

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»

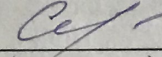
Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

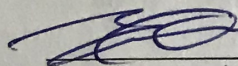
на тему

ИНТЕЛЛЕКТУАЛЬНАЯ ОБРАБОТКА
ДАННЫХ ФИЗИЧЕСКИХ
ЭКСПЕРИМЕНТОВ

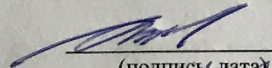
Студент

 Смирнова Алина Дмитриевна
(подпись, дата) 24.06.2024 (ФИО полностью)

Руководитель

 Зинкин С.А.
(подпись, дата) 24.06.2024 (фамилия, инициалы)

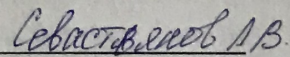
Нормоконтролёр

 Бычков А.С.
(подпись, дата) 24.06.2024 (фамилия, инициалы)

Рецензент

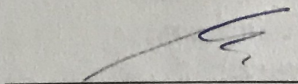
(должность, место работы)

(подпись, дата)

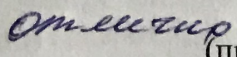
 Севаст'янов Д.В.
(фамилия, инициалы)

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

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой  (протокол заседания ГЭК от

25.06.2024г. № 47)

Секретарь ГЭК


(подпись)

Е.И. Гурин

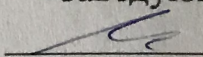
(фамилия, инициалы)

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 » 09 20 22 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Смирнова Алина Дмитриевна гр. 22ВВИм1 факультета вычислительной техники, направления 09.04.01.

2. Тема работы «Интеллектуальная обработка данных физических экспериментов»

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

3. Руководитель магистерской диссертации д.т.н., профессор Зинкин С.А.

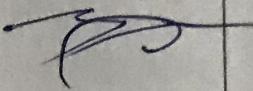
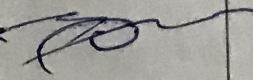
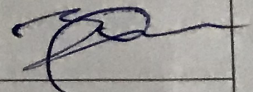
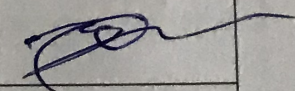
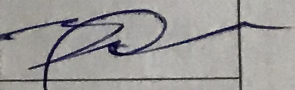
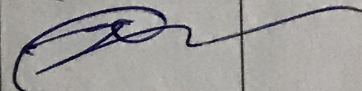
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки – исследование интеллектуальной обработки данных физических экспериментов, создание нейронной сети для анализа данных физического эксперимента. Показать применимость разработанных методов и средств на конкретных примерах.

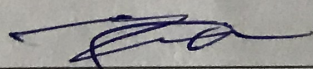
5. Перечень подлежащих разработке вопросов

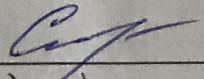
Требуется произвести анализ существующих методов обработки и анализа данных физических экспериментов, обзор существующих аналоговых разработок, выбрать технологии, применяемые при создании и обучении нейросетевых моделей, разработать нейронную сеть для обработки данных физических экспериментов, обучить модель на доступных данных, протестировать и сравнить полученную модель с существующими.

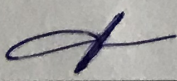
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ и исследование предметной области	15%	13.09.2022 – 31.12.2022	
2	Анализ и сравнение классических и нейросетевых алгоритмов для интеллектуальной обработки данных	10%	01.01.2023 – 28.02.2023	
3	Описательно-статистический анализ экспериментальных данных	20%	01.03.2023 – 31.05.2023	
4	Разработка нейронной сети	30%	01.06.2023 – 31.12.2023	
5	Тестирование и отладка разработанной нейронной сети	15%	01.01.2024 – 04.04.2024	
6	Оформление пояснительной записки	10%	05.04.2024 – 31.05.2024	

Дата выдачи задания "13" сентября 2022г.

