МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УО «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра математических методов в экономике

Специальность «Оптимальное планирование и управление в экономике»

Допущена к защите

Заведующий кафедрой

д-р экон. наук, доц.

­­­­\_\_\_\_\_\_\_\_ Г.О. Читая

31.05.2020

**ДИПЛОМНАЯ РАБОТА**

на тему: **Разработка нейронных сетей и их использование при принятии решений о выдаче кредита (на примере ОАО «Белинвестбанк»)**

|  |  |  |
| --- | --- | --- |
| Студент  ФЦЭ, 4-й курс, ДКК-1 |  | Ф.А. Кобак |
|  |  |  |
| Руководитель  кнад. экон. наук,  профессор |  | Э.М. Аксень |
|  |  |  |
| Нормоконтролер |  | И.В. Денисейко |

МИНСК 2022

**РЕФЕРАТ**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc99060264)

[1. Теория моделей искусственной нейронной сети при задаче классификации 5](#_Toc99060265)

[ЗАКЛЮЧЕНИЕ 6](#_Toc99060266)

# ВВЕДЕНИЕ

# Теория моделей искусственной нейронной сети в задаче классификации

# 1.2 Постановка задачи

Рассматриваемые в этой работе методы, на ряду с некоторыми другими, предназначены для решения задач классификации. Задачей классификации называется задача в которой требуется определить способ отнесения некоторых объектов к некоторым группам (классам).

Такие задачи разбиваются на два вида – те для которых известно число классов и те для которых число классов заранее неизвестно. В этой работе рассматриваются только задачи первого вида.

Особенностью описного типа задач является наличие предсказываемого фактора Y. То есть, для каждого объекта из изучаемой совокупности имеется признак который может принимать значения :

|  |  |
| --- | --- |
|  | (1.1) |

Предположим, имеются наблюдения за некоторым явлением или процессом которые можно представить подобно таблице 1.1.

**Таблица 1.1 – Исходная система данных**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | *Y* | *X*1 | *X*2 | … | *X*m |
| 1 | *y*1 | *x*11 | *x*12 | … | *x*1*m* |
| 2 | *y*2 | *x*21 | *x*22 | … | *x*2*m* |
| … | … | … | … | … | … |
| *n* | *y*n | *xn*1 | *xn2* | … | *xnm* |

Примечание – Источник: собственная разработка.

Где по строкам расположились наблюдения, а по столбцам некоторые переменные для этих наблюдений. На основе этих данных формируется правило, которое позволяет, получив произвольный набор значении переменных , наиболее точно, в некотором смысле, предсказать значения . В литературе это правило нередко обозначается так:

Это очень распространённая задача для прикладной статистики и машинного обучения, встречающаяся во многих сферах жизни: задача определения диагноза по симптомам и анализам; разбиение электронной почты на действительную и спам; задачи распознавания рукописного текста и множество других.

Итак, была получена задача, в которой имеется набор наблюдений с откликом *Y*, аналогичная задача решается классическим регрессионным анализом, разница лишь в типе отклика – для обычной регрессионной модели он численная переменная, в нашем же случае номинативная (категориальная).

Номинативной называют переменную, для которой не определены ни порядок, ни шкала[1 с.316]. Под отсудившем шкалы понимается, что мы не можем как-либо определить расстояние между двумя различными значениями переменной, например, если предсказывается факт возврата или невозврата кредитополучателем задолженности, нет никакой возможности показать на сколько возврат «выше» или «ниже» невозврата. Понятие порядка переменной будет более подробно раскрыто в третьем подразделе, когда речь пойдет о упорядоченной модели логистической регрессии.

Описанная разница в типе отклика и порождает непригодность использования классической регрессионной модели.

Рассмотрим задачу бинарной классификации – отклик может принимать лишь два значения (*K=2*), для простоты обозначим события (1.1) каким-либо числами, обычно используется:

После оценивания будет получено регрессионное уравнение следующего вида:

где – оценка коэффициента регрессии;

– оценка свободного члена;

– случайная ошибка.

Отдельного обсуждения достойна переменная , она представляет собой оценку вероятности того, что исследуемый объект принадлежит к классу советующему числу 1. Тут возникает первая проблема такого подхода – нет никаких оснований чтобы , что в корне неверно для понятия вероятности. Наглядная иллюстрация этой проблемы представлена на рисунке 1.1 слева – две выборки распределены вдоль некоторой переменной x, притом для тех для которых , значения *х* заметно выше, построив регрессионную прямую наглядно убеждаемся в том, что ряд наблюдений получают оценки вероятности за границами нуля и единицы.



**Рисунок 1.1 – Оценки вероятностей через линейную регрессионную модель**

Примечание ­­­– Источник: собственная разработка.

На рисунке 1.1 справа наглядно представлена еще одна проблема описанного подхода – к той же выборке, которая обсуждалась выше, был добавлен ряд наблюдений с уровнем исследуемого фактора соответствующим , с заметным смещением в большую сторону по фактору *х*. Регрессионная прямая в таком случае сместилась и заметно хуже предсказывает вероятности для старых наблюдений – вся группа советующая , получила оценки вероятностей того, что они принадлежат группе , в районе 0,4, что вообще говоря достаточно плохо для такого простого примера. Пример того как с аналогичной задачей справиться логистическая регрессия будет представлен в следующем подразделе.

Более того, такая модель плохо обобщается для случая, когда требуется решение для задачи не бинарной классификации. [2 c.145]

Наличие всех описанных проблем при использовании регрессионного анализа для моделирования процесса с номинативным откликом и вызвало потребность в разработке специальных методов классификации, одним из которых является логистическая регрессия, рассматриваемая в следующем подразделе.

# 1.2 Модель логистической регрессии – линейный классификатор

Отталкиваясь от того, что было сказано в прошлом разделе, проведем ряд рассуждений, которые приведут к модели логистической регрессии, для задачи бинарной классификации.

Обозначим вероятность того, что исследуемый признак примет значение как . Одной из ключевых проблем в данном случае является то что , в то время как отклик в модели регрессионного анализа принимает значение .

