процесса, собраны и очищены данные, выбран алгоритм обучения, обучены модели и развернуты как веб-сервисы.

По итогу выполнения работ была реализована система, позволяющая классифицировать заемщиков банков на основе методов машинного обучения. Проведенные эксперименты показали, что набор моделей градиентного бустинга дает наилучшую точность классификации, поэтому его применение более эффективно.

ЛИТЕРАТУРА

1. Скоринг (scoring). [Электронный ресурс] URL: https://www.banki.ru/wikibank/skoring/ (дата обращения: 26.04.2020).

2. Скоринг. [Электронный ресурс] URL: https://www.calltouch.ru/glossary/skoring (дата обращения: 26.04.2020).

3. Машинное обучение в микрофинансах: строим скоринговую модель для клиентов с пустой кредитной историей. [Электронный ресурс] URL: https://habr.com/ru/post/454574/ (дата обращения: 26.04.2020).

4. Скоринг как метод оценки кредитного риска. [Электронный ресурс] URL: https://www.cfin.ru/finanalysis/banks/scoring.shtml (дата обращения: 26.04.2020).

5. Analysis from 50+ papers on the Application of ML in Credit Lending. [Электронный ресурс] URL: https://towardsdatascience.com/my–analysis–from–50–papers–on–the–application–of–ml–in–credit–lending–b9b810a3f38 (дата обращения: 26.04.2020).

6. Метод опорных векторов (SVM) [Электронный ресурс] URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D1%8B%D1%85_%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2_(SVM) (дата обращения: 26.04.2020).

7. Credit Scoring using Least Squares Support Vector Machine based on data of Thai Financial Institutions. [Электронный ресурс] URL: https://ieeexplore.ieee.org/document/4195581 (дата обращения: 26.04.2020).

8. Credit Scoring: A Review on Support Vector Machines and Metaheuristic Approaches. [Электронный ресурс] URL: https://www.hindawi.com/journals/aor/2019/1974794/ (дата обращения: 26.04.2020).

9. A comparison of Gradient Boosting with Logistic Regression in Practical Cases. [Электронный ресурс] URL:

https://www.sas.com/content/dam/SAS/support/en/sas–global–forum–proceedings/2018/1857–2018.pdf/ (дата обращения: 26.04.2020).

10. Classification: ROC Curve and AUC. [Электронный ресурс] URL: https://developers.google.com/machine–learning/crash–course/classification/roc–and–auc.

11. Azure Microsoft. [Электронный ресурс] URL: https://azure.microsoft.com/ru–ru/ (дата обращения: 26.04.2020).

12. Azure Machine Learning. [Электронный ресурс] URL: https://azure.microsoft.com/ru–ru/services/machine–learning/ (дата обращения: 26.04.2020).

13. Математические методы обучения по прецедентам. [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron–ML–1 (дата обращения: 26.04.2020).

14. SHAP. [Электронный ресурс] URL: https://github.com/slundberg/shap (дата обращения: 26.04.2020).

15. AUC ROC. [Электронный ресурс] URL: https://dyakonov.org/2017/07/28/auc–roc–%D0%BF%D0%BB%D0%BE%D1%89%D0%B0%D0%B4%D1%8C–%D0%BF%D0%BE%D0%B4–%D0%BA%D1%80%D0%B8%D0%B2%D0%BE%D0%B9–%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA/ (дата обращения: 26.04.2020).

16. SHAP. [Электронный ресурс] URL: https://github.com/slundberg/shap (дата обращения: 26.04.2020).

17. Случайный лес (Random Forest). [Электронный ресурс] URL: https://dyakonov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9-%D0%BB%D0%B5%D1%81-random-forest/ (дата обращения: 26.04.2020).

18. Cross-validation: evaluating estimator performance. [Электронный ресурс] URL: https://scikit-learn.org/stable/modules/cross_validation.html.

19. Docker. [Электронный ресурс] URL: https://www.docker.com/.

20. Unit-testing. [Электронный ресурс] URL: https://habr.com/ru/post/169.