Руководитель магистерской диссертации  Зинкин С.А.
(подпись, дата)

Задание к исполнению принял студент  Смирнова А.Д.
(подпись, дата)

Работу к защите допустить
Декан факультета  01.07.2024 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Магистерская диссертация содержит 73 листов, 42 рисунка, 1 таблица, 26 источников

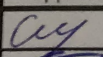
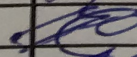
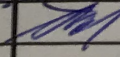
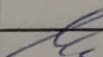
ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, НЕЙРОННЫЕ СЕТИ, АЛГОРИТМЫ, ФИЗИЧЕСКИЕ ЭКСПЕРИМЕНТЫ, МНОГОСЛОЙНЫЙ ПЕРСЕПТРОН, PYTHON

Цель работы сравнительный анализ и изучение нейросетевых алгоритмов для интеллектуальной обработки данных физических экспериментов, исследование и выбор архитектуры в качестве основы для нейронной сети.

В ходе выполнения работы были изучены виды интеллектуальной обработки, сравнены и выделены преимущества и недостатки существующего программного обеспечения для анализа и обработки экспериментальных данных, произведен сравнительный анализ и выбор нейронной сети. Разработана, обучена и протестирована нейронная сеть, способная прогнозировать результаты физических экспериментов.

Результатом работы является разработанный многослойный персептрон, удовлетворяющий поставленной задаче.

Разработка производилась с использованием широкого стека технологий, к которым относятся: язык программирования Python, сервис Google Colaboratory для написания кода на языке Python в Jupiter notebook и фреймворк машинного обучения PyTorch для разработки нейросетевой модели.

					ПГУ1.090401.23.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Интеллектуальная обработка данных физических экспериментов Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Смирнова А.Д.		24.06.24					
Провер.		Зинкин С.А.		24.06.24			2	73	
Н. Контр		Бычков А.С.		24.06.24		ФВТ, каф. ВТ гр. 22ВВИм1			
Утверд.		Митрохин М.А.		24.06.24					

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	4
ВВЕДЕНИЕ	5
1 Анализ предметной области.....	7
1.1 Физические эксперименты	7
1.2 Методы обработки данных.....	8
1.3 Обзор аналогичных программ.....	15
2 Интеллектуальный анализ данных.....	23
2.1 Предварительная обработка данных	28
2.2 Модель в виде нейронной сети.....	39
2.2.1 Выбор нейронной сети.....	39
2.2.2. Разработка нейронной сети.....	49
ЗАКЛЮЧЕНИЕ	57
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	59
Приложение А – Листинг программы	62
Приложение Б - Презентация	68

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей магистерской диссертации применяются следующие сокращения и обозначения

ГНС – гамильтоновы нейронные сети

ЛНС – лагранжевы нейронные сети

ИИ – искусственный интеллект

НС – нейронные сети

ОДА – однофакторный дисперсионный анализ

ПО – программное обеспечение

MLP (Multilayered perceptron) – многослойный персептрон

ВВЕДЕНИЕ

Экспериментальные исследования в физике включают в себя не только измерительную часть, но и обработку, анализ и интерпретацию экспериментальных данных, необходимые при оценке погрешности и статистическом анализе.

В точных науках основным способом получения экспериментальных данных является измерение различных физических величин. Для измерения физических величин используются не сами эталоны физических, а измерительными приборами, которые сверены с эталонами. Однако сравнение измерительных приборов и инструментов с эталонами сопровождается с неточностями калибровки, что приводит к появлению погрешности.

Развитие компьютерных технологий привело к тому, что практически ни один физических эксперимент не обходится без применения компьютеров и специализированных программных комплексов, позволяющих управлять экспериментальным установками, собирать и обрабатывать полученные данные, представлять результаты эксперимента в удобной и наглядной форме. Однако не все задачи могут быть решены с помощью реализованных специализированных библиотек. Например, эксперименты, проводимые на андронном коллайдре содержат большое количество данных, содержат большое количество данных. Обработка этих экспериментальных данных занимает много времени и ресурсов. Многие из существующих специализированных библиотек не способны произвести обработку и анализ данных такого объема. Поэтому ведется поиск новых методов для решения поставленных задач.