Введём понятие шанса события, шансом появления некоторого события называется отношение вероятности появления этого события к вероятности появления любого другого совместного события. В нашем случае справедливо:

Несложно показать, что . Пользуясь свойствами логарифма получим, что , а такая переменная уже хороший кандидат, для того чтобы быть описанной методом линейной регрессии. Таким образом идея логистической регрессии предлагает предсказывать не вероятность того что , а логарифм отношения шансов этого события. Логарифм отношения вероятностей в дальнейшем будет называть логит функцией.

Для краткости последующих записей обозначим:

Получим аналитическую запись рассматриваемой модели. Запишем теоретическую линейную модель предсказывающую логарифм отношения шансов:

Имея значение логарифма отношения шансов легко получить искомую вероятность. Сначала проведем потенцирование рассматриваемого выражения:

Используя свойства натурального логарифма, получим:

Решив это уравнение относительно , получим общую запись модели логистической регрессии:

|  |  |
| --- | --- |
|  | (1.2) |

Выражение (1.2) и есть модель логистической регрессии для случая бинарной классификации[3 c.32]. Функция, лежащая в основании этой модели, соответствует функции логистического распределения и имеет два ключевых полезных для рассматриваемой задачи свойства: во-первых, она принимает значения в диапазоне от нуля до единицы, во-вторых она принадлежит к классу сигмовидных функций, то есть это гладкая, возрастающая функция, имеющая на графике форму буквы «S». В литературе распространено название – сигма функция или сигмоида и в общем она записывается так:

Вернемся к примеру, из предыдущего раздела и посмотрим, как логистическая регрессия справиться с поставленной задачей. На рисунке 1.2 синими точками по-прежнему обозначены сгенерированные наблюдения, красной линией теперь обозначаются оценки вероятностей для различных значений предиктора *х*, полученные по модели подобной (1.2). Как видно обе обозначенные проблемы решены, предсказания лежат строго в пределах от нуля до единицы, и смещенная выборка почти не влияет на качество модели.



**Рисунок 1.2 – Оценки вероятностей через логистическую регрессионную модель**

Примечание ­­­– Источник: собственная разработка.

Рассмотрим более обобщённый вариант – когда предсказываемая переменная не бинарна, а имеет более двух уровней. Такую модель принято называть мультиномиальной логистической регрессией.

В данном случае предсказываемый фактор будет закодирован, так:

В данном случае нам понадобиться *K*-1 логит функции. Кроме того, надо выбрать базовый уровень выходной переменной, пусть, не нарушая общности, это будет *Y*=0. Для такого случая логиты можно записать так:

|  |  |
| --- | --- |
|  | (1.3) |

Притом, каждый из них будет описываться предикторами в соответствии со следующим правилом:

Чтобы получить в данном случае запись, подобную выражению (1.2) для биномиальной регрессии, потребуется поработать с системой (1.3). Для начала проведем потенцирование каждого уравнения системы:

|  |  |
| --- | --- |
|  | (1.4) |

Теперь из одного уравнения выразим , пусть, не нарушая общности, это будет первое уравнение:

Используя определение полной вероятности, получим:

|  |  |
| --- | --- |
|  | (1.5) |

Из оставшихся уравнений системы (1.3) получим:

|  |  |
| --- | --- |
| . | (1.6) |

Перенеся левую часть выражения (1.5) приведя и общему знаменателю, положив его неравным и вынеся за скобки, получим:

Выражая от сюда получим:

|  |  |  |
| --- | --- | --- |
|  |  | (1.7) |

Вообще говоря, выражение (1.6) будет справедливо и при *k*=1, потому подставляя туда (1.7), легко получим вероятности появления других уровней исследуемого признака:

|  |  |  |
| --- | --- | --- |
|  |  | (1.8) |

Выражения (1.7) и (1.8) – самая общая запись логистической регрессии. Для удобства последующих записей, предполагая что , запишем одной формулой:

|  |  |  |
| --- | --- | --- |
|  |  | (1.9) |

Сразу повторюсь, для заострения внимания – вероятность проявления каждого возможного уровня отклика имеет свою функцию от параметров и входных данных . Заметим, что оценки коэффициентов и , получают методом максимального правдоподобия.

Теперь поговорим о том, почему методы машинного обучения в задачах классификации не остановились на логистической регрессии. В литературе можно встретить утверждение: «модель логистической регрессии – линейный классификатор». На первый взгляд может показаться, что это утверждение неверно, ведь выражение (1.2) никак не назвать линейным входного набора данных и сигмоида на рисунке 1.2 в целом кривая. Дело тут кроется в принципе принятия решения о отнесении наблюдения к тому или иному классу. Данное явление лучше всего рассматривать на примере двумерной задачи.

На рисунке 1.3 представлена диаграмма рассеяния опять же сгенерированных случайных данных. Данные двумерные и разделяются на два класса.



**Рисунок 1.3 – Диаграмма рассеяния двумерных классифицированных данных**

Примечание ­­­– Источник: собственная разработка.

По прежнему, нет никакой сложности в том, чтобы провести вручную линию с полной точностью отделяющую один класс от другого (хотя, в отличии от одномерно примера, уже понадобиться знания аналитической геометрии чтобы записать классифицирующее правило). Однако, в целях изучения метода, взглянем на то как с этой задачей справиться логистическая регрессия. На рисунке А.1 двумерный аналог рисунка 1.2 – соответствующая рассматриваемому примеру двумерная сигмоида. На нее нанесены наблюдения с рисунка 1.3.

Из рисунка понятно, что можно подобрать такую высоту что бы точки принадлежащие отдельным классам были разделены, а соответствующая линия уровня «упадет» на рисунок 1.3 так что идеально разделит классы и в отрыве от высоты сигмоиды. Покажем, что такая линия уровня будет линейной – она возникает при сигмоиде равной :

|  |  |  |
| --- | --- | --- |
|  |  | (1.10) |

Требуется разрешить уравнение относительно выражения под сигмоидой. Такую операцию мы уже продевали раньше, только наоборот, потому просто запишем результат:

Заметив, что правая часть выражения константа, скажем, что разделяющая линия уровня линейна.

Тут и рождается «почва» для дальнейшей эволюции методов классификации связанных с логит функцией. Как и ранее рассмотрим пример с которым данный метод справится плохо – рисунок 1.4.



