В соответствии с представленным критерием точку отсечения лучше всего выбрать на при вероятности 0,049. При таком выборе точки отсечения удается 74,2% правильных предсказаний для всех дефолтов и 100%-28,2%=71,8% правильных предсказаний для всех кредитополучателей погасивших задолженность.

В таких вопросах всегда лучше иметь наиболее пессимистичную оценку, потому посмотрим как будет вести себя полученная модель при выбранной точке отсечения для тестовой выборки более подробно. Таблица 3.1, по определению, аналогична таблице 2.3, но заполненная числами характеризующими полученную модель для тестовых данных.

Таблица 3.1 – TP, FP, TN, FN для полученной модели

Истинный класс	Предсказанный класс		Всего
	0	*1*	
0	42794	9556	52350
1	674	1067	1741
Всего	43468	10623	54091

Примечание – Источник: собственная разработка.

А ее аналог в относительных величинах – таблица 3.2.

Таблица 3.2 – TPR, FPR, TNR, FNR для полученной модели

Истинный класс	Предсказанный класс	
	0	*1*
0	81,74%	18,26%
1	38,71%	61,29%
Всего	80,36%	19,64%

Примечание – Источник: собственная разработка.

В этих таблицах нулем кодируется клиент без дефолта с единицей с дефолтом.

Как видно, на тестовых данных получается отличный классификатор клиентов без дефолта и удовлетворительный классификатор клиентов с дефолтом.

Заметим, что при выборе точки отсечения почти никогда не применяют подобные вычислительные методы, зачастую все зависит от политики банка и решение принимается ответственным управленцем. Тем не менее, разработчик модели может ориентироваться на такой метод, когда просят представить его мнение.

Данный раздел работы наполнен техническими деталями связанными с использованием библиотеки pytorch языка программирования python для реализации поставленной задачи. Первым делом были представлены базовый минимум для начала работы с библиотекой – описано как создавать и использовать классы моделей, оптимизаторы, и целевые фукнции. Через созданный класс «ResultModel» полноценную реализацию, во всех возможных уточнениях идентификационной формы получила модель рассматриваемая в

первой главе. В качестве оптимизатора было решено использовать алгоритм Adam. Получила полноценное описание целевая функция бинарной кросс энтропии.

После того, читатель начинает погружаться в реализацию алгоритма обратного распространения ошибки и то, какую роль в ней играют описанные выше элементы. Обоснована необходимость и описывается идея обучения по эпохам – способа обучения модели при котором каждый шаг алгоритма обратного распространения ошибки вычисление градиента производиться не на полном наборе данных, но на некоторой его части. В pytorch обучение по эпохам реализовано с использованием загрузчиков данных, процесс объявления и создания которых подробно описан в пункте 3.1.2.

Для обучения модели создан отдельный класс языка программирования python – «Model_trainer» (листинг 3.2). Использовать для реализации операции объектно-ориентированный подход сначала казалось не здравым решением, но при обучении приходиться оперировать многими переменными, потому в конце концов класс оказался более эргономичным нежели функция. Удобно в конструкторе указать относительно постоянные элементы: модель, оптимизатор и целевую функцию; а обучение вывести в отдельный метод который можно вызывать по любой потребности в обучении или «доучивании» модели со специфичными настройками.

Перед тем как приступить, наконец, к обучению модели мы привели еще три преобразований над данными: OHE, нормализация и разбиение на тестовую и тренировочную выборки.

Мы решили вырезать предварительные неудачные версии моделей, чаще всего просто не удавалось добиться установленного минимального значения в 0,7 центральной метрики качества модели – показателя AUC. Достаточно неплохие результаты показала модель с одним скрытым слоем и 113 нейронами в нем. На исследовании ее процесса обучения и попытки улучшить ее показатели были направлены все наши дальнейшие усилия.

