

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
на тему
ПРОГНОЗИРОВАНИЕ УСПЕВАЕМОСТИ
СТУДЕНТОВ С ИСПОЛЬЗОВАНИЕМ
НЕЙРОННЫХ СЕТЕЙ

Студент 20.06.2024 Юрова Ольга Викторовна
(подпись, дата) (ФИО полностью)

Руководитель 20.06.2024 Карамышева Надежда Сергеевна
(подпись, дата) (фамилия, инициалы)

Нормоконтролер 20.06.2024 Бычков Андрей Станиславович
(подпись, дата) (фамилия, инициалы)

Рецензент Василийков А. В. АО «НПП «РУБИН»
(подпись, дата) (фамилия, инициалы) (должность, место работы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 № 11)

Заведующий кафедрой Митрохин Максим Александрович
(подпись) (фамилия, инициалы)

Работа защищена с отметкой (протокол заседания ГЭК от 21.06.24 № 33)

Секретарь ГЭК Гурин Е. И.
(подпись) (фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой

 М. А. Митрохин
«13» сентября 2022г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ МАГИСТРА**

1. Студентка Юрова Ольга Викторовна гр. 22ВВИм2 факультета вычислительной техники, направления 09.04.01.

2. Тема работы «Прогнозирование успеваемости студентов с использованием нейронных сетей».

Тема утверждена приказом ПГУ № 765/о от «13» сентября 2022 г.

3. Руководитель магистерской диссертации к.т.н., доцент кафедры ВТ ПГУ Карамышева Н. С.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

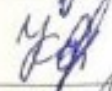
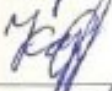
Назначение разработки: создание нейронной сети для прогнозирования успеваемости студентов.

Разработать нейронную сеть для прогнозирования успеваемости по сформированным наборам данных.

5. Перечень подлежащих разработке вопросов:

- анализ уже действующих решений из сферы нейронных сетей,
- анализ и выявление значимых признаков данных для прогнозирования,
- формирование набора данных, удовлетворяющего требованиям,
- разработка и реализация принципа и структуры нейронной сети,
- тестирование системы согласно техническому заданию на разработку.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучить теоретические основы	20	27.09.2022 - 15.11.2023	
2.	Анализ существующих нейронных сетей	40	16.11.2023 - 22.03.2023	
3.	Анализ и выявление значимых признаков данных для прогнозирования	40	23.03.2023 - 14.06.2023	
4.	Выбор инструментов для разработки и системы	50	15.06.2023 - 10.09.2023	
5.	Разработка принципа и структуры нейронной сети	50	11.09.2023 - 05.01.2024	
6.	Реализация нейронной сети	40	06.01.2024 - 17.04.2024	
7.	Апробация разработанной нейронной сети в форме тестирования	10	18.04.2024 - 09.05.2024	
8.	Оформление выпускной работы	20	10.05.2024 - 28.05.2024	

Дата выдачи задания «13» сентября 2022 г.

Руководитель магистерской диссертации  13.09.22 Карамышева Н.С.
(подпись, дата)

Задание к исполнению принял студент  13.09.22 Юрова О. В.
(подпись, дата)

Работу к защите допустить
Декан факультета  21.06.2024 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Магистерская работа содержит 81 лист, 23 рисунка, 16 источников, 2 приложения

НАБОР ДАННЫХ, DATASET, ПРОГНОЗИРОВАНИЕ, КЛАСС, НЕЙРОННЫЕ СЕТИ, PYTHON, IDE SPYDER

Цель работы – разработка диалоговой системы вуза для прогнозирования успешности обучения студентов в условиях электронной информационно-образовательной среды вуза на основе технологии нейронных сетей.

В результате проведенной работы было разработано программное обеспечение, позволяющее делать прогнозы успеваемости обучающихся за счет интеграции с нейронной сетью, позволяющие использовать конкретный датасет для прогнозирования.

В ходе разработки был использован язык программирования Python и среда разработки IDE Spyder.

					ПГУ1.090401.49.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Прогнозирование успеваемости студентов с использованием нейронных сетей	Лит.	Лист	Листов	
Разработал		Юрова О.В.		20.06.24					
Руководитель		Карамышева Н.С.		20.06.24			2	81	
Н. Контр		Бычкова А.С.		20.06.24		ФВТ, каф. ВТ			
						гр. 22ВВИМ2			
Утвердил		Митрохин М.А.		20.06.24	Пояснительная записка				

СОДЕРЖАНИЕ

Перечень сокращений и обозначений	5
ВВЕДЕНИЕ	6
1 Анализ предметной области	8
1.1 Общая характеристика задач машинного обучения	8
1.2 Архитектуры нейронных сетей	11
1.3 Виды нейронных сетей	12
1.3.1 Рекуррентные нейронные сети	12
1.3.2 Хаотические нейронные сети	15
1.3.3 Осцилляторные нейронные сети	16
1.3.4 Глубокие нейронные сети	18
1.3.5 Сверточные нейронные	19
1.3.6 Генеративно-состязательные нейросети	20
1.4 Постановка задачи обучения	21
2 Анализ аналогов	24
2.1 Gradescope	24
2.2 MyEdu	25
2.3 Dreambox	26
2.4 Smart Sparrow	27
2.5 Статистика успеваемости	28
3 Анализ и требования к используемым данным для обучения	29
3.1 Сбор данных для модели	30
3.2 Формирование датасета	37
4 Реализация работы с нейронной сетью	40
4.1 Реализация с использованием глубокой нейронной сети	40
4.1.1 Реализация глубокой нейронной сети	40
4.1.2 Реализация процесса обучения нейронной сети	43
4.1.3 Реализация процесса оценки обучения нейросети	45

4.2 Реализация с использованием RNN	46
4.2.1 Реализация RNN.....	46
4.2.2 Реализация процесса обучения RNN	49
4.2.3 Реализация процесса оценки обучения RNN	54
4.2.4 Реализация процесса прогнозирования	56
Заключение	60
Список использованных источников	63
Приложение А Листинг программы.....	66
Приложение А.1Файл main_Y.py	67
Приложение А.2Файл RNN_Y.py.....	69
ПриложениеА.3Файл RNN_main_train.py	71
Приложение Б Презентация	77

Перечень сокращений и обозначений

ИИ – искусственный интеллект

ПК – персональный компьютер

РНС – рекуррентная нейронная сеть

ЭИОС – электронная информационно-образовательная среда

ЯП – язык программирования

API (Application Programming Interface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

GPT (Generative Pre-trained Transformer) – генеративный, предобученный трансформер

IDE (Integrated development environment) – интегрированная среда разработки

NN (neural network) – нейронная сеть

RNN (Recurrent neural network) – рекуррентная нейронная сеть

UML (Unified Modeling Language) – унифицированный язык моделирования

ВВЕДЕНИЕ

Благодаря техническому прогрессу, многие компании в настоящее время применяют технологию *datamining*, позволяющую обрабатывать существенные объемы данных и извлекать из них полезную информацию.

Задачей этого анализа является выявление скрытых правил и закономерностей в наборах данных. Долгое время основным инструментом оставалась статистика, однако она не всегда способна эффективно решать задачи реальной жизни. Основным её применением является проверка заранее сформулированных гипотез [2].

Изначально использование нейронных сетей в анализе данных вызывало скептические настроения из-за недостатков нейросетей, таких как сложная структура, низкая интерпретируемость и долгое время обучения. Однако их преимущества, такие как хорошая адаптивность к зашумленным данным, постоянная оптимизация и улучшение алгоритмов обучения, правил и упрощения сетей, представляют нейронные сети как все более перспективное направление в *datamining*-области [3].

Нейросети применимы в совершенно различных областях, начиная автоматизацией распознавания образов, прогнозированием, адаптивным управлением и созданием экспертных систем, организацией ассоциативной памяти, заканчивая обработкой сигналов, синтезом и идентификацией электрических цепей и систем [1].

Развитие электронной среды в университетах, формирование электронных и дистанционных технологий образования становится нормой нашего времени. В помощь принятия решений по подготовке студентов приходит инструмент использования цифровых средств и электронной информационно-образовательной среды (ЭИОС). Современные технологии информатизации системы образования приводят к появлению большого количества разнообразных систем сбора, обработки и хранения больших массивов информации, которые можно использовать для эффективного

прогнозирования успеваемости учащихся. Прогнозирование успеваемости учащихся – важный инструмент оценки эффективности применяемых педагогических технологий и методических приемов для преподавателей.

Целью магистерской работы является проектирование, разработка и реализация модели прогнозирования класса обучения студентов (отличник, ударник, троечник, неуспевающий) с использованием электронной информационно-образовательной среды (ЭИОС) ВУЗа на основе технологии нейронных сетей.

Основными задачами работы является:

- изучение методов и способов классификации с использованием нейросети,
- реализация модели нейросети для прогнозирования класса обучающегося,
- обучение реализованной нейросети,
- апробация реализованного функционала.

Работа может быть полезна кураторам и сотрудникам, занимающимся работой со студенческой успеваемостью. Заранее определить на кого из учащихся стоит обратить особое внимание.

1 Анализ предметной области

Начальный этап задач, предполагающий работу в рамках информационных технологий, представляет собой теоретический анализ многих аспектов: технологической, предметной и информационной основы задачи, инструментов, с помощью которых можно прийти к наиболее эффективному решению, разделения задачи на последовательные функциональные блоки, реализации из которых приведет к ее решению, возможные архитектурные решения, которые могут быть использованы для получения приложения, отвечающего всем требованиям.

Первым шагом к выполнению необходимого анализа будет исследование существующих нейронных сетей, а также алгоритмов машинного обучения.

1.1 Общая характеристика задач машинного обучения

Методы Data Mining помогают решить многие задачи, с которыми сталкивается аналитик. Основными из НИХ являются: классификация, регрессия, поиск ассоциативных правил и кластеризация. Далее приведено краткое описание основных ключевых задач анализа данных:

- задача классификации сводится к определению класса объекта по его характеристикам. Важно отметить, что в этой задаче множество классов, к которым может быть отнесен объект исследования, заранее известно,
- задача регрессии, подобно задаче классификации, позволяет определить по известным характеристикам объекта исследования значение его параметра. В отличие от задачи классификации значением параметра является непрерывным множеством действительных чисел,
- при поиске ассоциативных правил основной целью обнаружение часто встречающихся связей между объектами или событиями. После нахождения этих связей они записываются в виде правил, которые могут быть использованы для более глубокого понимания рассматриваемых данных и предсказания будущих событий,

- задача кластеризации заключается в поиске независимых групп (кластеров) и их характеристик в общем объеме анализируемых данных. Решение этой задачи помогает лучше понять данные. Кроме того, группировка однородных объектов позволяет уменьшить их количество, что в свою очередь упрощает анализ.

Рассмотренные задачи по назначению делятся на описательные и предсказательные.

Описательные (descriptive) задачи уделяют внимание улучшению понимания анализируемых данных. Ключевой момент в таких моделях – легкость и прозрачность результатов для восприятия человеком. Возможно, обнаруженные закономерности будут специфической чертой именно конкретных исследуемых данных и больше нигде не встретятся, но это все равно может быть полезно и потому должно быть известно. К такому виду задач относятся кластеризация и поиск ассоциативных правил.

Решение предсказательных (predictive) задач разбивается на два этапа. На первом этапе на основании набора данных с известными результатами строится модель. На втором этапе она используется для предсказания результатов на основании новых наборов данных. При этом, естественно, требуется, чтобы построенные модели работали максимально точно. К данному виду задач относят задачи классификации и регрессии. Сюда можно отнести и задачу поиска ассоциативных правил, если результаты ее решения могут быть использованы для предсказания появления некоторых событий.

По способам решения задачи разделяют на supervised learning (обучение с учителем) и unsupervised learning (обучение без учителя) (рисунок 1). В случае supervised learning задача анализа данных решается в несколько этапов. Сначала с помощью какого-либо алгоритма DataMining строится модель анализируемых данных – классификатор. Затем классификатор подвергается обучению. Другими словами, проверяется

качество его работы и, если оно неудовлетворительно, происходит дополнительное обучение классификатора.

Так продолжается до тех пор, пока не будет достигнут требуемый уровень качества или не станет ясно, что выбранный алгоритм не работает корректно данными, либо же сами данные не имеют структуры, которую Можно выявить. К этому типу задач относят задачи классификации и регрессии.

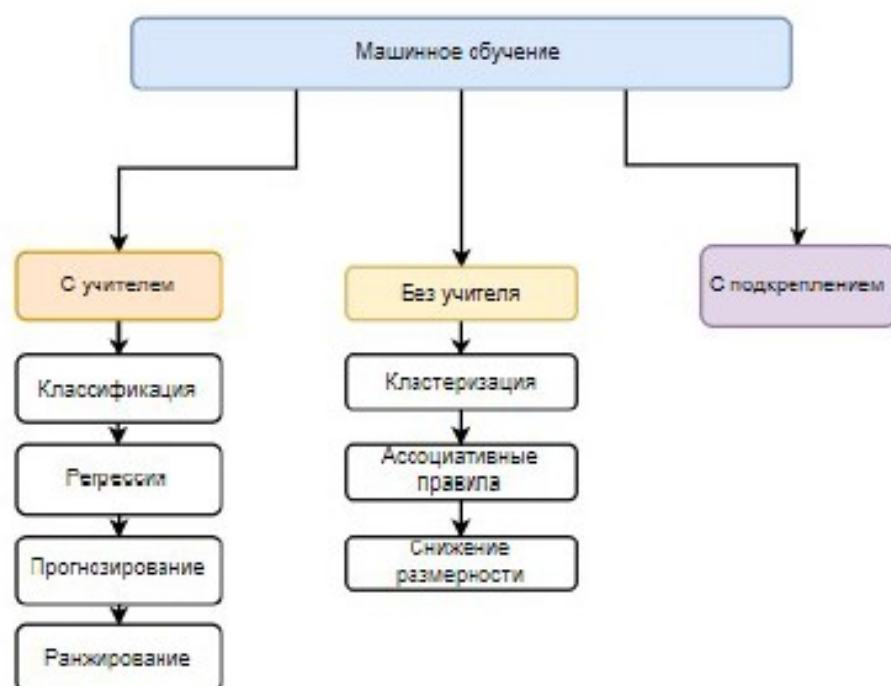


Рисунок 1 – Виды машинного обучения

Unsupervised learning объединяет задачи, выявляющие описательные модели, например закономерности в покупках, совершаемых клиентами большого магазина. Очевидно, что если эти закономерности есть, то модель должна их представить и неуместно говорить об ее обучении. Отсюда и Название – unsupervised learning. Достоинством таких задач является возможность их решения без каких-либо предварительных знаний об анализируемых данных. К ним относятся задачи кластеризации и поиска ассоциативных правил.

Кластеризация – алгоритм находит правила классификации и количество классов. Основное отличие от задач классификации заключается в том, что группы и принципы их разделения неизвестны. Алгоритм кластеризации часто используют при подготовке данных для обучения с учителем.

Поиск ассоциативных правил – исходные данные представляются в виде признаков описаний. Требуется найти такие наборы признаков, и такие значения этих признаков, которые особенно часто встречаются в признаковых описаниях объектов [5].

Снижение размерности – заключается в том, чтобы по исходным признакам с помощью некоторых функций преобразования перейти к наименьшему числу новых признаков, не потеряв при этом никакой существенной информации об объектах выборки. В классе линейных преобразований наиболее известным примером является метод главных компонент [4].

1.2 Архитектуры нейронных сетей

Во всем многообразии искусственных нейронных сетей можно выделить две базовые архитектуры – слоистые и полносвязные сети.

Слоистые сети: нейроны расположены в несколько слоев. Нейроны первого слоя получают входные сигналы, преобразуют их и через точки ветвления передают нейронам второго слоя. Далее срабатывает второй слой и т.д. до k -го слоя, который выдает выходные сигналы. Если не оговорено противное, то каждый выходной сигнал i -го слоя подается на вход всех нейронов $i + 1$ -го. Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях. Стандартный способ подачи входных сигналов: все нейроны первого слоя получают каждый входной сигнал.