**Рисунок 1.4 – Диаграмма рассеяния двумерных классифицированных данных при нелинейном принципе классификации**

Примечание ­­­– Источник: собственная разработка.

Слева все та-же диаграмма рассеяния, только несколько изменены принципы классификации. По по-прежнему можно вручную записать правило разделяющее классы, только в этот раз это будет не единственные уравнение прямой, но кусочно-линейная функция.

На этих данных была построена модель логистической регрессии. Выбрав по принципу наибольшей разности между распределениями классов по (этот принцип будет раскрыт в главе 3 на практике) и использовав (1.10) мы получили уравнение описывающее дискриминирующую прямую, так же нанесенную на график.

На рисунке справа более броско выделены ошибки модели. Как видно модель логистической регрессии, как и любая другая линейная модель с таким случаем справляется не идеально (хотя задача достаточно простая). Решение заключается в том, чтобы включить в модель некоторую нелинейность.

Этот подраздел посвящен логистической регрессии – модели которая не является целевой для данной работы, однако идеи в ней лежащие важны для полного раскрытия темы. Описаны плюсы по сравнению с линейной регрессией и выведены формулы позволяющие записать модель аналитически. Особо важной для дальнейшего повествования является логит функция (сигмоида, обратная функция логистического распределения). Показано на примере, почему логистическая регрессия является линейным классификатором и описаны причины дальнейшей эволюции методов классификации.

# 1.3 Модель искусственной нейронной сети в классификации

Как и любая другая математическая модель нейронная сеть может быть записана как система уравнений, но куда проще и понятнее рассматривать её как своеобразный граф по ребрам которого «текут» данные и преобразуются в узлах.

Базовой единицей выступает нейрон – узел графа. Раскроем его сруктуру. На рисунке 1.5 схематично представлен искусственный нейрон.



**Рисунок 1.5 – Схема искусственного нейрона**

Примечание ­­­– Источник: собственная разработка на основе [4 с.52].

Стрелки входящие в нейрон называют входными активациями. По сути просто число для отдельного примера с которым работает модель. Функция называется сумматорной, она отвечает за восприятие нейроном входных активаций. Функция – активационная (придаточная функция) отвечает за формирование выходной активации. Обозначение на описываемой диаграмме представляет выходную активацию нейрона.

Фактически нет ограничений на вид сумматорной и активационной функций, но в качестве сумматорной функции как правило используется просто линейная комбинация входных активаций:

|  |  |  |
| --- | --- | --- |
|  |  | (1.11) |

где – *i-я* входная актвация;

– вес *i-й* входной активации;

­– свободный член.

В данной работе не рассматривается моделей с сумматорной функцией другого вида, потому, впереть, при любом упоминании сумматорной функции имеется ввиду выражение (1.11). Давайте, для краткости записи за таким выражением по умолчанию, просто закрепим обозначение . А рисунок 1.5 примет вид как на рисунке 1.6.



**Рисунок 1.6 – Схема искусственного нейрона при линейной сумматорной функции**

Примечание ­­­– Источник: собственная разработка.

Даже в базовой литературе в области нейронных сетей разнообразие активационных функций куда шире. Можно сказать, что в случае схемы 1.6 вид нейрона полностью определяется видом его активационной функции.

В этой работе используются лишь два вида нейронов: сигмоидальный и ReLU.

С активационной функцией сигмоидального нейрона мы уже встречались раньше, это ничто иное как обычная сигмоида:

Опишем свойства сигмоидального нейрона которые для нас несут особую важность

ReLU (Rectified linear unit) нейрон имеет активационную функцию вида:

|  |  |
| --- | --- |
|  | (1.12) |

Теперь о важных свойствах ReLU нейрона.

Набравшись некоторой теории касательно аналитической записи отдельных нейронов и важных их свойств перейдем к рассмотрению того как нейроны объединяют в нейронные сети.

Выделяют ряд нейросетевых архитектур, но самая простая – архитектура прямого распространения. Слоем нейронной сети, в контексте сетей прямого распространения, будем называть множество нейронов одного типа, получающих информацию только от нейронов предыдущего слоя и передающих информацию только нейронам следующего слоя. Слои выстраиваются один за другим и формируют цепочку преобразований входных данных. Далее любая сеть о которой пойдет речь будет предполагаться сетью с архитектурой прямого распространения.

В общем такая нейросетевая архитектура может быть представлена в виде рисунка 1.7:



**Рисунок 1.7 – Обобщённая нейросетевая архитектура**

Примечание ­­­– Источник: собственная разработка на основе [4 c. 95].

Входной слой (нулевой слой, слой с номером 0) это даже не в полной мере нейроны – это представление на рисунке того как в модель попадают входные данные. Все последующие слои с номерами от *1-го* до *(L-1)-го* называют внутренними слоями нейронной сети. Активации выходного (слой с номером *L*) слоя представляют собой предсказания нейронной для данных заявленных во входном слое.

Более детально рассмотрим взаимодействие соседних слоев сети в терминах введенных выше. Обозначим – вес выходной активации *i-го* нейрона *(l-1)-го* слоя в суммарной функции *j-го* нейрона *l-го* слоя. В этом обозначении легко запутаться, потому при необходимости можно посматривать на рисунок 1.8:



**Рисунок 1.8 – Обозначения связывания соседних слоев**

Примечание ­­­– Источник: собственная разработка.

В целом, процесс идентификации нейронной сети и заключается в выборе последовательности слоёв нейронов различных видов и числа нейронов входящих в каждый из слоев. Не возникает сомнений, что архитектура искусственной нейронной сети может принимать очень разный вид, особенно для различных задач, но даже одна и та-же задача может иметь несколько обоснованных архитектур.

В этой работе будет рассмотрена архитектура, которую можно рассматривать как развитие идеи логистической регрессии, перейдем к ее описанию.

В предыдущем разделе было показано, почему логистическая регрессия является линейным классификатором и обоснована необходимость добавления некоторой нелинейности в названную модель.