Данные недостатки показывают, что выбранная тема исследования является актуальной.

Целью диссертационной работы является исследование нейросетевых алгоритмов и методов анализа данных физических экспериментов, способных прогнозировать результаты экспериментов.

Основные задачи для достижения поставленной цели:

- обзор и анализ существующих исследований и алгоритмов, применяемых и направленных на интеллектуальный анализ данных физических экспериментов,
- разработка нейросетевой модели для интеллектуальный анализ данных физических экспериментов,
- обучение модели для увеличения разрешения изображения,
- тестирование и сравнение полученной модели с аналогами.

Методами исследования для решения задач в диссертационной работе являются методы интеллектуального анализа данных, методы машинного обучения, классические и нейросетевые методы обработки данных физических экспериментов.

1 Анализ предметной области

1.1 Физические эксперименты

Эксперимент является важнейшим элементом в изучении физических явлений, закономерностей и прочего.

Эксперимент – основной общенаучный эмпирический метод исследования, научно поставленный опыт с точно учитываемыми условиями. Эксперимент обобщает ряд сопряженных понятий: опыт, целенаправленное наблюдение, воспроизведение объекта познания, организация особых условий осуществления, проверка предсказаний. Основная цель эксперимента: выявление свойств исследуемых объектов, проверка справедливости гипотез.

Физические эксперименты различают по способу формирования условий (естественный, искусственный), по целям исследования (преобразующий, констатирующий, контролирующий, поисковый, решающий), по месту проведения (лабораторный, натурный, полевой, производственный), по структуре (простой, сложный), по характеру внешних воздействий (вещественный, энергетический, информационный), по типу моделей (материальный, мысленный), по числу варьируемых факторов (одно- и многофакторный).

Лабораторные эксперименты проводят с применением типовых приборов, специальных установок, стендов, оборудования и т. д. Эти исследования позволяют изучить влияние одних характеристик (параметров) при изменении других. Лабораторные исследования в случае достаточно полного научного обоснования эксперимента позволяют получить информацию с минимальными затратами. Однако лабораторные исследования не всегда позволяют моделировать реальный ход изучаемого процесса в полном объеме, поэтому возникает потребность в проведении производственного эксперимента [1].

Производственные экспериментальные исследования изучают процесс (технология, устройство, систему и т. п.) в реальных условиях с учетом

воздействия различных случайных факторов производственной среды. Одной из разновидностей производственных экспериментов является собирание статистических материалов в организациях и на предприятиях, которые накапливают те или иные данные. Ценность этих материалов заключается в том, что они систематизированы за многие годы по единой методике. Такие данные хорошо поддаются обработке методами статистики и теории вероятностей [1].

Основной целью физического эксперимента является выявление свойств исследуемых объектов, проверка справедливости гипотез.

Можно выделить четыре основные задачи физического эксперимента:

- 1) экспериментальное доказательство существования физического явления,
- 2) изучение зависимостей связанных между собой физических величин,
- 3) определение физических постоянных,
- 4) обоснование возможностей практической реализации физических знаний.

Выполнение любого физического эксперимента состоит из 4 этапов [2]:

- 1) разработка методики эксперимента,
- 2) сбор данных, запись на носители, первичная обработка данных,
- 3) вторичная обработка данных, включающая математическую обработку полученных результатов и оценка погрешности,
- 4) анализ результатов.

1.2 Методы обработки данных

Существует три метода обработки и анализа данных в физических экспериментах [1]:

- 1) графическое представление. Является более наглядным и позволяет выявить характер функциональной зависимости изучаемых физических величин,

- 2) аппроксимация экспериментальных данных. Замена одних объектов другими, более близкими к исходным,
- 3) статистическая обработка экспериментальных данных.