Полносвязные сети: каждый нейрон передает свой выходной сигнал остальным нейронам, включая самого себя. Выходными сигналами сети

могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети. Все входные сигналы подаются всем нейронам.

Элементы слоистых и полносвязных сетей могут выбираться по-разному. Существует, впрочем, стандартный выбор нейрон с неоднородным адаптивным линейным сумматором на входе (рисунок 2).

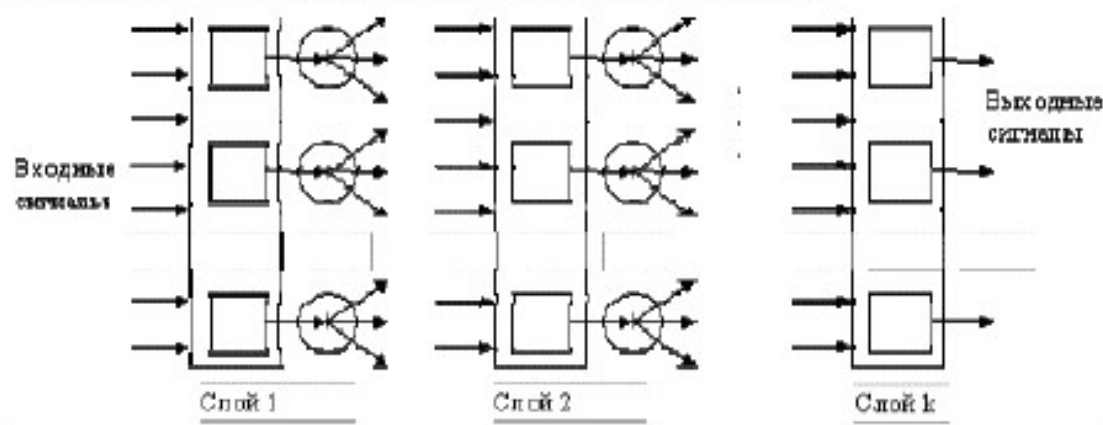


Рисунок 2 – Архитектуры нейронных сетей

1.3 Виды нейронных сетей

Существует широкий класс нейронных сетей и признаков, по которым их классифицируют. Рассмотрим подробнее некоторые типы сетей, представляющих особенный интерес с точки зрения возможности проведения динамического анализа и эффективности получаемых результатов.

1.3.1 Рекуррентные нейронные сети

Рекуррентными называются нейронные сети, имеющие одну или несколько обратных связей (рисунок 3). Обратные связи могут быть глобального типа. Рекуррентные сети, правило, используются в качестве ассоциативной памяти или сети отображения вход-выход. Применение обратной связи позволяет нейронным сетям представлять состояния, что делает их удобным инструментом в различных приложениях обработки сигнала и управления [6].

В класс рекуррентных сетей с глобальной связью входят следующие основные архитектуры сетей:

- сети нелинейной авторегрессии со внешними входами, в них используется обратная связь между выходными и входными слоями,

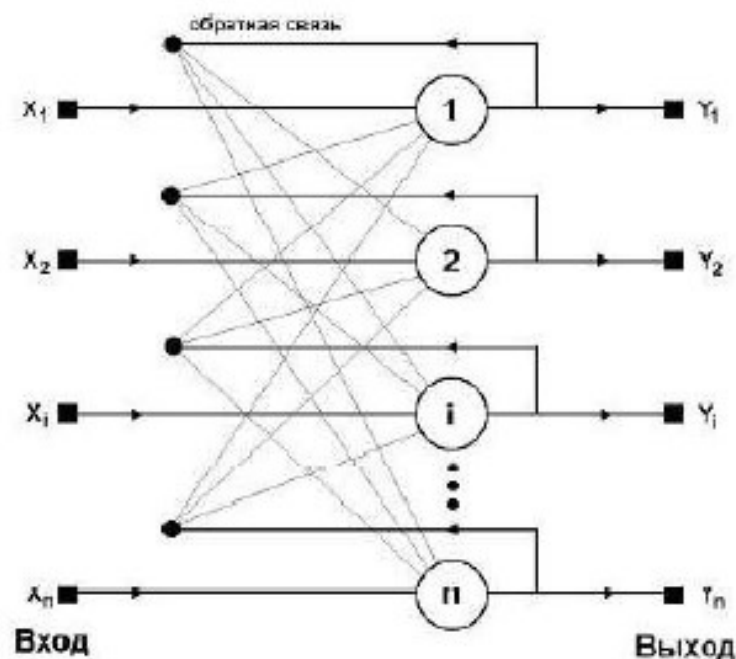


Рисунок 3 – Пример рекуррентной сети

- полносвязные рекуррентные сети с обратной связью между скрытым и входным слоем,
- рекуррентный многослойный перцептрон, содержащий более одного скрытого слоя, с обратными связями между каждым из расчета слоев и входным слоем,
- рекуррентные сети второго порядка, использующие нейроны второго порядка.

Первые три типа рекуррентных сетей допускают использование пространства состояний для изучения динамики среды. Этот подход берет начало в современной теории управления и является мощным методом изучения нелинейной динамики рекуррентных сетей.

Для обучения рекуррентных сетей можно выделить пять различных по своей сути групп подходов [7]:

1) использование известных соотношений между параметрами рекуррентной нейронной сети и ее динамикой (например, нейронная сеть Хакена и другие аттракторные рекуррентные нейронные сети, в которых путем задания весовых коэффициентов можно закодировать желаемые аттракторы на базе правила Хэбба),

2) использование алгоритмов обучения с учителем на базе методов оптимизации по типу алгоритма обратного распространения ошибки (например, алгоритмы обратного распространения во времени, рекуррентное обучение в реальном времени, рекуррентное обратное распространение и алгоритмы, использующие Калмановскую фильтрацию),

3) использование алгоритма обучения с учителем при рассмотрении сети как частично рекуррентной; сигналы с обратных связей рассматриваются как отдельные входные сигналы – контекстные нейроны, в результате задача обучения упрощается и сводится к обычным алгоритмам обучения нейронных сетей прямого распространения; к этому типу обучения относятся нейронные сети Элмана, Джордана и другие,

4) использование алгоритмов обучения без учителя (правило обучения Кохонена, синоптическая пластичность импульсных нейронных сетей),

5) отсутствие обучения в классическом смысле изменения параметров системы (неявное обучение) настройка весовых коэффициентов случайными значениями; при этом функции обучения возложены на специальное устройство – считыватель, который занимается классификацией динамики сети; такой принцип лег в основу новой парадигмы нейронных сетей - резервуарные вычисления.

Основная идея резервуарных вычислений заключается в использовании рекуррентной нейронной сети как резервуара с богатой динамикой и мощными вычислительными возможностями. При этом резервуар формируется случайным образом, что исключает необходимость проводить его обучение. При подаче на вход резервуара непрерывного сигнала он

попадает в определенный динамический режим или состояние. Это состояние зависит от входного сигнала. Резервуар формируется таким образом, чтобы для похожих входных сигналов это состояние было похожим, а для разных – разным. Выход резервуара соединен со специальными устройствами – считывателями, которые по состоянию резервуара решают поставленную задачу производят классификацию, предсказание, кластеризацию. В качестве считывателей могут использоваться статические обучающие машины с простыми алгоритмами обучения. Таким образом, резервуар интегрирует в своем состоянии динамику входного образа, а считыватели по состоянию резервуара распознают входной образ.

1.3.2 Хаотические нейронные сети

Один из разделов теории динамических систем связан с теорией хаоса и синергетикой. В хаотической динамике рождаются структуры под действием внешних возмущений, и она изначально может содержать все множество возможных вариантов. Хаотические системы позволяют выйти на следующий уровень обобщения и расширить возможности вычислений одновременно над целым множеством вариантов, причем это множество будет формироваться под воздействием внешних сигналов и тем самым обеспечивать адекватную сложность. Во многом это похоже на парадигму, используемую в квантовых вычислениях, в этом случае до момента считывания ответа содержится все множество вариантов решений [8].

Хаотическая нейронная сеть представляет собой однослойную рекуррентную сеть, в которой элементы связаны каждый с каждым, без образования связи сам на себя:

$$y_i(t+1) = \frac{1}{C_i} \sum_{j=1}^N w_{ij} f(y_j(t)), t = 1, \dots, T \quad (1)$$

$$f(y(t)) = 1 - 2y^2(t), \quad (2)$$

$$w_{ij} = \exp\left(-\frac{|x_i - x_j|^2}{2\alpha^2}\right), \quad (3)$$

где $C_i = \sum_{i \neq j} w_{ij}$, $i, j = \overline{1, N}$;

α – масштабирующая константа;

w_{ij} – сила связи (весовой коэффициент) между i и j нейронами;

N – число нейронов соответствующее числу параметров во входных данных, представленных в виде $X = (x_1, x_2, \dots, x_m)$;

m – размерность пространства образов;

T – время моделирования.

Обучение хаотической нейронной сети заключается в назначении весовых коэффициентов сети, которые определяются исходя из входного образа по соотношению (3) и однозначно определяют поле, которое действует на все нейроны сети.

1.3.3 Осцилляторные нейронные сети

Одним из перспективных средств, не использующих априорные знания об обрабатываемых данных, является осцилляторная нейронная сеть. Модель сети была разработана группой итальянских ученых под руководством Ангелини для задачи кластеризации в 2000 году [9].

Можно выделить четыре основные характеристики осцилляторной нейронной сети:

- 1) сеть является однослойной, рекуррентной и полносвязной,
- 2) элементами сети являются нейроны с передаточной функцией «логистическое отображение»,
- 3) сеть обладает свойством неаттракторности, то есть результат работы скрыт в динамике выходов нейронов,
- 4) для извлечения результата работы сети требуется анализ изменения выходов нейронов во времени.

Функциональной единицей сети, как правило, является осциллятор множество совместно функционирующих элементов (нейронов или нейронных ансамблей), способных работать в колебательном режиме. В зависимости от целей исследования выбирается не только определенная архитектура связей сети, но и конструкция от отдельного осциллятора. В наиболее сложных и максимально приближенных к реальности моделях из элементов с интегративно-используются осцилляторы, состоящие пороговыми свойствами разной степени сложности, являющиеся аналогами биологических нейронов. Осциллятор включает нейроны как с тормозными, так и с возбуждающими связями. Это приводит к тому, что при определенных условиях суммарная активность нейронов приобретает колебательный характер.

В литературе встречаются различные модели осцилляторов (Ван дер Поля, Курамото, Вилсона-Коуэна и т. д.). Для примера рассмотрим модель фазового осциллятора Курамото, позволяющего исследовать процессы и режимы синхронизации между осцилляторами, которые зависят от структуры и параметров нейронной сети.

Фаза осциллятора θ_i является основной переменной состояния и располагается в диапазоне от 0 до 2π . Внутренняя частота осциллятора ω_i в данной модели представляет собой смещение по фазе, проявляющееся в процессе синхронизации. Сила связи K между осцилляторами в нейронной сети является основным параметром, оказывающим влияние на режимы синхронизации. Параметр N в уравнении определяет количество осцилляторов в нейронной сети.

Модель Курамото описывается следующим уравнением:

$$\frac{\delta \theta_i}{\delta \tau} = \omega_i + \frac{K}{N} \sum_{j=1}^N \sin(\theta_j - \theta_i), i = 1, \dots, N. \quad (4)$$

1.3.4 Глубокие нейронные сети

Глубокие архитектуры, основанные на искусственных нейронных сетях, берут свое начало с неокогнитрона, разработанного, как было сказано ранее, Фукусимой в 1980 году. В 1989 году ЛеКуну удалось использовать алгоритм обратного распространения ошибки для обучения глубоких нейронных сетей при решении задачи распознавания написанных от руки почтовых индексов. Несмотря на успешный опыт, для обучения модели потребовалось три дня, что существенно ограничивало применение этого метода.

В 2006 году глубокое обучение было представлено в виде многослойной нейронной сети, первый слой которой выявлял основные особенности изображения, а последующие слои строили обобщенный образ объекта в виде комбинации простых примитивов. Процесс обучения осложнен наличием большого количества настраиваемых параметров. Обычно процесс обучения глубокой нейронной сети разбивают на два шага. На первом шаге проводится обучение однослойной сети, а на следующем шаге производится сопоставление весовых коэффициентов между слоями сети путем минимизации ошибки классификации. Такой подход позволяет сократить сложность обучения сети с большим количеством параметров благодаря оптимизации между слоями.

Глубокая нейронная сеть – это искусственная нейронная сеть с несколькими скрытыми слоями [10]. Дополнительные слои позволяют строить абстракции все более высоких уровней, что дает возможность формирования модели для распознавания сложных объектов реального мира. Обычно используются глубокие сети прямого распространения, однако, последние исследования показали успешное применение глубоких архитектур в рекуррентных сетях [11]. В задачах, связанных с обработкой изображений, преимущественно применяются сверточные нейронные сети ввиду их наибольшей эффективности.

Обучение глубоких нейронных сетей может быть осуществлено с помощью алгоритма обратного распространения ошибки. Таким образом, может быть использовано несколько правил настройки весов. Например, алгоритм стохастического градиентного спуска:

$$\omega_{ij}(t + 1) = \omega_{ij}(t) + \eta \frac{\delta L}{\delta \omega_{ij}}, \quad (5)$$

где η – постоянная для регулирования величины текущего шага,

L – функция потерь.

Выбор функции потерь может быть обусловлен классом задачи машинного обучения (с учителем, без учителя, с подкреплением) и функции активации.

1.3.5 Сверточные нейронные

Сверточные нейронные сети (CNN) – это тип нейронной сети, которая используется для анализа изображений и других визуальных данных. CNN состоят из сверточных слоев, которые обрабатывают изображения, и полносвязных слоев, которые объединяют информацию из разных областей изображения. CNN используются в задачах компьютерного зрения, таких как классификация изображений и распознавание объектов.

Основная идея сверточных нейронных сетей заключается в использовании сверточных слоев, которые позволяют извлекать признаки из изображений. Каждый сверточный слой состоит из нескольких фильтров, которые перемещаются по изображению и извлекают из него признаки. Затем эти признаки передаются на следующий слой, где они могут быть использованы для принятия решений.

Сверточные нейронные сети могут быть обучены на большом количестве изображений и способны распознавать объекты даже в сложных условиях, таких как изменение освещения, угла обзора и т.д. Однако, они

также могут быть подвержены проблемам переобучения и генерации шума, что может привести к неправильной классификации объектов на изображении.

1.3.6 Генеративно-сопоставительные нейросети

Генеративно-сопоставительные нейросети (Generative Adversarial Networks, GAN) – это тип нейронных сетей, которые используются для создания новых изображений, текстов и других объектов. GAN состоит из двух частей: генератора и дискриминатора. Генератор пытается создать реалистичные данные, а дискриминатор пытается отличить созданные данные от реальных.

Генератор и дискриминатор соревнуются друг с другом, при этом генератор обучается создавать данные, которые выглядят как настоящие, а дискриминатор обучается распознавать, какие данные были созданы генератором, а какие реальные.

GAN работает путем обучения генератора создавать новые данные, а также улучшать свои результаты, чтобы дискриминатор не мог отличить созданные им данные от настоящих. В результате GAN может создавать новые изображения, тексты или другие объекты, которые могут быть использованы для разных задач, таких как создание логотипов, создание новых видов животных или создание новых текстов.

Однако, GAN имеют некоторые недостатки, такие как сложность обучения, необходимость большого количества данных и возможность создания данных, которые не соответствуют реальности. Поэтому, GAN часто используются в сочетании с другими методами, такими как обучение с подкреплением или обучение на основе правил, чтобы получить более качественные результаты.

Это только некоторые из типов нейронных сетей, используемых в настоящее время. Каждая из них имеет свои преимущества и недостатки, и выбор конкретной сети зависит от конкретной задачи и доступных данных.