Возвращаясь к сигмоидальному нейрону получается, что модель логистической регрессии может быть представлена как нейронная сеть без скрытого слоя. Для биномиальной логистической регрессии такая архитектура представлена на рисунке Б.1. Для перехода к мультиномиальной достаточно добавить в выходной слой столько нейронов сколько имеется классов. Можно дорисовать мультиномиальную

Основная идея рассматриваемой в этой работе архитектуры состоит в том, чтобы добавить в модель скрытые слои содержащие ReLU нейроны. Схематично такая архитектура представлена на рисунке Б.2. Получится, что если выразить сумматорную функцию выходного слоя через веса соединяющие различные слои и активации входного слоя, то под сигмоидой станет кусочно-линейная функция, что обеспечит некоторую нелинейность при принятии решения.

Для простоты рассмотрения механизмов запрятанных в этой модели вернемся к примеру из предыдущего раздела с которым логистическая регрессия справилась плохо. На рисунке 1.9 представлена та-же задача, пока сконцентрируемся на графике слева.



**Рисунок 1.9 – Решение задачи с помощью нейронной сети**

Примечание ­­­– Источник: собственная разработка.

Теперь на график нанесены линии соответствующие уравнениям которые я положил в принцип распределения по классам. Заметим, что таких линии всего две. Забегая вперед, скажем, что для решения такой задачи, достаточно нейронной сети с одним скрытым слоем и всего двумя нейронами в нем, как на рисунке 1.10.

Распишем аналитически, эту модель. Сигмоиду которая является последним преобразованием этой модели подробно обсудили выше, потому сразу начнем с записи сумматорной функции выходного слоя .

где – значение сумматорной функции *i-го* нейрона *j-го* слоя;

­– свободный член *L-го* слоя.



**Рисунок 1.10 – Нейронная сеть для кусочно-линейной классификации**

Примечание ­­­– Источник: собственная разработка.

Используя определение ReLU функции (1.12) можно последнюю формулу разложить на четыре случая:

|  |  |
| --- | --- |
|  | (1.13) |

Для решеная задачи классификации будет хорошо если каждая область по штриховке на рисунке на рисунке получит свой вид решающего правила – получиться так, что, например, область синих точек отделенная обеими дискриминирующими линиями будет получать вероятности принадлежности ко второму классу по своему отдельному уравнению которое сформируется так чтобы дать наименьшие вероятности. Аналогично, но менее однозначно, вероятности будут формироваться для областей с единственной штриховкой. Для области без штриховки, очевидно, правило должно выстроиться так, чтобы давать наибольшие вероятности.

Зная законы по которым формировалась выборка из примера можно согласовать веса первого слоя так, чтобы каждому случаю из формулы (1.13) соответствовала своя область рисунка. Это можно сделать несколькими способами, но пусть незаштрихованной области соответствует первый первый, области с наклоненной влево штриховкой второй, вправо – третий и области с двумя штриховками остаётся четвертый.

Покажем, как провести данное согласование для такого рисунка, хотя в общем это может выглядеть по другому – все зависит от того как проведены дискриминирующие линии.

На легенде рисунка обозначены уравнения дискриминирующих линий, начнем с первого. В данном случае области с наклонённой влево штриховкой соответствует неравенство:

Потому присвоив весам нейронной сети , и свободному члену первого нейрона первого слоя , получим, что второй случай формулы (1.13) действительно соответствует области с наклоненной влево штриховкой.

Области с наклонённой вправо штриховкой соответствует неравенство:

Потому присвоив весам нейронной сети , и свободному члену второго нейрона первого слоя , получим, что третий случай формулы (1.13) действительно соответствует области с наклоненной вправо штриховкой.

При описанном выше распределении весов области с двумя штриховками автоматически будет соответствовать последний случай формулы (1.13). Так получиться, что первый слой идеально разделит область наблюденных данных на четыре части, что обеспечит такую ситуацию, что сигнал в выходной нейрон будут подавать только наблюдения обозначенные синим.

Выше было обозначено, что в рамках этого примера мы будем моделью оценивать вероятность того, что конкретное наблюдение принадлежит к второму классу обозначенному красным. Для того, чтобы оценки вероятностей области без штриховки были выше нежели любые другие, можно весам второго слоя присвоить любые отрицательные значения – положительный сигнал выходящий из первого слоя в результате попадания в модель любого «синего» наблюдения будет помножаться на отрицательное число и занижать сумматорную функцию и, как следствие, занижать оценки вероятностей первого класса, чего мы и добивались. В данном случае свободный член сумматорной функции выходного нейрона может принимать любое значение – все равно механизм описанный выше будет занижать оценки вероятностей для любого наблюдения из первого класса относительно наблюдений из второго класса, что обеспечит нам стопроцентную точность классификации.

Всё это мы вели к тому, что можно моделью нейронной сети архитектуры представленной на рисунке 1.10 добиться идеальной классификации рассматриваемой задачи – ограничения логистической регрессии преодолены.

Однако в любой реальной задаче классификации принципы разделения на классы, конечно, неизвестны, потому веса и свободные члены каждого слоя оцениваются статистически. На рисунке 1.9 справа показана работа реально обученной только на статистических данных нейронной сети. Для каждого наблюдения мы вычислили сумматорные функции нейронов первого слоя и обозначили на рисунке формой и цветом как они распределились по знаку названной функции. Присмотревшись к такой диаграмме рассеяния мы заметили, что без ошибок не обошлось – они выделены на рисунке чёрными кружками.

При построении моделей опирающихся на искусственные нейронные сети кроме множества возможностей для идентификации нейронной сети, существуют возможности настройки алгоритма обучения, что может вести вообще говоря, к различным моделям. Модель для поставленной задачи, точно можно привести к идеальной классификации, но этот пример нам даже на руку, можно показать роль, которую в финальном решении играют веса выходного слоя.

Как зависят предсказываемые вероятности от конкретной точки в системе координат представлено на рисунке А.2 – сигмоида состоящая из двух участков под разными углами. И несмотря на то, что ряд пограничных точек приводят к формированию сигнала в выходной слой (хотя по заложенным закономерностям недолжны), веса и свободный член выходного слоя придали им большую сумматорную функцию выходного слоя, нежели для наблюдений «синего» класса. Это привело к тому что все точки относящиеся ко второму классу на графике А.2 выше нежели точки первого класса – от сюда, правильно выбрав высоту , можно добиться 100% точности классификации.