Более подробно рассмотрим статистический анализ и обработку экспериментальных данных.

Методы статистического анализа делятся на одномерные и многомерные. Одномерный метод применяется, когда любой элемент выборки оценивается общим измерителем [3].

На рисунке 1 представлена классификация одномерного метода статистического анализа:

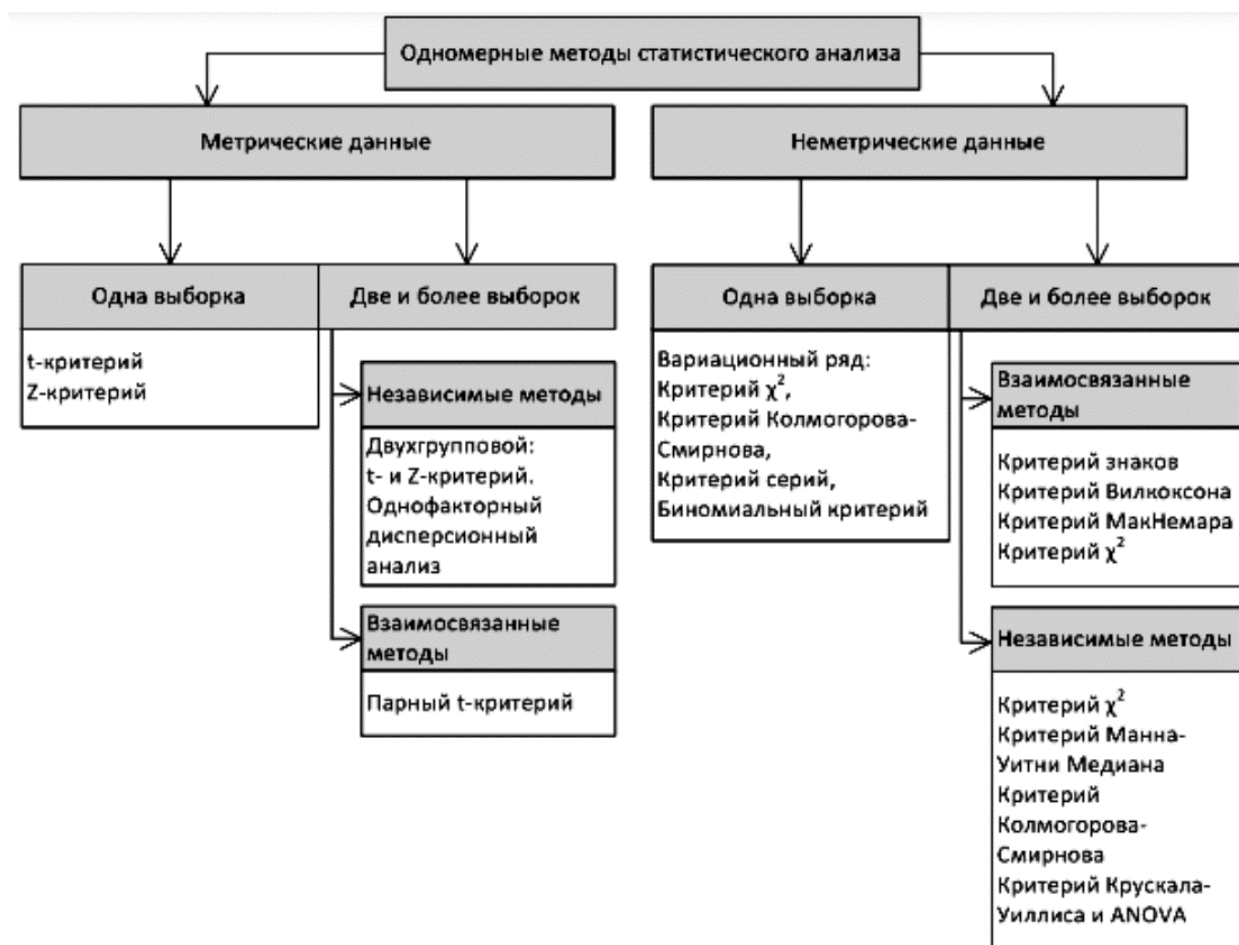


Рисунок 1 – Одномерные методы статистического анализа

Многомерные методы применяются, когда нельзя ограничиться аналитическим показателем. В этом случае требуется несколько измерителей в

разных шкалах, которые показывают различные стороны исследуемого предмета [3].