1.4 Постановка задачи обучения

Рассмотрим теорию обучения нейронных сетей в контексте обучения с учителем. Модель обучения с учителем состоит из трех взаимосвязанных компонентов, которые в математических терминах описываются следующим образом [17]:

1) среда: характеризуется распределением вероятностей $P_X(x)$ со случайно и независимо появляющимися случаями x ,

2) учитель: генерирует желаемый отклик d для каждого из входных векторов x , полученных из внешней среды, в соответствии с условной функцией распределения $P_X(dx)$.

Ни характеристика среды $P_X(x)$, ни правило классификации $P_X(dx)$ неизвестны. Однако известно, что обе функции существуют, т. е. существует совместное распределение вероятностей

$$P_X(d, x) = P(x) \cdot P_X(d|x). \quad (6)$$

Желаемый отклик d и входной вектор x связаны следующим соотношением:

$$d = f(x, v), \quad (7)$$

где v – шум, то есть изначально предполагается зашумленность данных учителя.

3) обучаемая машина: нейронная сеть способна реализовать множество функций отображения вход-выход, описываемых соотношением:

$$y = F(x, w), \quad (8)$$

где y – фактический отклик, сгенерированный обучаемой машиной в ответ на входной сигнал x ;

w – набор свободных параметров (синаптических весов), выбранных из пространства параметров W .

Задача обучения с учителем состоит в выборе конкретной функции $F(x, w)$, которая оптимально (в некотором статистическом смысле) аппроксимирует ожидаемый отклик d . Выбор, в свою очередь, основывается на множестве N независимых, равномерно распределенных примеров обучения:

$$T = \{(d_i, x_i)\}_{i=1}^N \quad (9)$$

Каждая пара выбирается обучаемой машиной из множества T с некоторой обобщенной функцией распределения вероятности $P_{XD}(d|x)$, которая, как и другие функции распределения, фиксирована, но неизвестна.

Рассмотрим задачу обучения с учителем как задачу аппроксимации, состоящую в нахождении функции $F(x, w)$, которая наилучшим образом приближает желаемую функцию $f(x)$.

Пусть $L(d, F(x, w))$ – мера потерь между желаемым откликом d , соответствующим входному вектору x , и откликом $F(x, w)$, сгенерированным обучаемой машиной. Тогда ожидаемая величина потерь определяется функционалом среднего риска:

$$R(w) = \int L(d, F(x, w)) dP_{XD}(d|x) \quad (10)$$

Традиционно выделяют три способа решения задачи минимизации функционала среднего риска:

- восстановление распределения вероятностей,
- построение рекуррентной процедуры,
- замена на функционал эмпирического риска.

Все эти способы обладают своими достоинствами и недостатками.

В результате анализа поставлена задача обучения с учителем на основе данных, собранных из ЭИОС ВУЗа и сгруппированных по определенным правилам. Группировка и полнота сбора данных способствуют повышению точности прогнозируемых результатов.

					ПГУ1.090401.49.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

2 Анализ аналогов

Существуют приложения, которые прогнозируют успеваемость студентов на основе их оценок за предыдущий период обучения. Эти приложения используют алгоритмы машинного обучения, такие как искусственные нейронные сети и регрессионные модели, для анализа данных и определения вероятности успеха студента в следующем семестре.

2.1 Gradescope

Gradescope (<https://www.gradescope.com>) – это онлайн-платформа, которая помогает педагогам проводить и оценивать тесты и задания, независимо от того, проводятся они онлайн или в классе (рисунок 4).

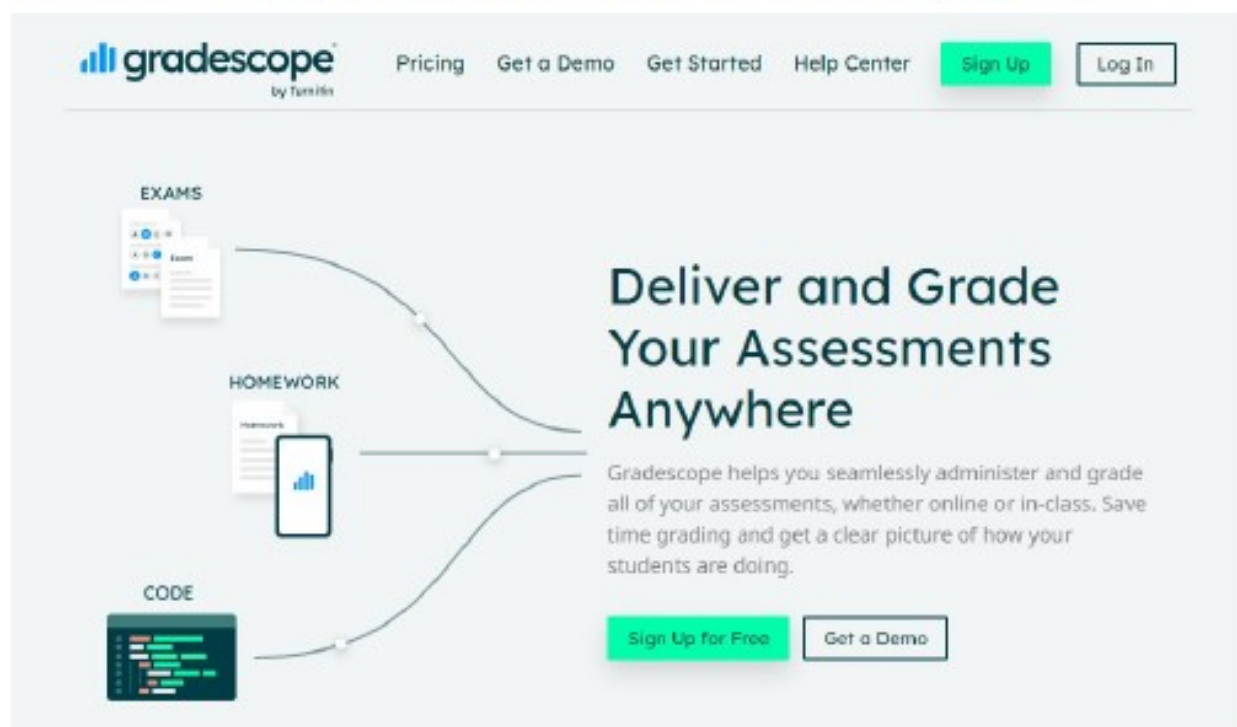


Рисунок 4 – Gradescope

С помощью Gradescope учителя могут оценивать все предметы и задания, включая задания с переменной длиной и задания с фиксированным шаблоном. Они также могут обрабатывать запросы на переоценку онлайн и видеть подробную статистику, чтобы понять, что знают их студенты и предоставляет прогнозную оценку успеваемости на основе этой информации.

2.2 MyEdu

MyEdu (<https://www.myeducomm.com>) – приложение для студентов, которое использует нейронные сети для анализа и предсказания успеваемости на основе исторических данных и текущих результатов студента (рисунок 5).



Рисунок 5 – MyEdu

Основной целью MyEdu является помощь студентам в достижении успеха в учебе. Платформа предлагает различные инструменты и ресурсы для организации академической жизни студентов. Студенты могут создавать свое личное расписание, управлять заданиями и проектами, записывать заметки, а также получать доступ к материалам и ресурсам, которые помогут им в учебе.

Помимо этого, MyEdu также предоставляет студентам возможность управлять своими академическими данными. Студенты могут загружать транскрипты оценок, резюме и портфолио на платформу, чтобы хранить их в одном месте и иметь быстрый и удобный доступ к ним.

Для учителей MyEdu предлагает инструменты для создания и организации учебных планов, автоматического выставления оценок и управления заданиями. Учителя могут также обмениваться материалами с другими преподавателями через платформу, чтобы обеспечить синхронизацию и сотрудничество в учебном процессе.

2.3 Dreambox

Dreambox (<https://www.dreambox.com>) – это приложение для обучения математике, которое использует нейронные сети для индивидуального прогнозирования успеваемости каждого ученика и адаптации материала под его потребности (рисунок 6).

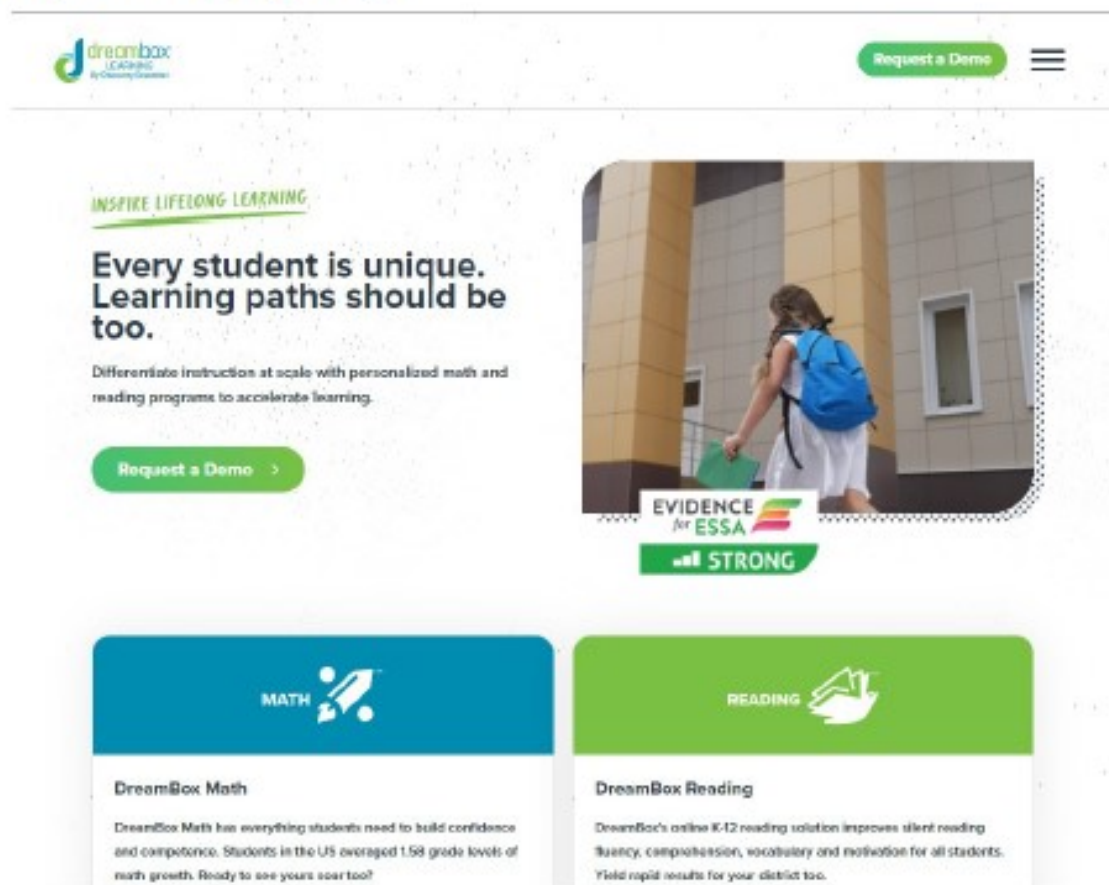


Рисунок 6 – Dreambox

2.4 SmartSparrow

SmartSparrow (<https://www.smartsparrow.com>) – образовательная платформа, которая использует нейронные сети для адаптации курсов под потребности каждого студента и прогнозирования его успеваемости (рисунок 7).

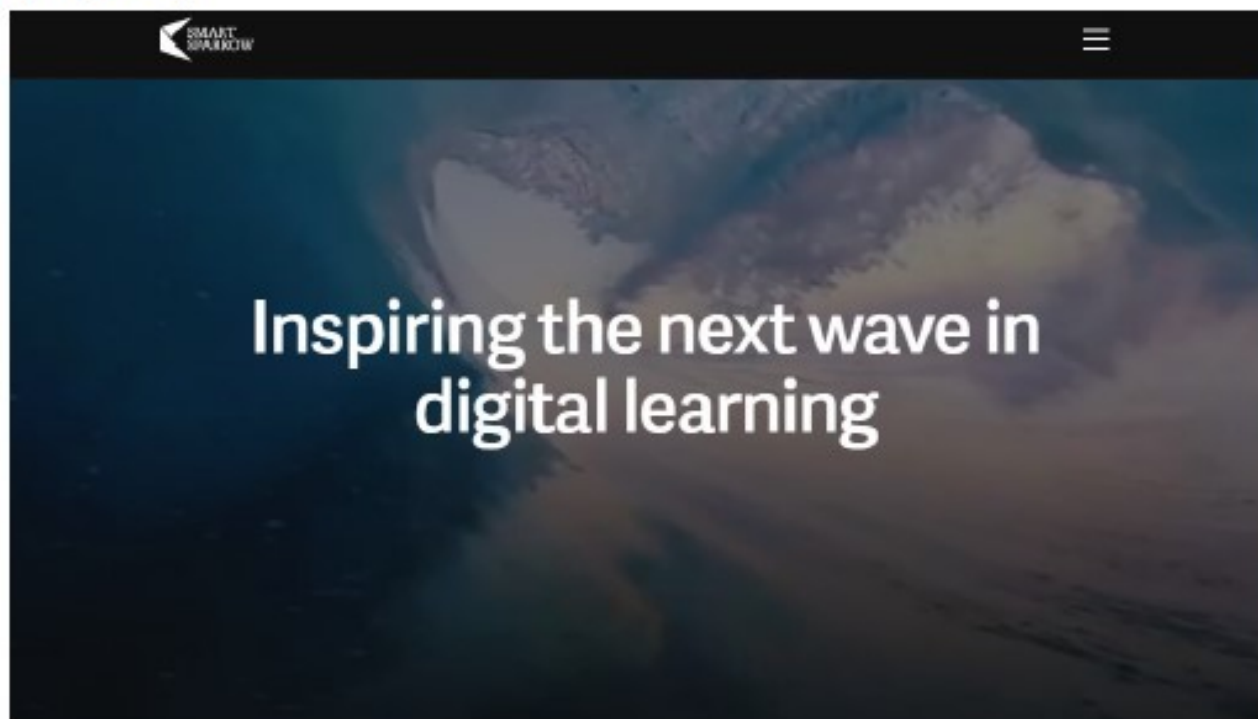


Рисунок 7 – SmartSparrow

Программные инструменты SmartSparrow, известные под общим названием платформа адаптивного электронного обучения, представляют собой веб-пакет, который разрабатывает адаптивный учебный контент и приложения, предоставляет этот материал учащимся и анализирует, как учащиеся усваивают его на основе своих ответов на материал. Платформа реализует интеллектуальную систему обучения, которая сочетает моделирование на основе ограничений с отслеживанием модели.

Изм.	Лист	№ докум.	Подпись	Дата

2.5 Статистика Успеваемости

Статистика Успеваемости (СтатУс: <https://statusapp.ru/>)— приложение, которое помогает ученикам и родителям достигать лучших результатов в учёбе (рисунок 8).