Этот подраздел работы вводит понятие искусственной нейронной сети, с краткими наиболее общими идеями лежащими в основе этой группы моделей. Нейронная сеть представляет собой последовательность преобразований входных сигналов, каждое из которых в своей сумматорной функции сочетает выходные сигналы предыдущего слоя и к этому сочетанию применяет активационную функцию результат которой отправляется в сумматорные функции следующего слоя или, для выходного слоя, формирует предсказание модели. Далее вводиться архитектура нейронной сети в которой скрытые слои представляют собой ReLU функции а выходной слой содержит сигмоидальный нейрон. На примере подробно раскрыты механизмы почему эта модель работает, как она связана и развивает идеи модели логистической регрессии.

В этом разделе основное внимание было уделено тому как названная модель применяется для известной закономерности, но на практике закономерности неизвестны, а имеются лишь наблюденные данные и идентификацию и оценку параметров надо производить в этих условиях. В следующем разделе описаны идеи как производиться оценка параметров, а более детальная идентификация модели в контексте нейронных сетей лучше всего раскрывается на практике, и будет представлена в третьей главе.

# Целевые функции и алгоритм обратного распространения ошибки

Процесс оценки коэффициентов в контексте нейронных сетей принято называть обучением модели.

Ключевым пунктом является выдвижение некоторого правила которое оценивает насколько модель соответствует наблюденным данным – целевую функцию. В общем, требуется ввести функцию:

где – реально наблюденный вектор предсказываемого явления;

­– вектор текущих предсказаний.

Очевидно, что вектор предсказаний формируется данными, проходящими через модель т.е. правомерна запись:

|  |  |
| --- | --- |
| где – | реальное множество наблюдений за факторами для которых предполагается влияние на ; |
| –ф | множество коэффициентов которые на разных этапах оказывают влияние на отклик вычисляемый моделью. |

Итак окончательно обобщённая целевая функция примет вид:

Она должна быть тем больше, чем сильнее отличаются наблюденные и предсказанные значения. Нам выгодно, чтобы полученные предсказания были максимально похожи на наблюденные, потому нам тем лучше чем меньше эта функция.

В контексте рассматриваемого вопроса можно сказать, что множества *X* и *Y* неизменны (хотя некоторые продвинутые обучающие алгоритмы могут использовать на разных стадиях обучения разные подмножества этих множеств).

Приходим к тому, что необходимо подобрать такие коэффициенты , чтобы целевая функция была минимальна, или более формально:

По выводам предыдущего подраздела очевидно, что в случае нейронной сети – нелинейная функция, да и все широко применяемые целевые функции также не линейны, потому очевидно, что перед нами стоит задача нелинейной оптимизации.

Для решения таких задач широко распространена группа методов основанных на градиентном спуске. Основная идея этих методов заключается в том, чтобы постепенно от некоторой выбранной начальной точки двигаться в направлении наискорейшего убывания .

В методах градиентного спуска используется свойство антиградиента функции, которое утверждает, что функция в каждой точке убывает быстрее всего в направлении её антиградиента, который определяется так:

|  |  |
| --- | --- |
| где – | некоторый отдельный коэффициент модели . |

Возвращаясь к искусственным нейронным сетям приходим к тому, что единственная сложность реализации этого алгоритма заключается в том, чтобы получить частные производные по всем весам. К решению этой проблемы призван метод обратного распространения ошибки. Будем его рассматривать как надстройку над методами градиентного спуска.

В процессе разъяснения принципа метода обратного распространения ошибки нам пригодится рисунок 1.11.



**Рисунок 1.11 – Связь соседних слоев**

Примечание ­­­– Источник: собственная разработка.

И некоторые обозначения. Матрица весов *(l+1)-го* слоя

Вектор из весов помножаемых на *j-ю* активацию *l-го* слоя, это-же столбец *j* матрицы весов *(l+1)-го* слоя

|  |  |  |
| --- | --- | --- |
|  |  | (1.14) |

И так называемая, ошибка *j-го* нейрона *l-го* слоя, определяемая так:

|  |  |  |
| --- | --- | --- |
|  |  | (1.15) |

Выясним взаимосвязь ошибок *l-го* и *(l+1)-го* слоёв. Для того придадим некоторое малое приращение активации и посмотрим какое приращение при этом получат сумматорные функции *(l+1)-го* слоя. В силу линейности сумматорных функции приращение для составит .

Пользуясь свойством, что при достаточно малом приращении аргументов соответствующее приращение функции близко к скалярному произведению градиента и вектора приращений аргументов получаем:

Или используя свойство ассоциативности скалярного произведения, обозначения (1.14) и (1.15):

где – вектор из значений сумматорной функции *(l+1)-го* слоя;

­– вектор ошибок *(l+1)-го* слоя.

Разделив обе части уравнения на , устремив его к нулю и вспомнив, что это приращение активационной функции запишем:

Далее вспоминая, что , функция от можно сказать и используя правило дифференцирования сложной функции:

Опять обратившись к (1.15) и переходя к матричной форме записи, получим:

|  |  |  |
| --- | --- | --- |
|  |  | (1.16) |

Формула (1.16) самый важный вывод метода обратного распространения ошибки, собственно она отражает название метода – каждая предыдущая ошибка итеративно вычисляется из следующей, конечно, кроме ошибки выходного слоя, которую можно записать так:

Или переходя к матричной записи:

|  |  |  |
| --- | --- | --- |
|  |  | (1.17) |

Заметим так-же, что основной целью было названо выделение частных производных целевой функции по весам и свободным членам. Учитывая (1.15) и то что функция от весов и свободных членов, не составляет труда очередной раз использовать правило дифференцирования сложной функции и получить:

|  |  |  |
| --- | --- | --- |
|  |  | (1.18) |

К сожалению ни в одном из источников мы не нашли причины, по которой, используется именно такой подход, а не просто аналитически выводятся производные по весам и смещениям. Но можно предположить, что такой подход однозначно проще подлежит программированию – достаточно знать лишь производные активационных функций по их аргументам, и вне зависимости от слоя в котором находиться нейрон, зная его активацию, для текущих параметров модели можно получить производную по весам и смещениям используя одни и те-же рекуррентные формулы (1.18).