На рисунке 2 представлена классификация многомерного метода статистического анализа:



Рисунок 2 – многомерные методы статистического анализа

Дисперсионный анализ – метод, направленный на поиск зависимостей в экспериментальных данных путем исследования значимости различий в средних значениях.

Дисперсионный анализ позволяет анализировать влияние различных факторов на результаты эксперимента. В зависимости от количества факторов дисперсионный анализ может быть – однофакторным, двухфакторным и трехфакторным и т.д.

Анализ основан на расчете статистики Фишера, представляющей из себя отношение двух дисперсий: межгрупповой и внутригрупповой.

Задача дисперсионного анализа – вычленив из общей вариативности признаки вариативности другого рода:

- вариативность, обусловленную действием каждой из исследуемых независимых переменных,
- вариативность, обусловленную взаимодействием исследуемых независимых переменных,
- случайную вариативность, обусловленную всеми другими неизвестными переменными.

Пусть при n значениях фактора A получены истинные результаты $x_1, x_2, \dots, x_n$. Если A влияет на них, то рассматривается только степень зависимости и в качестве меры влияния берется выборочная дисперсия, иначе они все одинаковые:

$$m_x^* = \frac{1}{n} \sum_{i=1}^n x_i, \quad (1)$$

$$D_A = \frac{1}{n-1} \sum_{i=1}^n (x_i - m_x^*)^2, \quad (2)$$

где D_A – аналог выборочной дисперсии и характеризует влияние фактора A , а не случайность.

Если нет других факторов и дисперсия фактора A при отсутствии случайности равна D_A , а генеральная дисперсия X при неизменном A равна D_0 , то генеральная дисперсия X будет равна:

$$D_x = D_A + D_0. \quad (3)$$

Однофакторный дисперсионный анализ (ОДА) – метод статистического анализа для сравнения средних значений двух или более групп [2].

ОДА используется для оценки степени влияния различных факторов на результат эксперимента. Он позволяет:

- сравнить среднее значение двух или более групп,
- определить, существует ли статически значимая разница между средними значениями групп,
- определить, в какой группе средние значения отличаются сильнее всего [23].

Корреляционный анализ – совокупность методов обработки данных для обнаружения статистической взаимосвязи между случайными величинами или признаками.

Задачи, решаемые методами корреляционного анализа:

- устанавливается наличие зависимости между переменными,
- предсказывает поведение одного признака на основе изменения другого,
- позволяет производить выбор набора входных переменных для модели, коррелирующих между собой в наименьшей степени и в наибольшей степени с выходными данными.

Степень взаимосвязи между двумя изучаемыми переменными показывает коэффициент корреляции [24]:

$$r_{xy} = \frac{\overline{xy} - \bar{x}\bar{y}}{\sigma(x)\sigma(y)}, \quad (4)$$

где $\sigma(x), \sigma(y)$ – среднее квадратичное отклонение,

$\overline{xy} = \frac{1}{n} \sum_{i=1}^n x_i y_i$ – выборочное среднее произведений случайных величин,

$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ – выборочное среднее случайной величины X ,

$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ – выборочное среднее случайной величины Y .

Интерпретация коэффициента корреляции:

- если $r_{xy} = 1$, то между двумя значениями идеальная положительная корреляция (точки на диаграмме рассеяния будут лежать на прямой линии),
- если $r_{xy} = -1$, то между двумя значениями идеальная отрицательная корреляция,
- если $r_{xy} = 0$, то между двумя значениями отсутствует корреляция.