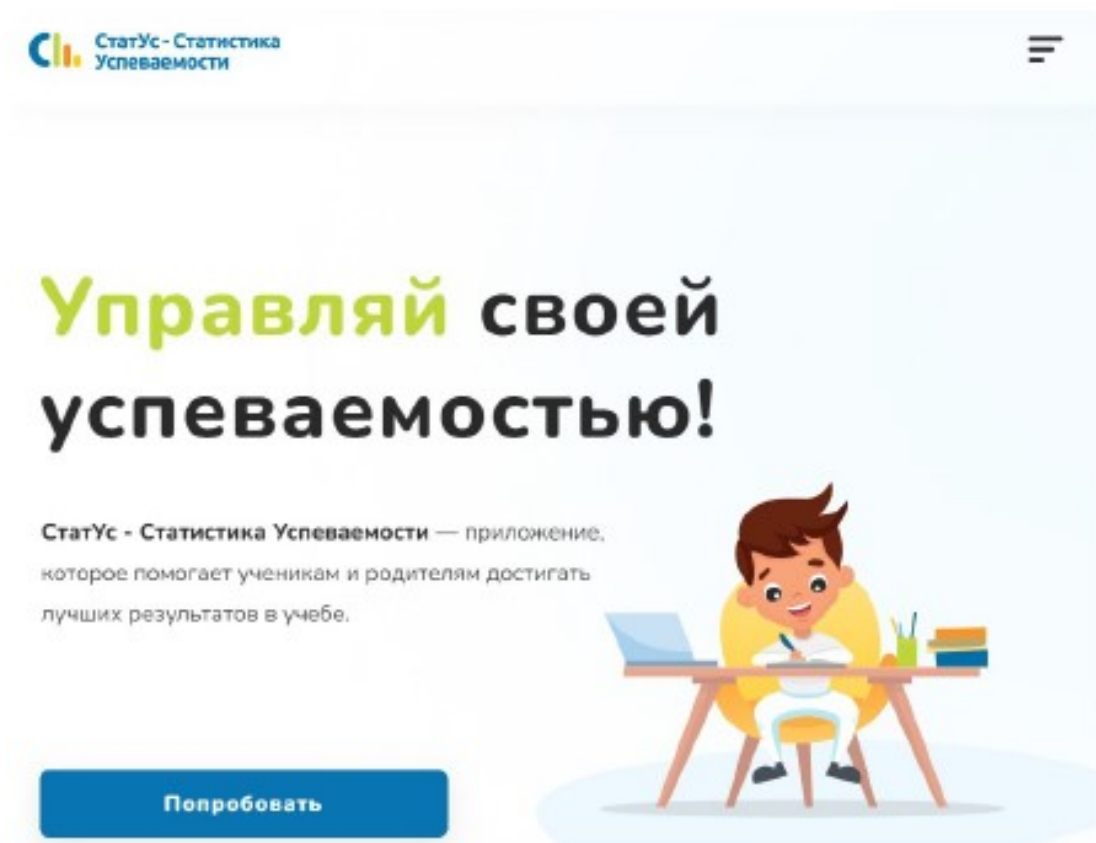


Рисунок 8 –Статистика Успеваемости (СтатУс)

Инструмент для детального анализа успеваемости учащегося с целью заинтересовать его, пробудить желание соревноваться с одноклассниками и мотивировать на улучшение результатов в учебе. Приложение отображает основные показатели и дает общую оценку успеваемости.

Это только некоторые примеры приложений, которые используют нейронные сети для прогнозирования успеваемости. С развитием искусственного интеллекта можно ожидать, что появятся новые интересные проекты в этой области.

3 Анализ и требования к используемым данным для обучения

Проводя поиск необходимых данных для прогнозирования успеваемости, были проанализированы многочисленные статьи по данной тематике. Большинство авторов, в частности [12,13], сходятся во мнении, что только оценочной базы за прошедший период не достаточно. Необходимо также учитывать посещаемость студента, общее количество задолженностей, проводить входное тестирование по предметам.

В статье [12] прогнозирование строилось для конкретного предмета на основе данных об успеваемости в смежных дисциплинах. Применялась радиальная модель нейронной сети. Учитывались такие данные как: средняя оценка по предшествующим смежным дисциплинам, результаты оценочного тестирования, наличие свободного посещения и имеющиеся задолженности. Результаты прогнозирования дали погрешность в 4%, что можно считать хорошим итогом.

В работе [13] нейронные сети применялись для прогнозирования группы риска по данным успеваемости студентов. Прогноз строился для студентов первого курса, в исследовании учитывались баллы ЕГЭ; иностранный язык, изучаемый в школе; наличие или отсутствие социальной стипендии; форма обучения; преподаватели по определенным дисциплинам и др. Авторы объясняют такой набор признаков тем, что они максимально позволяют описать образ первокурсника. В работе решалась задача бинарной классификации, модель определяла принадлежность студента к группе риска.

В реальной жизни поведение людей, в частности студентов, бывает довольно иррациональным. Но, по наблюдениям, наибольшего успеха в обучении достигают студенты, которые реже пропускают занятия, то есть их посещаемость стремится к 100%. Также в эту категорию попадают обучающиеся, следящие за графиком выполнения работ, то есть вовремя сдающие выполненные задания по дисциплинам. Конечно, есть студенты, обучающиеся самостоятельно, но таких гораздо меньшее количество.

Не всегда есть возможность получить от преподавателей сведения по входному тестированию, а также не всегда его проводят. Поэтому получение подобного показателя является сложной, почти невыполнимой задачей. В следствии чего от этого показателя решено было отказаться.

Анализ показателей позволил выявить, что данными для нейронной сети должны стать следующие показатели:

- набор оценок прошлого семестра,
- средняя оценка по предыдущему семестру,
- посещаемость по предыдущему семестру,
- посещаемость в текущем семестре,
- количество задолженностей по дисциплинам прошлого семестра.

Количество входных параметров составило 17 признаков.

На выходе планируется получить характеристику: будет ли ученик отличником (средний балл $> 4,75$), хорошистом ($4,75 \geq$ средний балл $> 3,75$), троечником ($3,75 \geq$ средний балл $> 2,75$) или неуспевающим студентом (средний балл $\leq 2,75$).

Для качественного прогнозирования данные должны быть собраны в достаточном количестве и подготовлены. Должны быть устранены явные выбросы и отсутствующие данные, а также произведена их нормализация.

3.1 Сбор данных для модели

Прогноз напрямую зависит от качества и количества данных. Сбор данных производился из разных источников, а также рассматривались готовые датасеты. Если набор данных подходящий, то и модель будет хорошей. Поэтому процесс сбора данных – задача, которая занимает много времени.

Первоначально набор данных представлял из себя выгрузку из ЭИОС (электронной информационной среды Пензенского государственного университета). Из преимуществ прямой выгрузки из информационной системы можно выделить достоверность данных, простоту и скорость

подготовки. Но также в данном наборе мы можем видеть пропуски и избыточность информации.

Данные, предоставленные организацией, в нашем случае Пензенским государственным университетом, были собраны путем загрузки за несколько лет, представлены на рисунке 9. Для задач прогнозирования и классификации такие данные подойдут, но можно увидеть некоторые недостатки в этом подходе: наличие пропусков, дублирования, недостаточное количество данных.

№	А	В	С	Д	Е	Ж	З	И	К
1	Ю	учащего	Название дисциплины	Тип отчетности	Курс	Семестр	Оценка	Дата выставлено	Дата выставления (по факту выставления)
2	2128152	Интеллектуальные системы	Экзамен		1	1	отл.(5)	27.01.2022	27.01.2022
3	2128152	Информационное общество и проблемы прикладной	Зачет		1	1	зачет	29.12.2021	19.01.2022
4	2128152	Информационное общество и проблемы прикладной	Экзамен		1	1	отл.(5)	19.01.2022	19.01.2022
5	2128152	Информационное общество и проблемы прикладной	Нурсовой проект		1	1	отл.(5)	29.12.2021	19.01.2022
6	2128152	Математическое моделирование	Зачет		1	1	зачет	10.01.2022	01.02.2022
7	2128152	Математическое моделирование	Экзамен		1	1	отл.(5)	11.01.2022	01.02.2022
8	2128152	Технические средства компьютерного дизайна	Экзамен		1	1	отл.(5)	15.01.2022	15.01.2022
9	2128152	Художественные основы интерфейсов и информации	Зачет с оценкой		1	1	отл.(5)	27.12.2021	05.01.2022
10	2128152	Английский язык	Зачет		1	2	зачет	07.08.2022	07.08.2022
11	2128152	Методология и технология проектирования информа	Зачет		1	2	зачет	10.06.2022	05.07.2022
12	2128152	Методология и технология проектирования информа	Экзамен		1	2	отл.(5)	22.06.2022	05.07.2022
13	2128152	Методология и технология проектирования информа	Нурсовой проект		1	2	отл.(5)	09.06.2022	05.07.2022
14	2128152	Планирование и организация научных исследований	Зачет		1	2	зачет	14.06.2022	14.06.2022
15	2128152	Программы в среде компьютерного дизайна	Зачет		1	2	зачет	13.06.2022	13.06.2022
16	2128152	Программы в среде компьютерного дизайна	Экзамен		1	2	отл.(5)	14.06.2022	14.06.2022
17	2128152	Производственная практика (научно-исследовательск	Зачет с оценкой		1	2	отл.(5)	10.06.2022	06.07.2022
18	2128152	Разработка фирменного стиля	Зачет		1	2	зачет	16.06.2022	16.06.2022
19	2128152	Разработка фирменного стиля	Экзамен		1	2	отл.(5)	05.07.2022	05.07.2022
20	2128152	Учебная (технологическая) (проектно-технологическ	Зачет с оценкой		1	2	отл.(5)	14.07.2022	14.07.2022
21	2128152	Философия и методология научных исследований	Зачет		1	2	зачет	26.05.2022	08.06.2022
22	2128152	Оценка и обеспечение информационной безопасности	Зачет		2	3	зачет	28.12.2022	23.01.2023
23	2128152	Оценка и обеспечение информационной безопасности	Экзамен		2	3	отл.(5)	23.01.2023	23.01.2023
24	2128152	Производственная практика (научно-исследовательск	Зачет с оценкой		2	3	отл.(5)	10.01.2023	23.01.2023
25	2128152	Технический английский язык	Экзамен		2	3	отл.(5)	10.01.2023	18.01.2023
26	2128152	Управление проектами в профессиональной сфере	Экзамен		2	3	отл.(5)	14.01.2023	26.01.2023
27	2128152	Управление разработкой программных средств комп	Зачет		2	3	зачет	23.12.2022	23.12.2022
28	2128152	Управление разработкой программных средств комп	Нурсовой проект		2	3	отл.(5)	24.12.2022	10.01.2023
29	2128152	Управление разработкой программных средств комп	Экзамен		2	3	отл.(5)	18.01.2023	18.01.2023
30	2361510	Вычислительные системы	Зачет с оценкой		1	1	отл.(5)	30.12.2021	30.12.2021
31	2361510	Интеллектуальные методы проектирования	Экзамен		1	1	отл.(5)	11.01.2022	12.01.2022
32	2361510	Интеллектуальные системы	Экзамен		1	1	отл.(5)	16.01.2022	16.01.2022
33	2361510	Методология системного анализа и синтеза в система	Экзамен		1	1	отл.(5)	24.01.2022	26.01.2022
34	2361510	Методы оптимизации	Экзамен		1	1	отл.(5)	19.01.2022	25.01.2022

Рисунок 9 –Первый набор данных

Данные были изучены на предмет адекватности и полезности. Была выявлена необходимость в предобработке данных:

- обработать пропуски и выбросы,
- устранить избыточность данных,
- устранить аномалии,
- масштабировать данные.

Работа с первым набором данных приведена ниже.

```

import csv

def print_hi(name):
    print(f'Hi, {name}')

if __name__ == '__main__':
    listExam = []
    listTest = []
    listKurosov = []
    len_list = 0

    with open("ue_red.csv", "r", newline='') as r_file:
        file_reader = csv.reader(r_file, delimiter=";")
        for row_read in file_reader:
            len_list += 1
            if (row_read[2] == "Экзамен"):
                if (row_read[1] not in listExam):
                    listExam.append(row_read[1])
            elif (row_read[2] == "Зачет" or (row_read[2] == "Зачетсоценкой"):
                if (row_read[1] not in listTest):
                    listTest.append(row_read[1])
            elif (row_read[2] == "Курсовойпроект"):
                if (row_read[1] not in listKurosov):
                    listKurosov.append(row_read[1])

    list_temp = []
    reset_list = []

    for i in range(len(listTest) + len(listExam) + len(listKurosov) + 5):
        list_temp.append('')
    for i in range(len(listTest) + len(listExam) + len(listKurosov) + 5):
        reset_list.append('')

    with open("output.csv", "w", newline='') as w_file:

```

```

file_writer = csv.writer(w_file, delimiter=";")
with open("ue_red.csv", "r", newline='') as r_file:
    file_reader = csv.reader(r_file, delimiter=";")
    count = 0
    id = None
    line = ''
    semestr = None

    for row_read in file_reader:

        if count == 0:
            start_line = [row_read[0], row_read[4], 'Зачет']
            start_line.extend(listTest)
            start_line.append('Экзамен')
            start_line.extend(listExam)
            start_line.append('Курсовой проект')
            start_line.extend(listKursov)
            file_writer.writerow(start_line)
            elif id == row_read[0]: # Если id не поменялся
                if semestr == row_read[4]:
                    if (row_read[2] == "Зачет") or (row_read[2] == "Зачетс оценкой"):
                        ind = listTest.index(row_read[1]) + 3
                        list_temp[ind] = row_read[5]
                    elif row_read[2] == "Экзамен":
                        ind = listExam.index(row_read[1]) + len(listTest) + 4
                        list_temp[ind] = row_read[5]

                    elif row_read[2] == "Курсовой проект":
                        ind = listKursov.index(row_read[1]) + len(listTest) + len(listExam) + 5
                        list_temp[ind] = row_read[5]
                    else:
                        file_writer.writerow(list_temp)
                        list_temp = list(reset_list)
                        semestr = row_read[4]

```

```

list_temp[1] = semestr
if (row_read[2] == "Зачет") or (row_read[2] == "Зачетоценкой"):
ind = listTest.index(row_read[1]) + 3
list_temp[ind] = row_read[5]
elif row_read[2] == "Экзамен":
ind = listExam.index(row_read[1]) + len(listTest) + 4
list_temp[ind] = row_read[5]

elif row_read[2] == "Курсовой проект":
ind = listKurosov.index(row_read[1]) + len(listTest) + len(listExam) + 5
list_temp[ind] = row_read[5]
elif id != row_read[0]: # Если id поменялся / new id
if count >= 2:
file_writer.writerow(list_temp)
list_temp = list(reset_list)
id = row_read[0]
semestr = row_read[4]
list_temp[0] = id
list_temp[1] = semestr
if (row_read[2] == "Зачет") or (row_read[2] == "Зачетоценкой"):
ind = listTest.index(row_read[1]) + 3
list_temp[ind] = row_read[5]
elif row_read[2] == "Экзамен":
ind = listExam.index(row_read[1]) + len(listTest) + 4
list_temp[ind] = row_read[5]

elif row_read[2] == "Курсовой проект":
ind = listKurosov.index(row_read[1]) + len(listTest) + len(listExam) + 5
list_temp[ind] = row_read[5]
count += 1
if count == len_list:
file_writer.writerow(list_temp)

```

Второй набор данных прошел следующие этапы: выборка, очистка, форматирование. Результат обработки программой представлен на рисунке 10. В этом наборе также отмечены недостатки: отсутствие данных, пустые значения. Этих данных будет недостаточно для обучения модели. Выявлена необходимость в поиске дополнительных признаков, таких как посещаемость, наличие долгов, и других.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	
1	Единица	Семестр	Зачет	Информация	Математика	Химия	Астрономия	Методы	Экономика	Программы	Программы	Разработка	Разработка	Информатика	Информатика	Системы	Управление	Воспитание	Мировая	Технологии	Учебные	Цифровые	Мировые	Современные	Технологии
2	2020/21	1	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
3	2020/21	2	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
4	2020/21	3	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
5	2020/21	4	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
6	2020/21	5	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
7	2020/21	6	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
8	2020/21	7	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
9	2020/21	8	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
10	2020/21	9	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
11	2020/21	10	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
12	2020/21	11	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
13	2020/21	12	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
14	2020/21	13	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
15	2020/21	14	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
16	2020/21	15	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
17	2020/21	16	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
18	2020/21	17	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
19	2020/21	18	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
20	2020/21	19	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
21	2020/21	20	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
22	2020/21	21	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
23	2020/21	22	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
24	2020/21	23	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
25	2020/21	24	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
26	2020/21	25	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
27	2020/21	26	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
28	2020/21	27	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
29	2020/21	28	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
30	2020/21	29	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
31	2020/21	30	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
32	2020/21	31	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет
33	2020/21	32	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет	зачет

Рисунок 10 – Второй набор данных

Следующий подход, комбинированный, оказался более удачным. Основные данные получены путем выгрузки из ЭИОС, недостающие данные сформированы путем парсинга сайта ПГУ/ручного сбора (рисунок 11). Для задач прогнозирования этот способ оказался наиболее эффективным в подготовке набора данных. Данные уже содержат не только оценки, но и признаки, учитывается посещаемость студентом занятий.