Заметим так же, что рассуждения, касающиеся этого метода, мы строили в отрыве от наблюденных данных *X*, *Y*. Все выше сказанное будет справедливо для каждого конкретного примера по отдельности, а должны быть учтены все примеры участвующие в обучении. Благо все целевые функции построены на том, что суммируют ошибки отдельных примеров, а производная суммы, как известно, сумма производных потому результаты полученные из (1.18) надо просто просуммировать для каждого отельного примера и будет получена действительно производная целевой функции.

И так, теперь, на основе метода обратного распространения ошибки сформируем одноименный алгоритм оценки параметров нейронной сети:

1. Выполнить прямое распространение активации для каждого примера (подставить *X* в модель), сохраняя промежуточные активации;
2. Использовать формулу (1.17) для вычисления ошибок выходного слоя;
3. Произвести обратное распространение ошибки используя формулу (1.16) – будут получены ошибки всех нейронов сети;
4. Используем формулы (1.18) просуммировав их значения для каждого примера – будут получены градиенты по весам и смещениям;
5. Обновление параметров вдоль антиградиента;

Описанные пункты повторяются до тех пор пока не будут выполнены условия остановки алгоритма выбранной вариации градиентного спуска.[5 с. 243]

В целом эта глава была посвящено методам классификации. Изначально была раскрыта подстановка задачи – требуется на основе наблюденных данных сформировать правило, которое получив некоторую информацию об объекте сможет отнести его к нужному классу. В приложении к кредитному риску: на основе данных предоставляемых клиентом в заявке на получение кредита, требуется принять решение о выдаче или удержании займа.

Описанная задача не нова и для ее решения широко используются, уже ставшие классическими, методы: логистическая регрессия и дискриминантный анализ. В начале главы описаны предпосылки использования логистической регрессии – невозможность решения поставленной задачи обычными регрессионными моделями объясняется, тем что эта группа моделей предсказывает число, в то время как нам требуется получить класс в качестве предсказания. Логистическая регрессия обходит это ограничение благодаря свойствам логит функции, которые позволяют предсказанное значение интерпретировать как вероятность отнесения к некоторому классу. В подразделе 1.2 подробно описывается процесс перехода к модели именно такой формы.

В том же разделе мы указали и на ограничения модели логистической регрессии – несмотря на очевидную нелинейность используемой формулы модель логистической регрессии остается линейным классификатором.

Для преодоления этого ограничения мы использовали концепцию нейронной сети, рассмотренной как развитие модели логистической регрессии через усложнение формы функции от показателей лежащей под логит функцией. По сути вместо линейной функции под логит функцию была положена кусочно-линейная функция. В подразделе 1.3 мы постарались раскрыть как этот механизм приводит к улучшению классифицирующих свойств модели в некоторых случаях.

Завершается глава исчерпывающим описанием математики используемой для формирования алгоритма оценки коэффициентов нейронной сети – метода обратного распространения ошибки. В целом, это тот же численный метод оптимизации как градиентный спуск, но частные производные целевой функции вычисляются особым способом, очевидно, особенно удобным для программирования.

На практике, кроме оценки коэффициентов, для модели требуется дополнительно еще целый комплекс мер связанных подготовкой данных. Кроме того в идентификация модели нейронной сети это отдельное «искусство» неразрывно связано с методом подбора решения – хитрости и уловки используемые соответствующими специалистами лучше всего раскрывать на практике. Этим вопросам и посвящены следующие разделы этой работы.

# Анализ и подготовка данных

# 2.1 Анализ наблюденной статистики и отбор предикторов

Сразу укажем, что все действия связанные с вычислениями и обработкой информации мы производили на языке программирования python3 – одном из самых популярных и востребованных на сегодняшний день инструментов для математического моделирования и обработки данных. В работе использованы библиотеки-расширения pandas[6], numpy[7], scipy[8], sklearn[9], pytorch[10]. Весь исходный код в формате jupyter notebook или исполняемых «.py» файлов, результаты его выполнения в необработанном виде и даже текст этой работы можно найти на предварительно мной созданном репозитории[[1]](#footnote-1) в сети интернет.

В связи с коммерческой тайной мы не можем предоставить данные использованные для построения модели в открытый доступ, приходиться принять, вызванную этим фактом, непрозрачность описываемого исследования. Тем не менее, мы на словах и промежуточных результатах вычислений постараемся по максимуму раскрыть полный цикл разработки модели – от «сырой» таблицы показателей и до валидации готовой модели.

Этот раздел посвящен анализу полученной из баз данных банка информации и, на его основании, обоснования и применения решений связанных с преобразованием данных способствующих дальнейшему успеху при моделировании.

Для преобразования данных зачастую пишут отдельную программу со своими настройками, так как по результатам моделирования может потребоваться подгонка еще и параметров предварительной обработки данных – в этом случае следует просто поменять некоторые настройки процессинга данных и забрать по сути новый набор данных. Такую программу в западных источниках называют «pipeline» (пайплайн), мы будем называть конвейер данных. Учитывая, что конвейер данных собирается под конкретную структуру данных и то, что нет возможности предоставить данные в свободный доступ, не вижу никакого смысла, в техническом представлении этой программы в работе. Периодически лишь будут мелькать обобщённые методы для конкретных преобразований и представления результатов вычислений.

Та часть практики которая связана с аналитикой и подготовкой данных может быть найдена в папке «data\_processing» названного репозитория. Особое внимание следует обратить на файл «processing1.ipynb», его можно открыть прямо в браузере.

Изначально входной файл идет в формате «.xlsx» – его, при соблюдении внутри структуры как в таблице 1.1 можно в одну строку открыть с помощью метода «read\_excel» библиотеки pandas. На выходе будет получен@ «padnas.DataFrame» – объект представляющий собой данные вида таблицы 1.1. Заметим, что большой объем данных из формата «.xlsx» загружается достаточно долго, потому его лучше преобразовать в формат «.csv», который значительно проще. Сделать это можно прямо внутри pandas используя метод «to\_csv» объекта «padnas.DataFrame», важно в аргумент «index» передать значение «none» для того, чтобы не был сохранён еще и столбец с индексом который был автоматически создан при загрузке. После этой процедуры можно пользоваться созданным «.csv» файлом, загружая его функцией «read\_csv» библиотеки «pandas».