На рисунке 3 представлены графики корреляции:

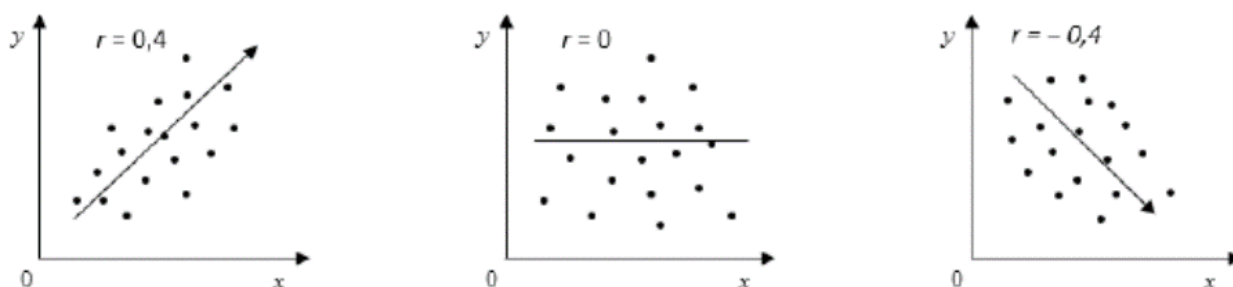


Рисунок 3 – Графическое изображение данных.

Можно выделить три вида коэффициента корреляции:

- 1) коэффициент Пирсона,
- 2) коэффициент Кендалла,
- 3) коэффициент Гудмена-Краскела.

Свойства коэффициента Пирсона:

- количество значений в исследуемых переменных X и Y должно быть одинаковым,

- исследуемые переменные должны быть измерены в интервальной шкале,
- исследуемые переменные должны быть распределены нормально.

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{X})^2 \sum_{i=1}^n (y_i - \bar{Y})^2}}, \quad (5)$$

где x_i – значение переменной X ,

y_i – значение переменной Y ,

$\bar{X}$ – среднее выборочное по X ,

$\bar{Y}$ – среднее выборочное по Y .

Сильная чувствительность к выбросам является недостатком коэффициента Пирсона [3].

Коэффициент Кендалла предназначен для взаимосвязи между двумя ранговыми переменными:

$$\tau_K = 1 - \frac{4K}{N * (N - 1)}, \quad (6)$$

где $K = P - Q$,

P – число совпадений,

Q – число инверсий,

N – число ранжируемых признаков.

Регрессионный анализ – набор статистических методов исследования влияния одной или нескольких независимых переменных на зависимую переменную.

Целями регрессионного анализа являются:

- определение степени детерминированности вариации зависимой переменной к независимым переменным,
- предсказание значения зависимой переменной с помощью независимой переменной.

Существует множество методов регрессионного анализа. Самые распространённые – линейная регрессия, логистическая регрессия и среднеквадратичная регрессия.

В линейной регрессии функция регрессии имеет вид:

$$f(x, b) = b_0 + b_1x_1 + b_2x_2 + \dots + b_kx_k, \quad (7)$$

где b_i – коэффициент регрессии, x_i – факторы модели, k – количество факторов модели [25].

Оценивание параметров линейной регрессии основано на методе наименьших квадратов.

Корреляционный и регрессионный анализ тесно связаны между собой. Корреляционно-регрессионный анализ – метод стохастического моделирования, используемый для исследования формы связи между случайными величинами изучаемого процесса [4].

1.3 Обзор аналогичных программ

Рассмотрим существующие методы и специализированные программы.

Система Matlab решает задачи матричного анализа, математической физики, оптимизационные задачи, выполняет обработку и визуализацию данных, обработку сигналов и изображений и многие другие задачи.