В процессе подготовки были выделены признаки, произведена очистка данных, выборка данных, интеграция и форматирование. Были исправлены ошибки, опечатки, исключено дублирование данных, пропуски. Данные были объединены из разных источников, произведена агрегация параметров, новые значения были получены путем суммирования, нахождения среднего значения из набора существующих записей. Была произведена сортировка

№	Имя	Информация о студенте										Информация о родителях	Информация о семье	Информация о здоровье	Информация о социальном статусе	Информация о успеваемости	Информация о поведении
		1	2	3	4	5	6	7	8	9	10						
1	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
2	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
3	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
4	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
5	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
6	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
7	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
8	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
9	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
10	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
11	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
12	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
13	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
14	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
15	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
16	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
17	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
18	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
19	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
20	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
21	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
22	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
23	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
24	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
25	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
26	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
27	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
28	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
29	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
30	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
31	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
32	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
33	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
34	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
35	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
36	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
37	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
38	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
39	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
40	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
41	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
42	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
43	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
44	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
45	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
46	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
47	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
48	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
49	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50	Александр Александрович	1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Рисунок 11 –Третий набор данных

в определенном порядке, выбраны допустимые значения для каждого столбца, проверка полей на влияние одних на другие, определены типы данных.

Выводы по анализу: Данные могут быть использованы. Но точность модели будет не высока, так как данные содержат малое количество показателей, влияющих на успеваемость студентов. При дополнении данного набора данными о жизни студента: возраст, наличие работы, количество времени на самообразование, рейтинг здоровья, текстового описания причин неуспеваемости и других. Правильность предсказаний увеличится.

3.2 Формирование датасета

Как говорилось в предыдущем пункте, были выявлены наиболее доступные для сбора данные, которые удовлетворяют поставленной задаче прогнозирования.

Для удобства пользователя был разработан программный код, позволяющий автоматически преобразовывать данные в необходимый формат для дальнейшего прогнозирования. Определены параметры, которые должны присутствовать в файле:

- фамилия, имя, отчество,
- номер предыдущего семестра,
- набор оценок по предметам (колонки с номерами от 1 до 12). Причем, полное заполнение не обязательно, приложение обработает этот момент,
- количество долгов предыдущего семестра,
- посещаемость в предыдущем семестре,
- средняя оценка предыдущего семестра (формируется автоматически),
- посещаемость в текущем семестре.

Шапка формируемого документа должны выглядеть следующим образом (см. рисунок 12).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	ФНО	номер семестра	1	2	3	4	5	6	7	8	9	10	11	12	количество заданных	посещаемость	средний балл	посещаемость следующего семестра
1																		
2	Агаджан Павел Александрович	1	зачет	зачет	зачет	зачет	зачет	зачет			удовл	удовл	удовл	зачет	0	74		81
3	Аликина Анастасия Сергеевна	1	зачет	зачет	зачет	зачет	зачет	зачет			удовл	хорошо	удовл	зачет	0	65		96
4	Аликин Олег Сергеевич	1	зачет	зачет	зачет	зачет	зачет	зачет			удовл	удовл	удовл	зачет	0	64		77
5	Вадонин Михаил Александрович	1	зачет	зачет	зачет		зачет	зачет		зачет	удовл	удовл	удовл	хорошо	0	88		80
6	Баранова Раиса Николаевна	1	зачет	зачет	зачет		зачет	зачет	зачет		хорошо	хорошо	хорошо	хорошо	0	92		96
7	Брандас Дмитрий Александров	1	зачет	зачет	зачет		н/я	зачет	зачет		удовл	удовл	н/я	зачет	2	56		76
8	Брыкина Владимир Павлович	1	зачет	зачет	зачет	зачет	зачет	зачет			удовл	удовл	удовл	н/я	1	49		80
9	Натанов Владимир Александрович	1	зачет	зачет	зачет	зачет	зачет	зачет			удовл	удовл	удовл	зачет	0	44		87

Рисунок 12 – Шапка файла набора данных для обработки

Для корректной работы файл должен иметь название 'newBook.xlsx'. Ниже приведен код для загрузки файла:

```
import pandas as pd
import numpy as np

# Загрузка данных из файла Excel
data = pd.read_excel('newBook.xlsx')
```

После чего происходит замена строковых выражений числовыми данными, соответствующими текущей системе оценивания. А именно: отлично – 5, хорошо – 4, удовлетворительно – 3, такие отметки как «не допущен», «н/я», «незачет» считаются нулём. Отметка «зачет» заменяется пустым значением и в дальнейшем будет заменена величиной средней оценки по семестру. Для этих целей была реализована функция `normalased_data(data)`, принимающая загруженный файл в качестве параметра:

```
def normalased_data(data):
    norm_data = data.copy()
    norm_data.replace('отлично', 5 ,inplace=True)
    norm_data.replace('хорошо', 4 ,inplace=True)
    norm_data.replace('удовл', 3 ,inplace=True)
    norm_data.replace(['недопущен','н/я', 'незачет'], 0 ,
inplace=True)
    norm_data.replace('зачет', np.nan ,inplace=True)
```

Средняя оценка по семестру вычисляется автоматически и подставляется в необходимую колонку, а именно – «текущий средний балл»:

```
#Вычисление среднего значения по строке
averages = []
table1 = norm_data.iloc[:,2:-4].values
for row in table1:
    numbers = [float(num) for num in row if num != 'nan']
    avg = np.nanmean(numbers) if len(numbers) > 0 else np.nan
    averages.append(avg)

#Переформатирование массива в столбец для подстановки в датасет
vec = pd.Series(np.array(averages).T)

#Вычисление количества пустых ячеек
df_numeric = norm_data.select_dtypes(include=[np.number])
numeric_cols = df_numeric.columns.values
```

Выполняется вставка по столбцам только в те ячейки, в которых отсутствуют значения:

```
for col in numeric_cols:
    if norm_data[col].isnull().sum() > 0:
        norm_data[col].fillna(vec, inplace=True)
    return norm_data
```

После всех преобразований, данные сохраняются в файл, который будет использоваться для входных параметров прогнозирования:

```
normal_data = normalased_data(data)
normal_data.to_excel('new_data.xlsx')
```

4 Реализация работы с нейронной сетью

В виду того, что благодаря ЭИОС мы можем получить и обработать данные, а, следовательно, можно получить ожидаемый результат, разрабатываемые сети будут реализовывать подход обучения «с учителем».

4.1 Реализация с использованием глубокой нейронной сети

В ходе исследования первым решением было использовать глубокую нейронную сеть для прогнозирования успеваемости. Глубокие сети представляют собой мощные алгоритмы машинного обучения, которые могут моделировать сложные взаимосвязи и обучаться на больших объемах данных. Они состоят из множества слоев нейронов, каждый из которых выполняет определенные вычисления и передает данные на следующий слой.

Одной из основных особенностей является способность извлекать сложные признаки из данных на разных уровнях абстракции.

4.1.1 Реализация глубокой нейронной сети

В целом, глубокие нейронные сети представляют собой мощный инструмент машинного обучения, который позволяет создавать сложные модели для решения разнообразных задач.

В ходе проектирования разработан класс, реализующий пять слоев нейронов в сети. Слои сумматоров чередуются с функциями активации:

```
import numpy as np
import torch
import torch.optim
import torch.nn as nn

class Model(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2,
                 hidden_size3, hidden_size4, output_size):
        nn.Module.__init__(self)
        self.layers = nn.Sequential(nn.Linear(input_size, hidden_size1),
                                     # слой линейных сумматоров
                                     nn.ReLU(), # функция активации
```

```

nn.Linear(hidden_size1, hidden_size2),
nn.ReLU(),
nn.Linear(hidden_size2, hidden_size3),
nn.Tanh(),
nn.Linear(hidden_size3, hidden_size4),
nn.ReLU(),
nn.Linear(hidden_size4, output_size),
nn.Softmax(),
    )

# прямойпроход
def forward(self, x):
    return self.layers(x)

```

Функция активации ReLU (Rectified Linear Activation) в Python является наиболее распространенным выбором функции активации для глубокого обучения. ReLU обеспечивает качественные результаты и в то же время очень эффективен с точки зрения вычислений.

Математическая формула выглядит следующим образом:

$$f(x) = \max(0, x) = \begin{cases} x_i & \text{if } x_i > 0 \\ 0 & \text{if } x_i < 0 \end{cases} \quad (11)$$

К ReLU применимо такое понятие, как «сила активации». Но главная особенность, которая делает ReLU привлекательнее других функций активации — простое вычисление производной:

$$d(\text{ReLU}(x)) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases} \quad (12)$$

Следовательно, ReLU вычислительно эффективнее других функций активации (гиперболический тангенс, сигмоида и др.).

Функция Softmax обычно используется на последнем слое сети, в виду того, что она преобразует вектор с последнего слоя в вектор «вероятностей»: каждый элемент вектора лежит от 0 до 1, а их сумма равна 1. Она не меняет размерность вектора.

$$Softmax_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (13)$$

При поиске производной определяем: так как x – вектор, и в знаменателе всегда присутствуют все его элементы, то при получении производной получим якобиан:

$$J_{Softmax} = \begin{cases} x_i - x_i \cdot x_j, i = j \\ -x_i \cdot x_j, i \neq j \end{cases} \quad (14)$$

Вектор производных dz приходит от предыдущего слоя (обычно это функция ошибки). Если dz пришел от функции ошибки на MNIST, следовательно dz – 10-мерный вектор. В этом случае якобиан имеет размерность 10×10 . Для получения dz_{new} , который пойдет далее, на предыдущий слой от конца к началу сети при обратном распространении ошибки, нужно умножить dz на $J_{Softmax}$ (строка на столбец) [16]:

$$dz_{new} = dz \times J_{Softmax} \quad (15)$$

На выходе получается 10-мерный вектор производных.

На выходе каждого полносвязного слоя NN строго обязательна нелинейная функция активации. Без неё невозможно обучить нейронную

сеть. При обратном распространении ошибки берется производная по каждому весу матрицы.

4.1.2 Реализация процесса обучения нейронной сети

Для обучения сети написан модуль, реализующий обучение и поверку на достоверность результатов.

Данные считываются из файла. Выбираются входные данные и указания учителя. Далее разбиваем выборку на обучающую и тестовую:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, y,
test_size = 0.3)
```

Формируем тестовые и тренировочные выходные векторы:

```
Y_tr = np.zeros((Y_train.shape[0], np.unique(Y_train).shape[0]))
for i in np.unique(Y_train):
Y_tr[:,i-2] = np.where(Y_train == i, 1, 0).reshape(1,-1)
```

```
Y_te = np.zeros((Y_test.shape[0], np.unique(Y_test).shape[0]))
for i in np.unique(Y_test):
Y_te[:,i-2] = np.where(Y_test == i, 1, 0).reshape(1,-1)
```

Определяем количество входных и выходных сигналов, а также размеры скрытых слоев:

```
input_size = X.shape[1]
output_size = Y_tr.shape[1] if len(Y_tr.shape) else 1
hidden_size1 = int(input_size * 3)
hidden_size2 = int(input_size * 5)
hidden_size3 = int(input_size * 2)
hidden_size4 = int(input_size * 1)
```

Далее определяется шаг и количество эпох обучения:

```
learning_rate = 0.001
num_epochs = 300000
```

Задаем модель, определяем функцию ошибки и оптимайзер:

```
model = Model(input_size, hidden_size1, hidden_size2,
hidden_size3, hidden_size4, output_size)
loss = nn.MSELoss()
```

```
optimizer = torch.optim.SGD(model.parameters(),
lr=learning_rate)
```

MSELoss – среднеквадратичная ошибка. Вычисляется квадратная разница между значениями в тензоре прогнозирования и значениями целевого тензора. При этом относительно большие различия оказываются больше, в то время как относительно небольшие различия оказываются меньше. Вычисление производится по формуле:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (16)$$

SGD – оптимизатор Stochastic Gradient Descent (стохастический градиентный спуск). Очень эффективный метод подгонки линейных регрессоров и классификаторов под выпуклые функции потерь.

```
for epochs in range(num_epochs):
optimizer.zero_grad()
out = model(torch.from_numpy(X_train.astype(np.float32)))
error = loss(out, torch.from_numpy(Y_tr.astype(np.float32)))
error.backward()
optimizer.step()
if (epochs + 1) % 1000 == 0:
print(f'Epoch {epochs + 1}, loss: {error.item():.3f}')
```

Для снижения оценочной ошибки до 9% потребовалось порядка 1000000 эпох обучения.

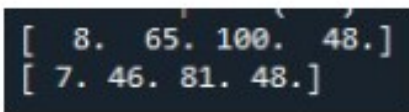
4.1.3 Реализация процесса оценки обучения нейросети

Для получения достоверных результатов ошибки прогнозирования был разработан модуль оценочного тестирования:

```
pred = model.forward(torch.from_numpy(X_train.astype  
(np.float32))).detach().numpy()  
err = sum(abs((pred>0.5)-Y_tr))  
print(err)
```

```
pred = model.forward(torch.from_numpy(X_test.astype  
(np.float32))).detach().numpy()  
err = sum(abs((pred>0.5)-Y_te))  
print(err)
```

Результатом выполнения данного кода будет количество ошибочных результатов по каждому классу прогнозируемых значений. Как видно из полученных результатов – количество ошибочных очень велико (рисунок 13).