И так, изначальный набор данных имеет 247062 записей и 45 столбцов-показателей. Для начала нужно разобраться с составом столбцов, их типом данных и структурой. Все эти операции можно провести используя только возможности pandas, но удобнее, чтобы это можно было сделать в одну строку кода. Для решения такой задачи создана функция языка программирования python «get\_data\_frame\_settings» исходный код представлен на листинге В.1. Применив её к объекту сданными на выходе будет получен, так-же «pandas.DataFrame». Для наших данных результат представлен в таблице Г.1.

Первый столбец содержит название показателя. Второй столбец посвящен типу данных, конечно в pandas типы данных другие – но в листинге В.1 объявлен словарь «types\_natural\_names» в котором реальным названиям типов данных pandas поставлены в соответствие русскоязычные названия. Можно заметить, что перечислены не все типы данных pandas и те которые не упомянуты будут отображаться так, как они бы выглядели в pandas, в случае необходимости можно добавить свой пункт в этот словарь. И так, у нас в таблице имеются данные следующих типов – дата, целое число, действительное число и номинативная.

Третий столбец содержит информацию о возможных значениях которые принимает соответствующий показатель. Для дат и чисел это будет интервал, для номинативных показателей перечисленные через запятую уровни наблюденные в этом столбце. Сразу заметим, что показатели «Адрес проживания - Населенный пункт», «Вид деятельности по ОКЭД», «Кредитный продукт» и «Место работы» принимали настолько много уровней, что все их выписать в эту работу не представлялось возможным, из соображений оформления, потому они были заменены.

Четвёртый столбец, для номинативных переменных содержит число уровней которые они принимают.

Отдельную сложность в подготовке и анализе данных представляют ячейки с пропущенными значениями, в последнем столбце таблицы Г.1 выписаны количества таких наблюдений. Некоторые столбцы содержат до 95% пропусков.

Теперь, отталкиваясь от этой таблицы попробуем принять некоторые преобразования данных.

Особую ценность для нас составляет столбец «Дефолт», в нем содержится, целевая для нас информация – сколько дней, на момент среза, таблицы каждый контракт находиться в состоянии просрочки платежа. По указанию начальства, для целей моделирования, вышедшим в дефолт считается контракт, у которого в этом столбце наблюдается значение свыше 59 дней. Произведём перекодировку в новый столбец «*Y*» по правилу:

Исходный столбец «Дефолт» подлежит удалению.

Выше уже упоминались столбцы «Адрес проживания - Населенный пункт», «Вид деятельности по ОКЭД», «Кредитный продукт» и «Место работы». Было принято решение произвести их удаление. Столбцы «Адрес проживания - Населенный пункт» и «Место работы» в чистом виде было невозможно использовать для моделирования – в них содержалось слишком много различных уровней 6122 и 124712 соответственно, это вело бы к колоссальному увеличению коэффициентов модели, что, в свою очередь, к огромным объемам вычислений которые не привели бы к значимым результатам, ведь если, условно, в некотором городе найдется один кредитополучатель не выполнивший обязательства в срок это еще не значит, что от всех жителей этого года следует ожидать аналогичного поведения. Они, кончено могли подлежать укрупнению, но в данной работе этого сделано не было.

# ЗАКЛЮЧЕНИЕ

**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. – 400с.
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. – 2nd 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. Pandas, main page [Electronic resource]. – 02.04.2022.. – Mode of access: <https://pandas.pydata.org/>. – Date of access: 10.04.2022.
7. Numpy, main page [Electronic resource]. – 2022. – Mode of access: <https://numpy.org/>. – Date of access: 10.04.2022.
8. Scipy, main page [Electronic resource]. – 05.02.2022. – Mode of access: <https://scipy.org/>. – Date of access: 10.04.2022.
9. Scikit-learn, machine learning in python [Electronic resuorce]. – 2021. – Mode of access: <https://scikit-learn.org/stable/>. – Date of access: 10.04.2022.
10. PyTorch, main page [Electronic resource]. – Mode of access: <https://pytorch.org/>. – Date of access: 10.04.2022.

# ПРИЛОЖЕНИЕ А

# Визуализация сигмоиды двух переменных



**Рисунок А.1 – Сигмоида линейной модели двух переменных**

Примечание ­­­– Источник: собственная разработка.



**Рисунок А.2 – Сигмоида кусочно-линейной модели двух переменных**

Примечание ­­­– Источник: собственная разработка.

# ПРИЛОЖЕНИЕ Б

# Архитектуры с сигмоидой на выходном слое



**Рисунок Б.1 – Логистическая регрессия как нейронная сеть**

Примечание ­­­– Источник: собственная разработка.



**Рисунок Б.2 – Центральная архитектура сетей используемая в работе**

Примечание ­­­– Источник: собственная разработка.

# ПРИЛОЖЕНИЕ В

# Обощенные программные функции написанные для работы

import pandas as pd

import numpy as np

types\_natural\_names = {

"datetime64[ns]" : 'Дата',

"category" : "Номинативная",

"bool" : "Номинативная",

"int64" : "Целое число",

"int32" : "Целое число",

"float64": "Дейсвительное число"

}

def get\_col\_av\_values(col):

'''Получить область допустимых значений колонки'''

if pd.api.types.is\_numeric\_dtype(col):

return "[" + str(np.min(col)) + ";" + str(np.max(col)) + ']'

if pd.api.types.is\_datetime64\_any\_dtype(col):

return "[" + np.min(col).strftime("%d.%m.%Y") + ";" +\

np.max(col).strftime("%d.%m.%Y") + ']'

else:

return str(col.cat.categories.tolist())[1:-1].replace("'", "")

def get\_col\_cat\_count(col):

'''получить для категориальной переменной число

уровней, а для численной "-"'''

if pd.api.types.is\_numeric\_dtype(col) or\

pd.api.types.is\_datetime64\_any\_dtype(col):

return "-"

else:

return len(col.cat.categories)

def get\_data\_frame\_settings(df):

result = pd.DataFrame(columns = [

'Тип данных',

'Область допустимых значений',

'Число допустимых значений',

'Число пропусков'

])

for col in df.columns:

result.loc[col, :] = [

df[col].dtype,

get\_col\_av\_values(df[col]),

get\_col\_cat\_count(df[col]),

sum(df[col].isna())

]

result["Тип данных"] = result["Тип данных"].replace(types\_natural\_names)

return result

**Листинг В.1 – Функция python3 для получения описания типов данных столбцов произвольного pandas.DataFrame**

Примечания

1 Источник: собственная разработка.