При диагностике модели мы в основном обращали внимание на кривые обучения. В работе нет даже половины всех базовых конфигураций кривых обучения, тем не менее, те ситуации в которые попадали мы, тщательно описаны и предложены возможные пути выхода из них. Так если кривая заканчивается на стремительном спаде, следует увеличить число эпох или изменить критерий остановки алгоритма, если кривая обучения на некотором шаге теряет стабильность то следует искать возможность уменьшить шаг обучения, например, через уменьшение параметра learning rate и действовать наоборот в случае если кривая начала заметно замедляться в спуске.

Действуя таким образом нам удалось развить модель от 0,75 до 0,78 показателя AUC. Далее было уже сложно в ручную конфигурировать настройки learning rate потому мы решили включить в алгоритм обучения некоторый компонент, который мог бы полуавтоматически проводить настройку. В pytoch это можно сделать с использованием планировщика learning rate, возможность его использования, кстати, также заложена в класс «model_trainer». Мы

применяли достаточно простой планировщик понижающий learning rate каждую эпоху в установленное число раз. Но даже с таким простым правилом обучения следуют обращаться аккуратно так, при первой попытке использования планировщика, установив слишком высокую скорость понижения learning rate, мы даже ухудшили результаты по сравнению с моделями полученными без его использования.

В конце концов, нам удалось подобрать некоторые настройки, которые привели к появлению модели у которой показатель AUC составил 0,8, что считается более чем удовлетворительным результатом. Все дальнейшие попытки улучшить модель не увенчались успехом потому эту модель мы оставили как финальную.

Кроме *AUC* для классификационной модели разумно подсчитать долю правильных предсказаний для каждого из класса. При точке отсечения выбранной, опираясь на *KS* статистику, названные показатели модели можно увидеть в таблицах 3.1 и 3.2. Модель отлично справляется с выделением хорошего клиента определяя более 80% правильно, но несколько проседает с определением клиента с зафиксированным дефолтом, предсказывая 61%. Это обыденная ситуация в кредитном скоринге, опытные аналитики связывают это с тем, что в выборке этой предметной области всегда доминируют наблюдения без дефолта, а те, что вышли в дефолт нередко выходят по уникальным причинам без прослеживаемой статистической закономерности. В целом, показатели модели удовлетворительные. На этом считаем целесообразным закончить данную работу.

ЗАКЛЮЧЕНИЕ

По завершению исследования, приведенного в данной работе, мы пришли к моделям искусственных нейронных сетей, подготовили набор данных и построили модель соответствующей идентификационной формы. Далее выводы по каждому из разделов.

Первая глава начиналась с постановки задачи классификации, кратко: имея набор данных вида как в таблице 1.1 нужно сформировать правило позволяющее проводить классификацию наиболее точно в некотором, пока точно не определенном, смысле, то есть предсказывать элемент столбца Y (отклик) располагая только элементами столбцов X (предикторами, показателями, переменными).

Модель линейной регрессии оказалась плохим решением, так как она предполагает численный тип отклика а у нас он номинативный, из того вытекали два основные последствия: предсказания выходят за границы возможных вероятностей и переход от одного класса к другому происходит линейно, что осложняет их разделение по предсказанным вероятностям.

Для решения этой проблемы в логистической регрессии используется особая функция – логит, свойства которой позволяют избавиться от названных проблем. Однако логистическая регрессия остается линейным классификатором. Это происходит потому, что решение о отнесении к тому или иному классу принимается опираясь на некоторую точку отсечения – если предсказана большая вероятность считается, что исследуемый признак проявился, получается, мы выбираем некоторую линию уровня после которой считаем наблюдения принадлежащие к тому или иному классу. Несложно доказать, что линии уровня логит функции линейные, потому и правило классификации получается линейным. В работе предложены достаточна понятная визуализация на рисунках 1.3 и А.1.

Линейное правило бывает слишком простым, очень во многих случаях его недостаточно. На рисунке 1.4 показано как может справиться логистическая регрессия с нелинейным случаем. В самом деле, результат не слишком плохой но добавив в модель нелинейность можно добиться стопроцентной точности классификации.