На рисунке 4 представлен интерфейс Matlab R2023a:

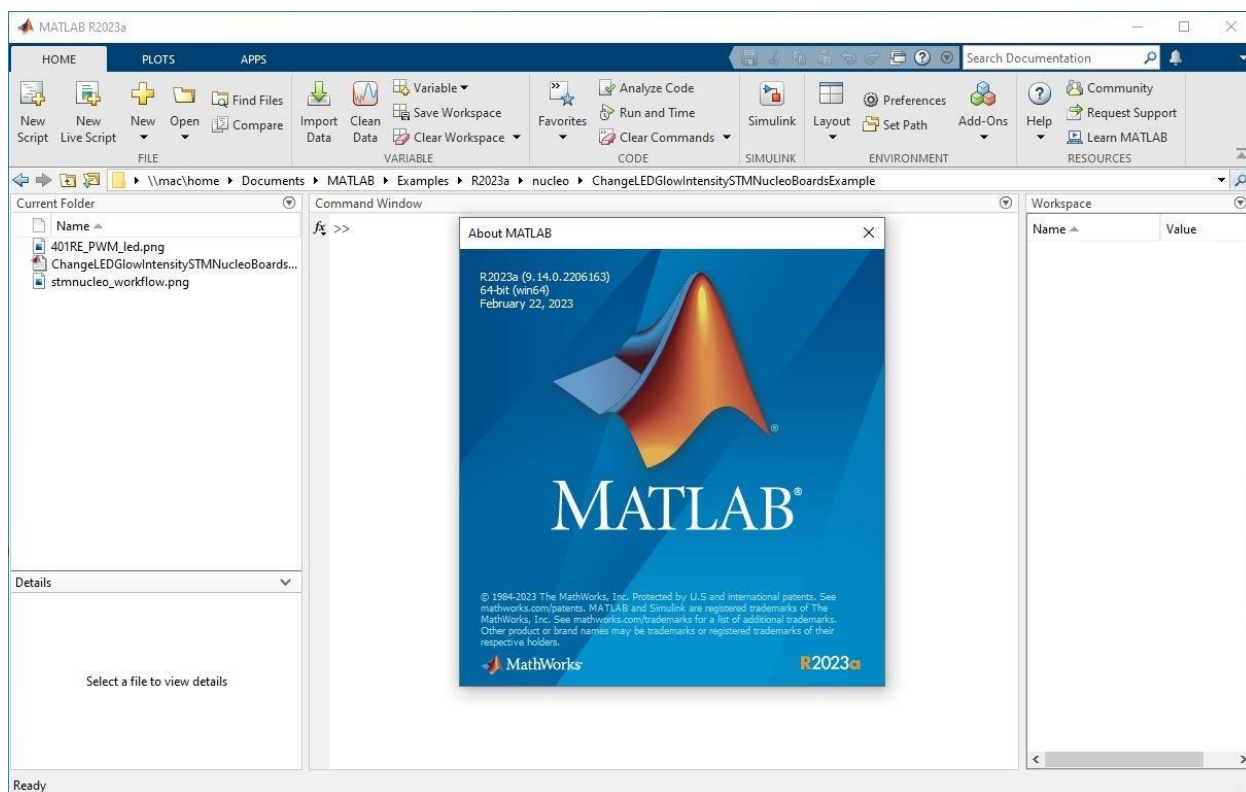


Рисунок 4 - Интерфейс Matlab R2023a

В Matlab имеются встроенный язык программирования и множество функций, позволяющие реализовать собственные алгоритмы. Для повышения производительности вычислений используется дополнительный модуль Matlab Compiler, обеспечивающий компиляцию программ, написанных на языке Matlab.

Система Matlab позволяет строить двумерные и трехмерные графики функций, заданных в виде векторов, матриц или аналитическом виде.