2 Для корректной работы функций предварительно необходимо установить библиотеки pandas, numpy.

# ПРИЛОЖЕНИЕ Г

# Таблицы, описывающие данные на разных этапах обработки

**Таблица Г.1 – Начальный формат данных.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название показателя | Тип данных | Область допустимых значений | Число допустимых значений | Число пропусков |
| 1 | 2 | 3 | 4 | 5 |
| Автомобиль год выпуска1 | Дата | [18.01.1986;22.12.2019] | - | 202744 |
| Автомобиль год выпуска2 | Дата | [22.01.1986;10.11.2019] | - | 245403 |
| Автомобиль год выпуска3 | Дата | [11.03.1987;30.05.2019] | - | 246960 |
| Воинская служба | Номинативная | военнослужащий, другое, не отслужил, невоеннообязанный, отсрочка, призывник, уволен в запас | 7 | 108914 |
| Количество детей | Дейсвительное число | [1.0;10.0] | - | 161319 |
| Количество иждивенцев | Дейсвительное число | [1.0;10.0] | - | 237508 |
| Недвижимость | Номинативная | есть, нет | 2 | 40 |
| Образование | Номинативная | высшее, незаконченное высшее, неполное среднее, среднее, среднее специальное | 5 | 70933 |
| Отношение к банку | Номинативная | акционер, другое, не имеет отношения, работник | 4 | 1060 |
| Работа занимаемая должность | Номинативная | безработный (временно не работающий), государственный служащий, заместитель руководителя, индивидуальный предприниматель, пенсионер, рабочий, руководитель, специалист, студент | 9 | 0 |

**Продолжение таблицы Г.1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Работа последнее место стаж лет | Дейсвительное число | [-998.0;50.0] | - | 1035 |
| Работа уровень дохода BYR | Дейсвительное число | [-1500.0;92207.0] | - | 923 |
| Семейное положение | Номинативная | вдовец/вдова, женат/замужем, повторный брак, разведен/разведена, холост/не замужем | 5 | 63178 |
| Собственная квартира | Номинативная | есть, нет | 2 | 40 |
| Собственный дом | Номинативная | есть, нет | 2 | 40 |
| Уголовная ответственность | Номинативная | есть, нет | 2 | 40 |
| Адрес проживания - Населенный пункт | Номинативная | Название населенного пункта | 6122 | 0 |
| Адрес проживания - Тип населенного пункта | Номинативная | Агрогородок, Город, Городcкой поселок, Деревня, Курортный поселок, Поселок сельского типа, Рабочий поселок, Село, Сельский населенный пункт, Хутор | 10 | 80333 |
| Вид деятельности по ОКЭД | Номинативная | Вид деятельности по ОКЭД | 44 | 15253 |
| Гражданин РБ | Номинативная | Без гражданства, Другое, РБ | 3 | 68942 |
| Дата регистрации договора | Дата | [04.01.2016;21.12.2020] | - | 0 |
| Дата рождения | Дата | [03.01.1878;30.09.2002] | - | 0 |
| Был ли хоть один договор прекращен досрочно | Номинативная | есть, нет | 2 | 137919 |
| Количество действующих договоров обеспечения | Дейсвительное число | [0.0;44.0] | - | 179682 |
| Количество действующих кредитных договоров | Дейсвительное число | [0.0;26.0] | - | 149948 |
| Количество запросов в КБ за последние 30 дней | Дейсвительное число | [0.0;15.0] | - | 135276 |

**Продолжение таблицы Г.1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Количество фактов просрочки по основному долгу | Дейсвительное число | [0.0;284.0] | - | 165097 |
| Максимальное количество дней просрочки | Дейсвительное число | [0.0;4471.0] | - | 172113 |
| Максимальный срок, на который заключался договор, в годах | Дейсвительное число | [0.0;122.08] | - | 139103 |
| Наличие кредитной истории | Номинативная | есть, нет | 2 | 134471 |
| Общее количество запросов в КБ | Дейсвительное число | [0.0;222.0] | - | 135276 |
| Отношение факт срока к плановому при прекращении КД | Дейсвительное число | [0.0;10000.0] | - | 145229 |
| Причина прекращения договора | Номинативная | Прекращение договора исполнением, Прекращение договора по иным основаниям предусмотренным законодательством Республики Беларусь, Прекращение договора по решению суда, Прощение долга | 4 | 151260 |
| Сумма кредитных лимитов | Дейсвительное число | [0.0;678562.6] | - | 149948 |
| Дата планируемого закрытия | Дата | [20.05.2016;20.07.2058] | - | 10566 |
| Дата фактического закрытия | Дата | [15.01.2016;08.07.2021] | - | 107885 |
| Кредитный продукт | Номинативная | Название кредитного продукта | 262 | 748 |
| Овердрафт | Номинативная | есть, нет | 2 | 0 |
| Сумма договора | Дейсвительное число | [-549.81;600000.0] | - | 0 |
| Количество потребляемых банковских продуктов | Номинативная | 1, 2, более 2-х | 3 | 118779 |

**Окончание таблицы Г.1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Место работы | Номинативная | Наименование предприятия | 123712 | 1445 |
| Пол | Номинативная | Ж, М | 2 | 0 |
| Социальная группа | Номинативная | безработный, индивидуальный предприниматель, пенсионер, работающий по найму, служащий, учащийся | 6 | 89495 |
| Дефолт | Целое число | [0;1693] | - | 0 |
| Код подразделения | Номинативная | 739-100, 739-200, 739-200-202, 739-200-203, 739-200-228, 739-300, 739-400, 739-600, 739-800, 739-800-831, 739-900, 739-900-500, 739-900-527, 739-900-535, 739-900-536, 739-900-537, 739-900-538, 739-900-905, 739-900-906, 739-900-907, 739-900-932, 739-900-933 | 22 | 0 |

1. https://github.com/Dranikf/diplom\_project [↑](#footnote-ref-1)