Нелинейность может быть добавлена путем усложнения идентификационной формы модели. Мы предлагаем использовать кусочно-линейную функцию под логит функцией, таким образом «ломать» дискриминирующую прямую в нужных местах. Абсолютно аналогичные модели получатся при использовании идей искусственных нейронных сетей при включении логит нейрона в выходном слое и ReLU нейронов в скрытом слое. Как решает задачу с которой не справился алгоритм логистической регрессии подобная модель искусственной нейронной сети представлено на рисунке А.2.

По завершению первой главы мы коснулись вопроса оценки коэффициентов в нейронной сети. Популярным методом оценки коэффициентов является алгоритм обратного распространения ошибки. Формально, это тоже

градиентный спуск (метод нелинейной оптимизации), в удобной записи для работы с моделями типа нейронной сети.

Для построения модели был получен набор данных содержащий 247062 записи и 45 показателей. Сразу было решено исследовать типы входящих переменных, для того использовался созданный самостоятельно инструмент, позволяющий формировать таблицы, подобные тем, что представлены в приложении Г. Опираясь на информацию из этих таблиц, мы сразу сделали выводы относительно целесообразности использования некоторых столбцов. Так сразу опустили столбцы которые содержат слишком много уровней и проводить укрупнение не целесообразно. Другие столбцы пытались укрупнить. Создавали новые столбцы с, предположительно, лучшей классифицирующей способностью и т.д.

Особый интерес представляет процедура, которую мы использовали при отборе показателей в модель. Мы оперлись на показатель *AUC* – популярный способ оценить качество готовой модели. Он тем больше, чем сильнее различаются распределения по предсказанным вероятностям для конкурирующих классов. Для того, чтобы использовать его для оценки классифицирующей способности отдельных предикторов, мы рассмотрели однофакторные классификаторы вида (2.5) и для них вычисляли *AUC*. В данном случае даже важнее *KS* статистика, а точнее, соответствующий тест Колмогорова-Смирнова для двух выборок. В тесте мы сравнивали распределения хороших и плохих клиентов вдоль изучаемого предиктора и не допускали показатель в модель в том случае, если удавалось подтвердить нулевую гипотезу об однородности выборок. Это делалось автоматически благодаря созданному инструменту, результатом его работы стали таблицы, подобные таблицам приложения И.

Заканчивается второй раздел описанием способов избавления от пропусков. Для номинативной переменной пропуск воспринимался как отдельный уровень. Для численной применялось при основных подхода: перекодировка в по принципу «есть данные/нет данных», перекодировка с использованием точки соответствующей *KS* статистике в качестве разделителя классов и заполнение пропусков нулем. Полученные таким образом переменные сравнивались (в том числе и с исходной переменной) и из них выбиралась наилучшая соответствующая здравому смыслу.

Для построения модели в работе используется библиотека для создания нейронных сетей pytorch. Первым делом описываются самые базовые принципы: оптимизатор, целевая функция и класс модели. Оптимизатор используется Adam. В качестве целевой функции задействована бинарная кросс-энтропия. Класс для описания модели представлен на листинге К.1. Для реализации базового алгоритма обратного распространения ошибки этого, кончено достаточно, но на практике сложно получить качественные оценки параметров используя только их.

Далее мы описываем более продвинутые (но по прежнему базовые) техники используемые для обучения моделей. Это разбиение выборки на тренировочные

и тестовые данные и использование обучения по эпохам. Первая техника позволяет избежать феномена переобучения, вторая обойти больше локальных минимумов в задаче оптимизации.

Используя только эти подходы нам удалось обучить модель с одним скрытым слоем и 113 нейронами в нем с удовлетворительной характеристикой в 0,78 *AUC*. Но исследуя кривые обучения, удалось сделать выводы о возможности улучшения модели путем динамического изменения learning rate. Для того использовались особые объекты – планировщики learning rate. С их использованием удалось улучшить модель.