```
[ 8. 65. 100. 48.]  
[ 7. 46. 81. 48.]
```

Рисунок 13 – результаты тестирования NN

Вывод по использованию данной сети: глубокая нейронная сеть очень хорошо обучается на больших объемах данных, которыми мы не располагаем. В следствии чего точность прогнозируемых результатов крайне низка, порядка 68%. В виду вышеизложенного – нет целесообразности использовать данную сеть в перспективе.

4.2 Реализация с использованием RNN

В ходе исследования были проанализированы различные модели нейронных сетей. Выбор остановился на разработке рекуррентной нейронной сети (RNN) ввиду того что РНС является динамической системой. [7]

RNN широко применяются в анализе последовательных данных. Они могут анализировать временные ряды, например: финансовые или погодные данные, а также медицинские измерения. Они способны выявлять закономерности и тренды, а главное – прогнозировать будущие значения.

4.2.1 Реализация RNN

Базовые рекуррентные нейронные сети имеют проблему затухающего градиента, а также неспособны эффективно учить долгосрочные зависимости. Для преодоления этих ограничений были разработаны более сложные архитектуры, такие как LSTM (Long Short-Term Memory). Они имеют механизмы для преодоления этих проблем.

Для создания рекуррентной нейронной сети (RNN) с использованием Python для прогнозирования класса успеваемости студентов, мы будем использовать несколько основных модулей Python: torch, torch.nn, и torch.optim. Модель состоит из нескольких слоев, включая рекуррентный слой (LSTM) и линейный слой для классификации.

В LSTM слое (longshort-termmemory), кроме скрытого состояния h между ячейками передаётся "состояние памяти" c . Размерность H этого вектора совпадает с размерностью скрытого состояния h . Векторы регулируют то, какие признаки надо запомнить или забыть при передаче к следующей ячейке, что улучшает долгосрочную память. [14]

Внутри LSTM-ячейки присутствует четыре линейных слоя с H нейронами. Три из них имеют сигмоидную активацию σ (на выходе вектор со значениями $[0...1][0...1]$) и один слой с гиперболическим тангенсом $\tanh$: $[-1...1][-1...1]$. Структура сети представлена на рисунке 14.

$$\begin{cases} \mathbf{h}^{(t)} = \tanh \mathbf{c}^{(t)} \odot \mathbf{f}(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}), \\ \mathbf{c}^{(t)} = \mathbf{g}(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}, \mathbf{c}^{(t-1)}) \end{cases} \quad (17)$$

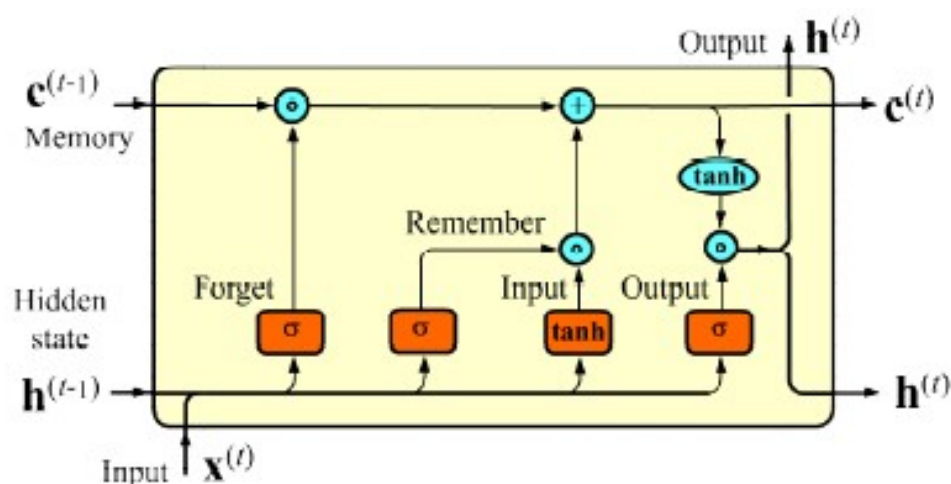


Рисунок 14 –Ячейка LSTM

Сначала происходит изменение значения входящего в ячейку вектора памяти $\mathbf{c}^{(t-1)}$. Сначала $\mathbf{x}^{(t)}$, $\mathbf{h}^{(t-1)}$ попадают в гейт забывания (полносвязный слой Forget с сигмоидой). Размерность выхода данного слоя соответствует H (аналогично как у $\mathbf{h}$ и $\mathbf{c}$). Этот вектор без свёртки умножается на компоненты предыдущего вектора памяти $\mathbf{c}^{(t-1)}$. Предполагается, что при умножении некоторые признаки в $\mathbf{c}^{(t-1)}$ забываются (при умножении на 0), а какие-то продвигаются дальше (при умножении на 1).

Скрытое состояние ячейки $\mathbf{h}$ вычисляется подобно RNN. После чего он перемножается на $\tanh(\mathbf{c})$.

Аналогичным образом работают гейт, реализующие запоминание. Особенности, обязательные к запоминанию добавляются в вектор $\mathbf{c}$. Слой с $\tanh$ $[-1...1]$ формирует "фичи-кандидаты", а слой с сигмоидом $[0...1]$ отвечает за усиление или ослабление роли запоминаемой

фичи. Вычисления в LSTM-ячейке аналитически выглядят следующим образом:

$$\begin{cases} \mathbf{F} = \sigma(\mathbf{x}^{(t)} \mathbf{W}_f + \mathbf{h}^{(t-1)} \mathbf{H}_f + \mathbf{b}_f), \\ \mathbf{I} = \sigma(\mathbf{x}^{(t)} \mathbf{W}_i + \mathbf{h}^{(t-1)} \mathbf{H}_i + \mathbf{b}_i), \\ \mathbf{R} = \tanh(\mathbf{x}^{(t)} \mathbf{W}_r + \mathbf{h}^{(t-1)} \mathbf{H}_r + \mathbf{b}_r), \\ \mathbf{O} = \sigma(\mathbf{x}^{(t)} \mathbf{W}_o + \mathbf{h}^{(t-1)} \mathbf{H}_o + \mathbf{b}_o), \end{cases} \quad \begin{cases} \mathbf{c}^{(t)} = \mathbf{F} \odot \mathbf{c}^{(t-1)} + \mathbf{R} \odot \mathbf{I}, \\ \mathbf{h}^{(t)} = \tanh(\mathbf{c}^{(t)}) \odot \mathbf{O}. \end{cases} \quad (18)$$

Размерности матриц для $E = \dim(x)$, $H = \dim(h)$, $\dim(c)$ равны:

$\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_r, \mathbf{W}_o : (E, H)$; $\mathbf{H}_i, \mathbf{H}_f, \mathbf{H}_r, \mathbf{H}_o : (H, H)$; $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_r, \mathbf{b}_o : (1, H)$.

В Python LSTM-сеть реализуется классом `torch.nn.LSTM`.

Для решения поставленной задачи модель рекуррентной нейронной сети с использованием LSTM модуля:

```
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

# Определение модели RNN
class RNNModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers=1):
        super(RNNModel, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
```

```

        h0      =      torch.zeros(self.num_layers,      x.size(0),
self.hidden_size).to(x.device)
        c0      =      torch.zeros(self.num_layers,      x.size(0),
self.hidden_size).to(x.device)

out, _ = self.lstm(x.unsqueeze(1), (h0, c0))
out = self.fc(out[:, -1, :])
return out

```

4.2.2 Реализация процесса обучения RNN

После реализации основного модуля нейронной сети она подлежит обучению. Для этого был написан модуль, реализующий обучение и проверку на достоверность результатов.

Сначала нужно подключить необходимые модули:

```

import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from RNN_Y import RNNModel

```

Далее происходит загрузка данных из подготовленного заранее файла с обучающими материалами:

```

# Загрузка данных из файла Excel
data = pd.read_excel('newBook_341.xlsx')

```

Следом необходимо провести предобработку данных: произвести извлечение признаков и меток, преобразовать категориальные данные в числовые и произвести нормализацию данных, чтобы избежать чрезмерного влияния одного из признаков на результат:

```

# Извлечение признаков и меток
features      =      data.drop(columns=['ФИО',      'класс',      'величина
класса'])

```

```

labels = data.iloc[:, -1:]

# Преобразование категориальных данных в числовые
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(labels)

# Нормализация данных
scaler = StandardScaler()
features = scaler.fit_transform(features)

```

Далее производится разделение подготовленных данных на наборы тренировочных и тестовых данных:

```

X_train, X_test, y_train, y_test = train_test_split(features,
labels, test_size=0.2, random_state=42)

```

```

class StudentPerformanceDataset(Dataset):
    def __init__(self, data, labels):
        self.data = torch.tensor(data, dtype=torch.float32)
        self.labels = torch.tensor(labels, dtype=torch.long)

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        return self.data[idx], self.labels[idx]

```

Создаются датасеты и загрузчики данных:

```

train_dataset = StudentPerformanceDataset(X_train, y_train)
test_dataset = StudentPerformanceDataset(X_test, y_test)

train_loader = DataLoader(train_dataset, batch_size=32,
shuffle=True)

```

```
test_loader = DataLoader(test_dataset, batch_size=32,
shuffle=False)
```

Далее определяются параметры модели и сама модель:

```
input_size = X_train.shape[1]
hidden_size = 128
output_size = len(label_encoder.classes_)
num_layers = 2

device = torch.device('cuda' if torch.cuda.is_available() else
'cpu')

model = RNNModel(input_size, hidden_size, output_size,
num_layers).to(device)
```

CrossEntropyLoss — это функция потерь, которая используется при многоклассовой классификации.

Она представляет собой комбинацию функции активации Softmax и функции потери Negative Log Likelihood (NLL).

Сеть должна иметь C выходов, т.е. в ошибку из модели входит тензор y_{ia} формы (N, C) , где N – размер батча. Целевыми являются целые числа $y_i \in [0 \dots C-1]$ – это и есть номера классов каждого примера формы (N) . Ошибка для одного i -того примера и $y_i \neq c$ равна:

$$L(y, c) = -w_c \log \left(\frac{\exp y_{ic}}{\sum_{\alpha} \exp y_{i\alpha}} \right) \quad (19)$$

Функция Softmax используется для преобразования вывода последнего слоя нейронной сети в вероятности. Функция NLL Loss применяется для вычисления ошибки между прогнозируемыми вероятностями и истинными

метками. Таким образом, CrossEntropyLoss совмещает в себе Softmax и на выходе сети его ставить не нужно. Поэтому была выбрана именно CrossEntropyLoss:

```
criterion = nn.CrossEntropyLoss()
```

Адам – метод стохастической оптимизации. Алгоритм для оптимизации случайных целевых функций на основе градиента первого порядка, основанный на адаптивных оценках моментов более низкого порядка. Метод прост в реализации, эффективен с точки зрения вычислений, имеет небольшие требования к памяти[15]. Функция может быть записана как функция градиентов на всех предыдущих временных интервалах:

$$v_t = (1 - \beta_2) \sum_{i=1}^t \beta_2^{t-i} \cdot g_i^2 \quad (20)$$

После успешного введения метода Adam в нейросети его окрестили «method of choice» в задачах стохастической оптимизации. Это было обусловлено его успехом в обучении нейронных сетей с нестабильными архитектурами, такими как RNN.

```
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Разработана функция для обучения модели с выводом ошибки на каждой итерации (эпохе обучения):

```
def train_model(model, train_loader, criterion, optimizer,
num_epochs):
    model.train()
    for epoch in range(num_epochs):
        running_loss = 0.0
        for inputs, labels in train_loader:
```

```

        inputs, labels = inputs.to(device),
labels.to(device)

        outputs = model(inputs)
        loss = criterion(outputs, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        if (epoch + 1) % 5 == 0:
            print(f'Epoch {epoch+1}/{num_epochs}, Loss:
{running_loss/len(train_loader):.4f}')

# Обучениемодели
train_model(model, train_loader, criterion, optimizer,
num_epochs=30)

```

После успешного окончания обучения параметры модели необходимо сохранить для последующего использования (рисунок 15):

```
torch.save(model.state_dict(), 'data_0609.pth')
```



```

_, predicted = torch.max(outputs.data, 1)
total += labels.size(0)
correct += (predicted == labels).sum().item()
all_preds.extend(predicted.cpu().numpy())
all_labels.extend(labels.cpu().numpy())

accuracy = 100 * correct / total
print(f'Accuracy: {accuracy:.2f}%')
all_preds = label_encoder.inverse_transform(all_preds)
all_labels = label_encoder.inverse_transform(all_labels)

return all_preds, all_labels, accuracy

```

Произведена оценка модели и вывод прогнозов и фактических значений:

```

preds, labels, accuracy = evaluate_model(model, test_loader,
label_encoder)

```

```

# Вывод результатов прогноза
results = pd.DataFrame({'Predicted': preds, 'Actual':
labels})
print(results)

```

Результаты обучения и оценка точности модели, выведенные в консоль, представлены на рисунке 16.

```

Accuracy: 97.93%
   Predicted  Actual
0          3        3
1          3        3
2          5        5
3          3        3
4          5        5
..         ...     ...
188         4        4
189         3        3
190         5        5
191         3        3
192         4        4
[193 rows x 2 columns]

```

Рисунок 16 – Результаты обучения и оценка точности модели

4.2.4 Реализация процесса прогнозирования

Для возможности получения прогноза на основе загружаемых данных реализована функция `predict()`:

```
# функция для предсказания классов на новых данных
def predict(model, data, label_encoder):
    model.eval()
    data = torch.tensor(data,
dtype=torch.float32).to(device)
    with torch.no_grad():
        outputs = model(data)
        _, predicted = torch.max(outputs.data, 1)
    predicted =
label_encoder.inverse_transform(predicted.cpu().numpy())
    return predicted
```

На основе сформированного файла (см. пп. 2.1.2) можно получить прогнозные данные, используя код ниже. Считываемый датасет не является нормализованным, следовательно, после загрузки файла, его так же необходимо привести к надлежащему состоянию:

```
df = pd.read_excel('new_data.xlsx', index_col=0)

# Предобработка данных
# Извлечение признаков и меток
features1 = df.drop(columns=['ФИО'])

# Нормализация данных
scaler = StandardScaler()
features1 = scaler.fit_transform(features1)

new_preds = predict(model, features1, label_encoder)
print("New predictions:", new_preds)
```

После получения прогнозных значений, их необходимо представить в удобной для просмотра форме. А именно: таблицу, содержащую порядковый номер студента в группе, ФИО, прогнозируемая категория студента.

```

preds_data = df.iloc[:, :1]
preds_data['прогноз'] = new_preds
preds_data['класспрогноза'] = new_preds
preds_data['класспрогноза'].replace(5, 'отличник', inplace=True)
preds_data['класспрогноза'].replace(4, 'ударник', inplace=True)
preds_data['класспрогноза'].replace(3, 'троечник', inplace=True)
preds_data['класспрогноза'].replace(2, 'неуспевающий',
inplace=True)
print(preds_data)

preds_data.to_excel('preds_data.xlsx')

```

Данные, выведенные в консоль и сохраненные в файл, представлены на рисунках 17 и 18 соответственно:

	ФИО	прогноз	класс прогноза
0	Агашин Павел Александрович	3	троечник
1	Алёшин Анатолий Сергеевич	4	ударник
2	Амирян Олег Евгеньевич	4	ударник
3	Вдовин Никита Александрович	4	ударник
4	Карманов Родион Николаевич	5	отличник
5	Кузнецов Дмитрий Александрович	3	троечник
6	Кузьмин Владислав Павлович	3	троечник
7	Нагаев Ярослав Александрович	3	троечник
8	Сидоров Кирилл Андреевич	5	отличник
9	Сикритов Михаил Сергеевич	3	троечник
10	Соустин Алексей Александрович	5	отличник
11	Сухов Даниил Алексеевич	4	ударник
12	Тырцаков Дмитрий Александрович	4	ударник
13	Хальметов Камилъ Наилевич	5	отличник
14	Худин Артем Александрович	3	троечник
15	Челушкин Александр Александрович	3	троечник
16	Чирьев Даниил Владимирович	3	троечник
17	Шашкин Егор Алексеевич	3	троечник

Рисунок 17 – Данные прогнозирования, выведенные консольно

	A	B	C	D
1		ФИО	прогноз	класс прогноза
2	0	Агашин Павел Александрович	3	троечник
3	1	Алёшин Анатолий Сергеевич	4	ударник
4	2	Амирян Олег Евгеньевич	4	ударник
5	3	Вдовин Никита Александрович	4	ударник
6	4	Карманов Родион Николаевич	5	отличник
7	5	Кузнецов Дмитрий Александрович	3	троечник
8	6	Кузьмин Владислав Павлович	3	троечник
9	7	Нагаев Ярослав Александрович	3	троечник
10	8	Сидоров Кирилл Андреевич	5	отличник
11	9	Сикритов Михаил Сергеевич	3	троечник
12	10	Соустин Алексей Александрович	5	отличник
13	11	Сухов Даниил Алексеевич	4	ударник
14	12	Тырцаков Дмитрий Александрович	4	ударник
15	13	Хальметов Камиль Наилевич	5	отличник
16	14	Худин Артем Александрович	3	троечник
17	15	Челушкин Александр Александрович	3	троечник
18	16	Чирьев Даниил Владимирович	3	троечник
19	17	Шашкин Егор Алексеевич	3	троечник

Рисунок 18 – Данные прогнозирования, выведенные в файл

По спрогнозированным данным были построены диаграммы, характеризующие общую успеваемость группы и индивидуальную успеваемостью. Диаграммы представлены на рисунках 19, 20 и 21 соответственно. На рисунке 19 расположена диаграмма отражающая количество студентов в каждой категории:



Рисунок 19 – Количественные данные прогнозирования

На рисунке 20 расположена круговая диаграмма, отражающая процентное соотношение студентов в каждой категории:

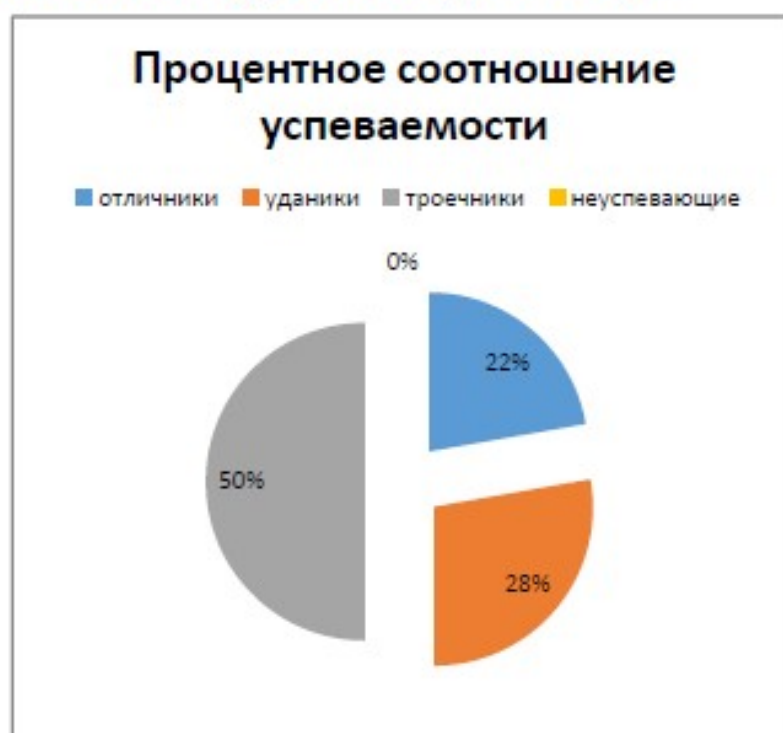


Рисунок 20 – Данные прогнозирования в процентном соотношении

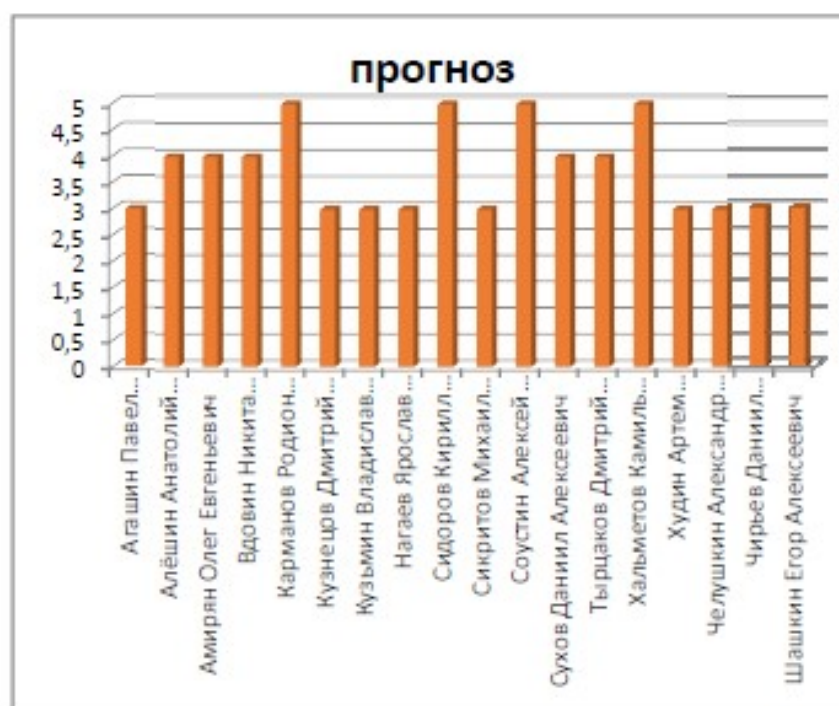


Рисунок 21 – Данные прогнозирования индивидуальной успеваемости

Заключение

В результате магистерской работы было проведено исследование существующих на данный момент нейронных сетей, а также их реализаций. Была выбрана модель, наиболее подходящая для прогнозирования по наборам данных. Перед началом проектирования был произведен анализ данных и выявлены требования к используемым данным для обучения.

По окончании работ было принято решение о реализации рекуррентной нейронной сети на языке программирования Python. Средой разработки была выбрана IDE Spyder.

На основании анализа удалось разработать рекуррентную нейронную сеть, реализующую прогнозирование на основе данных, собранных из электронной информационно-образовательной среды ВУЗа (ЭИОС). Анализ модели показал её работоспособность и высокий процент точности.

В перспективе работа может быть полезна кураторам и сотрудникам университета, занимающимся работой со студенческой успеваемостью. Заранее определить на кого из учащихся стоит обратить особое внимание.

Были спрогнозированы данные для учащихся курируемых групп. Результаты представлены на рисунках 22 и 23. Эти предсказанные результаты будут проверяться в ходе текущей весенней сессии 2024 года.

Предлагается за нулевой показатель взять средний уровень успеваемости, то есть оценку 3 («удовлетворительно»). Следовательно, все остальные показатели будут распределяться относительно этого уровня. Все оценки ниже нулевого уровня окажутся в отрицательной области, то есть в зоне риска (от 0 до 3). Как результат – выделяется часть неуспевающих студентов, на которых следует обратить внимание и более тщательно с ними работать в плане мотивации, либо контроля успеваемости.

23	Видяев Алексей Антонович	4 ударник
24	Власова Андриана	4 ударник
25	Галкин Александр Павлович	4 ударник
26	Гирак Дмитрий Андреевич	3 троечник
27	Головушкина Александра Борисовна	5 отличник
28	Евстифеев Данила Александрович	4 ударник
29	Жиганов Никита Денисович	5 отличник
30	Иванов Егор Андреевич	2 неуспевающий
31	Казаков Александр Дмитриевич	4 ударник
32	Кузьмин Антон Алексеевич	4 ударник
33	Куликов Илья Александрович	5 отличник
34	Куликовский Вадим Алексеевич	4 ударник
35	Липатов Денис Сергеевич	4 ударник
36	Макаров Алексей Сергеевич	4 ударник
37	Марушкин Данила Игоревич	2 неуспевающий
38	Молев Павел Олегович	4 ударник
39	Никитин Владислав Александрович	4 ударник
40	Николаев Александр Владимирович	5 отличник
41	Панин Артём Сергеевич	4 ударник
42	Панькин Роман Александрович	4 ударник
43	Прохоров Даниил Сергеевич	5 отличник
44	Пыльников Никита Николаевич	4 ударник
45	Соколов Вадим Дмитриевич	3 троечник
46	Усов Александр Сергеевич	5 отличник
47	Швырев Артём Ильич	3 троечник
48	Юров Даниил Максимович	5 отличник

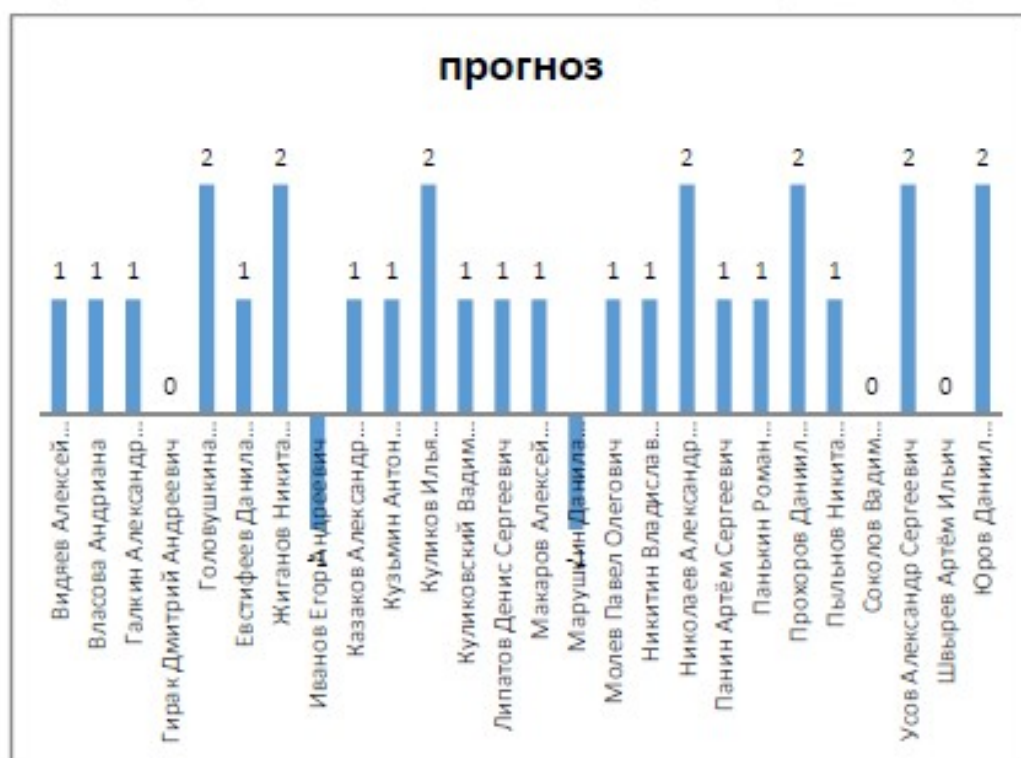


Рисунок 22 – Прогноз оценок группы 23BBB1

	ФИО	прогноз	класс прогноза
0	Бурков Антон Игоревич	4	ударник
1	Герасимов Владислав Романович	4	ударник
2	Кандаков Андрей Александрович	4	ударник
3	Кокарев Данила Сергеевич	4	ударник
4	Конкин Дмитрий Станиславович	4	ударник
5	Кочегин Валерий Вильданович	5	отличник
6	Кузин Максим Евгеньевич	5	отличник
7	Лисов Егор Алексеевич	4	ударник
8	Мадамкин Виталий Михайлович	3	троечник
9	Монин Иван Алексеевич	5	отличник
10	Пичаев Иван Андреевич	5	отличник
11	Попков Сергей Андреевич	4	ударник
12	Пырков Денис Алексеевич	4	ударник
13	Родионов Алексей Александрович	5	отличник
14	Севостьянов Олег Денисович	4	ударник
15	Синичкин Даниил Анатольевич	3	троечник
16	Скалдин Вадим Сергеевич	4	ударник
17	Смушко Леонид Игоревич	3	троечник
18	Стрельцов Александр Павлович	4	ударник
19	Федонин Павел Александрович	5	отличник
20	Федоров Борис Максимович	4	ударник
21	Чебанов Андрей Алексеевич	3	троечник
22	Чупраков Сергей Владимирович	5	отличник

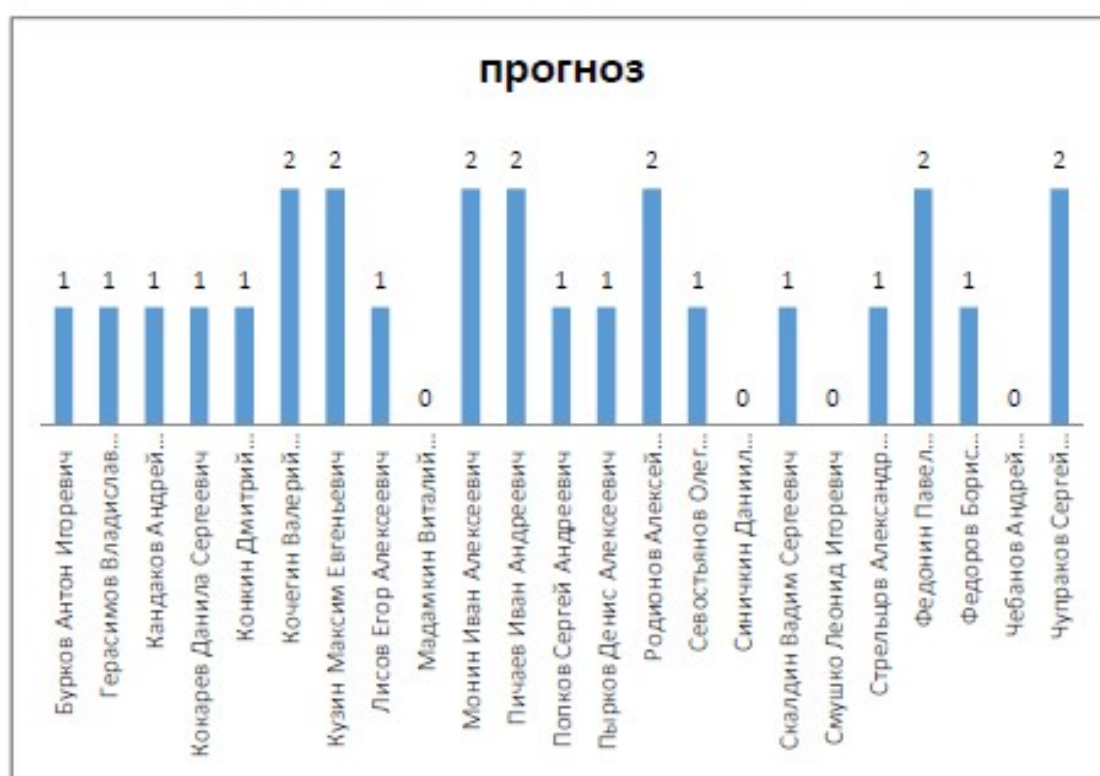


Рисунок 23 – Прогноз оценок группы 23BBB2

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бирюков Е.В., Корнев М.С. Практическая реализация нечеткой нейронной сети при краткосрочном прогнозировании электрической нагрузки // Портал магистров ДонНТУ [Электронный ресурс]: сайт. - URL: http://www.masters.donntu.edu.ua/2006/kita/chuykov/library/library/article_5.htm (дата обращения 02.09.2023).
2. Дюк В.А. Data Mining – интеллектуальный анализ данных // Информационные технологии [Электронный ресурс]: сайт. – URL: <http://www.inftech.webservis.ru/it/database/datamining/ar2.html> (дата обращения 01.09.2023)
3. Xianjun Ni Research of Data Mining Based on Neural Networks // World Academy of Science, Engineering and Technology. – 2008. – № 39. – P. 381-384.
4. Абхишек Шарма. Машинное обучение – Приложения // Информационные технологии [Электронный ресурс]: сайт. - URL: <https://progler.ru/blog/mashinnoe-obuchenie-prilozheniya> (дата обращения 01.11.2023)
5. Хенрик Б., Джозеф Р., Марк Ф. Машинное обучение. – Издательский дом «Питер», 2017. – 336 с.
6. Хайкин С. Нейронные сети: полный курс, 2-е издание. – М.: Вильямс.2008.
7. Бендерская Е. Н., Никитин К. В. Рекуррентная нейронная сеть как динамическая система и подходы к ее обучению // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика. Телекоммуникации. Управление. 2013. №4. С. 29.
8. Бендерская Е. Н., Граничин О. Н. Использование сложности входного образа в управлении структурной сложностью интеллектуальной

системы // 5-я Российская мультikonференция по проблемам управления. Санкт-Петербург. 2012. С. 276–280.

9. Бендерская Е. Н., Толстов А. А. Реализация осцилляторной хаотической нейронной сети с применением технологии NVIDIA CUDA для решения задач кластеризации // Информационно-управляющие системы. 2014. №4. С. 94–101.

10. Schmidhuber J. Deep Learning in Neural Networks: An Overview // Neural Networks. 2015. Vol. 61. P. 85–117.

11. Mikolov T., Karafiat M., Burget L., Cernocky J., Khudanpur S. Recurrent neural network based language model // 11th Annual Conference of the International Speech Communication Association. Japan. 2010. P. 1045–1048.

12. Прошкина, Е. Н. Анализ и прогнозирование успеваемости студентов на основе радиальной базисной нейронной сети / Е. Н. Прошкина, И. Ю. Балашова. – Текст: непосредственный // Технические науки: традиции и инновации: материалы III Междунар. науч. конф. (г. Казань, март 2018 г.). – Казань: Молодой ученый, 2018. – С. 24-28. – [Электронный ресурс]: URL: <https://moluch.ru/conf/tech/archive/287/13683/> (дата обращения: 27.05.2024)

13. Русаков С.В., Русакова О.Л., Посохина К.А. Нейросетевая модель прогнозирования группы риска по успеваемости студентов первого курса // Современные информационные технологии и ИТ-образование. 2018. № 4. С. 815–822.