И так, финальная модель достигла 0,80 показателя *AUC* на тестовых данных и, при выборе точки отсечения опираясь на значение соответствующее *KS* статистике, было достигнуто 61% правильной классификации наблюдений с дефолтом и 81% правильной классификации наблюдений без дефолта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах; пер. с англ. А.А. Слинкина. – М.: ДМК Пресс, 2015. – 400 с.
2. Введение в статистическое обучение с примерами на языке R / Г. Джеймс [и др.]; пер с англ. С.Э. Мастицкого. – М.: МДК Пресс, 2017. – 456 с.
3. Hosmer, D.W. Applied Logistic Regression / D.W. Hosmer, J.S. Lemeshow // University of Massachusetts. – 2-nd ed. – 2000. – 376 p.
4. Микелуччи У. Прикладное глубокое обучение. Подход к пониманию глубоких нейронных сетей на основе метода кейсов / У. Микелуччи; пер. с англ. – СПб: БХВ-Петербург, 2020. – 368 с.
5. Грас Дж. Data Science. Наука о данных с нуля / Дж. Грас; Пер. с англ. – СПб.: БХВ-Петербург, 2017. – 336с.
6. Финансовая отчетность в соответствии с НСФО/ Белинвестбанк [Электронный ресурс]. – Режим доступа: https://belinvestbank.by/about-bank/finance-statistic/nsfo/. – Дата доступа: 06.03.2022.
7. Pandas, main page [Electronic resource]. – 02.04.2022. – Mode of access: https://pandas.pydata.org/. – Date of access: 10.04.2022.
8. Numpy, main page [Electronic resource]. – 2022. – Mode of access: https://numpy.org/. – Date of access: 10.04.2022.
9. Scipy, main page [Electronic resource]. – 05.02.2022. – Mode of access: https://scipy.org/. – Date of access: 10.04.2022.
10. Scikit-learn, machine learning in python [Electronic resuorce]. – 2021. – Mode of access: https://scikit-learn.org/stable/. – Date of access: 10.04.2022.
11. PyTorch, main page [Electronic resource]. – Mode of access: https://pytorch.org/. – Date of access: 10.04.2022.
12. Виды экономической деятельности, общегосударственный классификатор Республики Беларусь. – Минск, 2011. – 364 с.
13. Выброс (статистика) / Википедия, свободная энциклопедия [Электронный ресурс]. – 2021. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0). – Дата доступа: 10.04.2022.
14. Lopes, R.H.C. The two-dimentional Kolmogorov-Smirnov test / R.H.C. Lopes, I. Reid, P.R. Hobson // XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research. – 2007. – Amsterdam, Netherlands.
15. Loss functions/ PYTORCH DOCUMENTATION [Electronic resource]. – Mode of access: https://pytorch.org/docs/stable/nn.html#loss-functions. – Date of access: 16.04.2022.
16. TORCH.OPTIM/ PYTORCH DOCUMENTATION [Electronic resource]. – Mode of access: https://pytorch.org/docs/stable/optim.html. – Date of access: 16.04.2022.

17. Пойнтер, Я. Программируем с PyTorch: Создание приложений глубокого обучения / Я. Пойнтер. – СПб.: Питер, 2020. – 256 с.

18. Goboy, D. Understanding binary cross-entropy/ log loss: a visual explanation [Electronic resource]. – Mode of access: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a. – Date of access: 23.04.2022.

19. Хендрик, Б. Машинное обучение / Б. Хендрик, Дж. Ричардс, М. Феверолф. – СПб.: Питер, 2017. – 336 с.

20. Тарик, Р. Создаем нейронную сеть.: Пер. с англ / Р. Тарик. – СПб.: ООО «Альфа-книга», 2017. – 272 с.

21. Novak, G. Building a One Hot Ecoding Layer with TensorFrow[Electronic resource] / Towards Data Science. – 07.06.2020. – Mode of access: https://towardsdatascience.com/building-a-one-hot-encoding-layer-with-tensorflow-f907d686bf39. – Date of access: 20.04.2022.

22. Шолле, Ф. Глубокое обучение на python / Ф. Шолее. – СПб.: Питер, 2018. – 400 с.