14. ML: Рекуррентные сети на PyTorch [Электронный ресурс]: сайт. – URL: https://qudata.com/ml/ru/NN_RNN_Torch.html (дата обращения 09.03.2024)

15. Diederik P. Kingma, Jimmy Lei Ba. ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION // arXiv:1412.6980v9 [cs.LG] 30 Jan 2017

16. Андрей Алексеев. Пишем простую нейронную сеть с использованием математики и NumPy // Хабр [Электронный ресурс]: сайт. – URL: <https://habr.com/ru/articles/460589/> (дата обращения: 27.05.2024)

17. Ерофеева В. А. Обзор теории интеллектуального анализа данных на базе нейронных сетей // Стохастическая оптимизация в информатике. 2015. Т. 11. № 3. С. 3-17.

ПРИЛОЖЕНИЕ А

(Обязательное)

Прогнозирование успеваемости студентов с использованием
нейронных сетей
Листинг программы

					ПГУ1.090401.49.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		66

Приложение А.1

Файл main_Y.py

```
# -*- coding: utf-8 -*-
"""
@author: Myata
"""

import pandas as pd
import numpy as np

# Загрузка данных из файла Excel
data = pd.read_excel('newBook.xlsx')

def normalised_data(data):
    norm_data = data.copy()
    norm_data.replace('отлично', 5 ,inplace=True)
    norm_data.replace('хорошо', 4 ,inplace=True)
    norm_data.replace('удовл', 3 ,inplace=True)
    norm_data.replace(['недопущен','н/я', 'незачет'], 0
,inplace=True)
    norm_data.replace('зачет', np.nan ,inplace=True)

#Вычисление среднего значения по строке
averages = []
table1 = norm_data.iloc[:,2:-4].values
for row in table1:
    numbers = [float(num) for num in row if num != 'nan']
    avg = np.nanmean(numbers) if len(numbers) > 0 else
np.nan
averages.append(avg)
```

```

#Переформатирование массива в столбец для подстановки в
датасет
vec = pd.Series(np.array(averages).T)

#Вычислениеколичествапустыхячеек
df_numeric = norm_data.select_dtypes(include=[np.number])
numeric_cols = df_numeric.columns.values

# выполняет вставку для столбцов, только в которых отсутствуют
значения.
for col in numeric_cols:
    if norm_data[col].isnull().sum() > 0:
        norm_data[col].fillna(vec, inplace=True)

return norm_data

normal_data = normalased_data(data)

normal_data.to_excel('new_data.xlsx')

```

Приложение А.2

Файл RNN_Y.py

```
# -*- coding: utf-8 -*-
"""
@author: Myata
"""

import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

# Определение модели RNN
class RNNModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size,
                 num_layers=1):
        super(RNNModel, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers,
                             batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0),
                         self.hidden_size).to(x.device)
        c0 = torch.zeros(self.num_layers, x.size(0),
                         self.hidden_size).to(x.device)
```

```

out, _ = self.lstm(x.unsqueeze(1), (h0, c0))
out = self.fc(out[:, -1, :])
return out

```

					ПГУ1.090401.49.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.3
Файл RNN_main_train.py

```
# -*- coding: utf-8 -*-
"""
@author: Myata
"""

import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from RNN_Y import RNNModel

# Загрузка данных из файла Excel
data = pd.read_excel('newBook_341.xlsx')

# Предобработка данных
# Извлечение признаков и меток
features = data.drop(columns=['ФИО', 'класс', 'величина класса'])
labels = data.iloc[:, -1:]

# Преобразование категориальных данных в числовые
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(labels)

# Нормализация данных
```

```

scaler = StandardScaler()
features = scaler.fit_transform(features)

# Разделение данных на тренировочный и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(features,
labels, test_size=0.2, random_state=42)

class StudentPerformanceDataset(Dataset):
    def __init__(self, data, labels):
        self.data = torch.tensor(data, dtype=torch.float32)
        self.labels = torch.tensor(labels, dtype=torch.long)

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        return self.data[idx], self.labels[idx]

# Создания датасетов и загрузчиков данных
train_dataset = StudentPerformanceDataset(X_train, y_train)
test_dataset = StudentPerformanceDataset(X_test, y_test)

train_loader = DataLoader(train_dataset, batch_size=32,
shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32,
shuffle=False)

# Параметры модели
input_size = X_train.shape[1]
hidden_size = 128
output_size = len(label_encoder.classes_)
num_layers = 2

```

```

device = torch.device('cuda' if torch.cuda.is_available() else
'cpu')

model = RNNModel(input_size, hidden_size, output_size,
num_layers).to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# функция для обучения модели
def train_model(model, train_loader, criterion, optimizer,
num_epochs):
    model.train()
    for epoch in range(num_epochs):
        running_loss = 0.0
        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device),
labels.to(device)

            outputs = model(inputs)
            loss = criterion(outputs, labels)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
        if (epoch + 1) % 5 == 0:
            print(f'Epoch {epoch+1}/{num_epochs}, Loss:
{running_loss/len(train_loader):.4f}')

# Обучение модели

```

```

train_model(model, train_loader, criterion, optimizer,
num_epochs=30)

# функция для оценки модели
def evaluate_model(model, test_loader, label_encoder):
    model.eval()
    correct = 0
    total = 0
    all_preds = []
    all_labels = []
    with torch.no_grad():
        for inputs, labels in test_loader:
            inputs, labels = inputs.to(device),
labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
            all_preds.extend(predicted.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    accuracy = 100 * correct / total
    print(f'Accuracy: {accuracy:.2f}%')

    all_preds = label_encoder.inverse_transform(all_preds)
    all_labels = label_encoder.inverse_transform(all_labels)

    return all_preds, all_labels, accuracy

# функция для предсказания классов на новых данных
def predict(model, data, label_encoder):

```

```

model.eval()

data = torch.tensor(data, dtype=torch.float32).to(device)

with torch.no_grad():
    outputs = model(data)
    _, predicted = torch.max(outputs.data, 1)

predicted =
label_encoder.inverse_transform(predicted.cpu().numpy())
return predicted

# Оценка модели и вывод прогнозов и фактических значений
preds, labels, accuracy = evaluate_model(model, test_loader,
label_encoder)

# Вывод результатов прогноза
results = pd.DataFrame({'Predicted': preds, 'Actual': labels})
print(results)

# Сохранение параметров модели
torch.save(model.state_dict(), 'data_0609.pth')

new_data = X_test[50:55] # Используем первые 5 примеров из
тестового набора для демонстрации
new_preds = predict(model, new_data, label_encoder)
print("New predictions:", new_preds)

model.load_state_dict(torch.load('data_0609.pth'))

df = pd.read_excel('new_data.xlsx', index_col=0)

# Предобработка данных
# Извлечение признаков и меток
features1 = df.drop(columns=['ФИО'])

```

```

# Нормализация данных
scaler = StandardScaler()
features1 = scaler.fit_transform(features1)

new_preds = predict(model, features1, label_encoder)
print("New predictions:", new_preds)

preds_data = df.iloc[:, :1]
preds_data['прогноз'] = new_preds
preds_data['класс прогноза'] = new_preds
preds_data['класс прогноза'].replace(5, 'отличник', inplace=True)
preds_data['класс прогноза'].replace(4, 'ударник', inplace=True)
preds_data['класс прогноза'].replace(3, 'троечник', inplace=True)
preds_data['класс прогноза'].replace(2, 'неуспевающий',
inplace=True)
print(preds_data)

preds_data.to_excel('preds_data.xlsx')

```

Приложение Б
(Обязательное)

Прогнозирование успеваемости студентов с использованием
нейронных сетей
Презентация

					ПГУ1.090401.49.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		77

ПРОГНОЗИРОВАНИЕ УСПЕВАЕМОСТИ СТУДЕНТОВ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Автор работы – магистрант гр. 22ВВИм2 Юрова О. В.
Научный руководитель – к.т.н., доцент Карамышева Н. С.

Пенза, 2024

Цель и задачи исследования

Цель: проектирование, разработка и реализация модели прогнозирования класса обучения студентов с использованием электронной информационно-образовательной среды (ЭИОС) ВУЗа на основе технологии нейронных сетей.

Задачи:

- 1) анализ и формирование набора данных для прогнозирования,
- 2) изучение методов и способов классификации с использованием нейросети,
- 3) реализация модели нейросети для прогнозирования класса обучающегося,
- 4) обучение реализованной нейросети,
- 5) апробация реализованного функционала – получение прогноза.

Выбор инструментов для решения поставленных задач



Системы, использующие прогнозирование успеваемости обучающихся



Управляй своей успеваемостью!

Статистика Успеваемости — приложение, которое помогает учителям и родителям достигать лучших результатов в учебе.

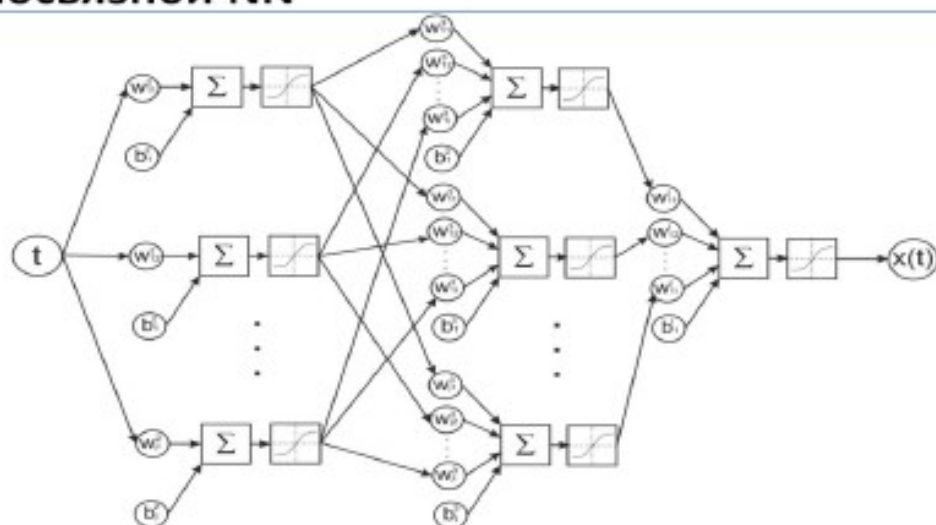


Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090401.49.001 ПЗ

Лист
79

Прогнозирование с использованием полносвязной NN



ReLU (Rectified Linear Activation)

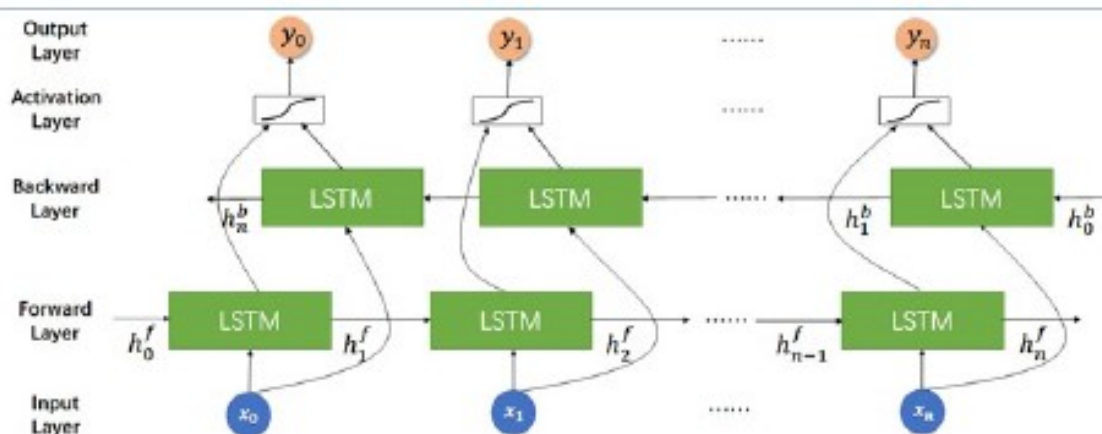
$$f(x) = \max(0, x) = \begin{cases} x_i & \text{if } x_i > 0 \\ 0 & \text{if } x_i < 0 \end{cases}$$

Softmax

$$\text{Softmax}_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

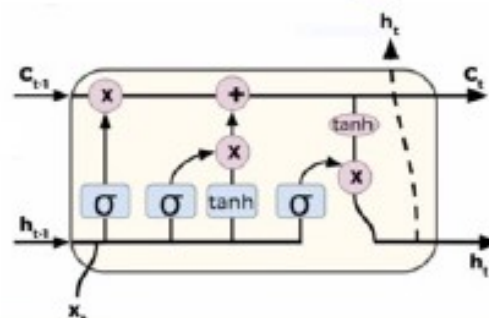
...

Прогнозирование с использованием RNN



LSTM-ячейка:

$$\begin{cases} h^{(t)} = \tanh c^{(t)} \odot f(x^{(t)}, h^{(t-1)}), \\ c^{(t)} = g(x^{(t)}, h^{(t-1)}, c^{(t-1)}) \end{cases}$$



Прогнозирование с использованием RNN

LSTM-ячейка:

функция потерь CrossEntropyLoss:

$$\begin{cases} \mathbf{h}^{(t)} = \tanh \mathbf{c}^{(t)} \odot f(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}), \\ \mathbf{c}^{(t)} = g(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}, \mathbf{c}^{(t-1)}) \end{cases} \quad L(y, c) = -w_c \log \left(\frac{\exp y_{ic}}{\sum_{\alpha} \exp y_{i\alpha}} \right)$$

Adam – метод стохастической оптимизации

$$v_t = (1 - \beta_2) \sum_{i=1}^t \beta_2^{t-i} \cdot g_i^2$$

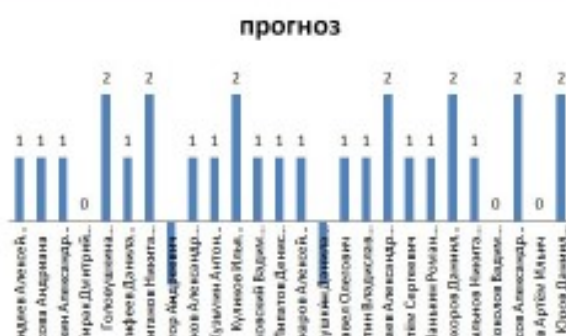
Accuracy: 87.93%

	Predicted	Actual
0	3	3
1	3	3
2	5	5
3	3	3
4	5	5
...	...	...
188	4	4
189	3	3
190	5	5
191	3	3
192	4	4

[193 rows x 2 columns]

Результаты прогнозирования

23	Виднев Алексей Антонович	4 ударник
24	Власова Андриана	4 ударник
25	Галкин Александр Павлович	4 ударник
26	Гирак Дмитрий Андреевич	3 троечник
27	Головущкина Александра Борисовна	5 отличник
28	Евстифеев Данила Александрович	4 ударник
29	Жиганов Никита Денисович	5 отличник
30	Иванов Егор Андреевич	2 неуспевающий
31	Казakov Александр Дмитриевич	4 ударник
32	Кузьмин Антон Алексеевич	4 ударник
33	Куликов Илья Александрович	5 отличник
34	Куликовский Вадим Алексеевич	4 ударник
35	Липатов Денис Сергеевич	4 ударник
36	Макаров Алексей Сергеевич	4 ударник
37	Марушкин Данила Игоревич	2 неуспевающий
38	Молев Павел Олегович	4 ударник
39	Никитин Владислав Александрович	4 ударник
40	Николаев Александр Владимирович	5 отличник
41	Панин Артём Сергеевич	4 ударник
42	Панькин Роман Александрович	4 ударник
43	Прохоров Даниил Сергеевич	5 отличник
44	Пыльников Никита Николаевич	4 ударник
45	Соколов Вадим Дмитриевич	3 троечник
46	Усов Александр Сергеевич	5 отличник
47	Шкырев Артём Ильич	3 троечник
48	Юров Даниил Максимович	5 отличник



Процентное соотношение успеваемости