Также в системе Matlab можно моделировать искусственные нейронные сети. Примером является реализация искусственной нейронной сети в системе Matlab при решении ряда задач синтеза цепей и сетей связи в

телекоммуникационных систем [1, 2]. На рисунке 5 представлена структура нейронной сети в Matlab:

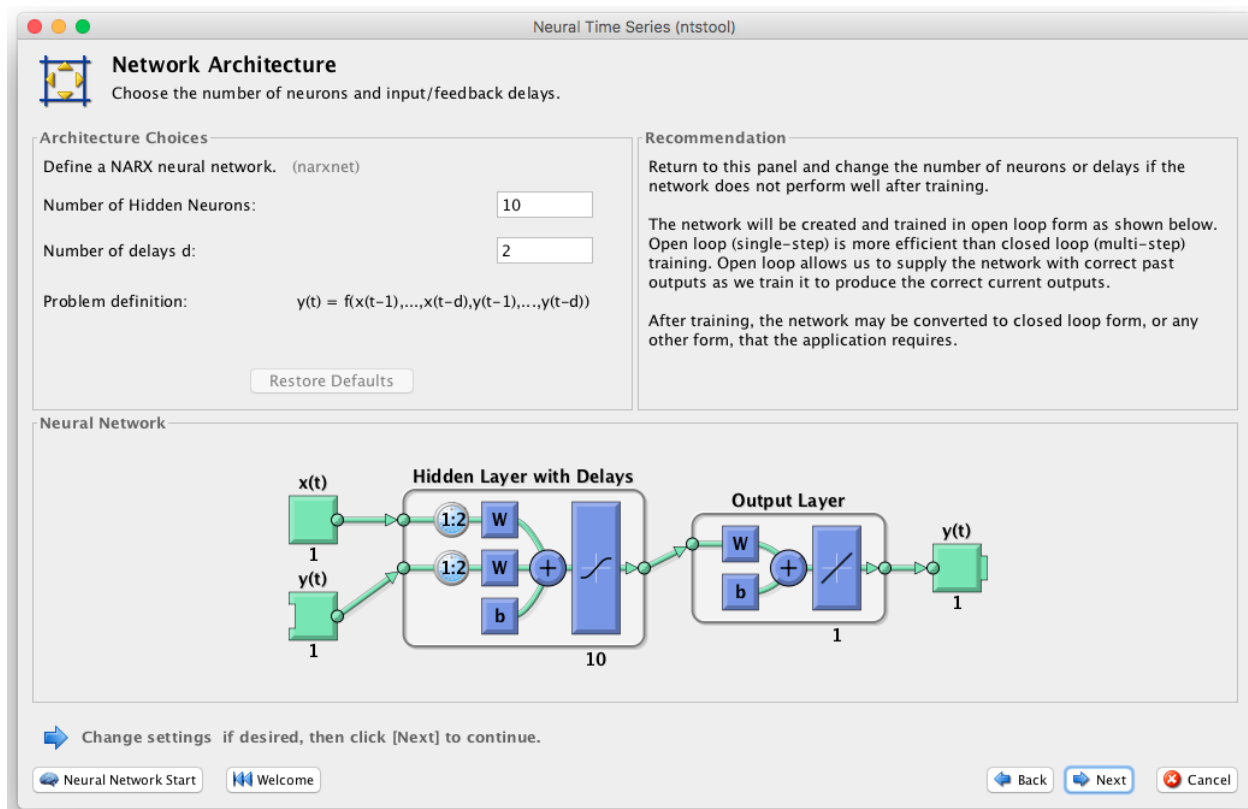


Рисунок 5 – Структура нейронной сети в Matlab

Недостатками являются дороговизна ПО, отсутствие возможности использования на других платформах.

Microsoft Excel является часто используемой программой для первичной обработки данных. Организация данных происходит в табличном формате. Как и в системе Matlab, MS Excel позволяет строить диаграммы, графики, использовать встроенные формулы для анализа и вычислений данных экспериментов [3]. Благодаря статистическому пакету анализа данных можно выполнять однофакторный дисперсионный анализ, корреляционный анализ и регрессионный анализ данных эксперимента [4].

На рисунке 6 представлен интерфейс MS excel:

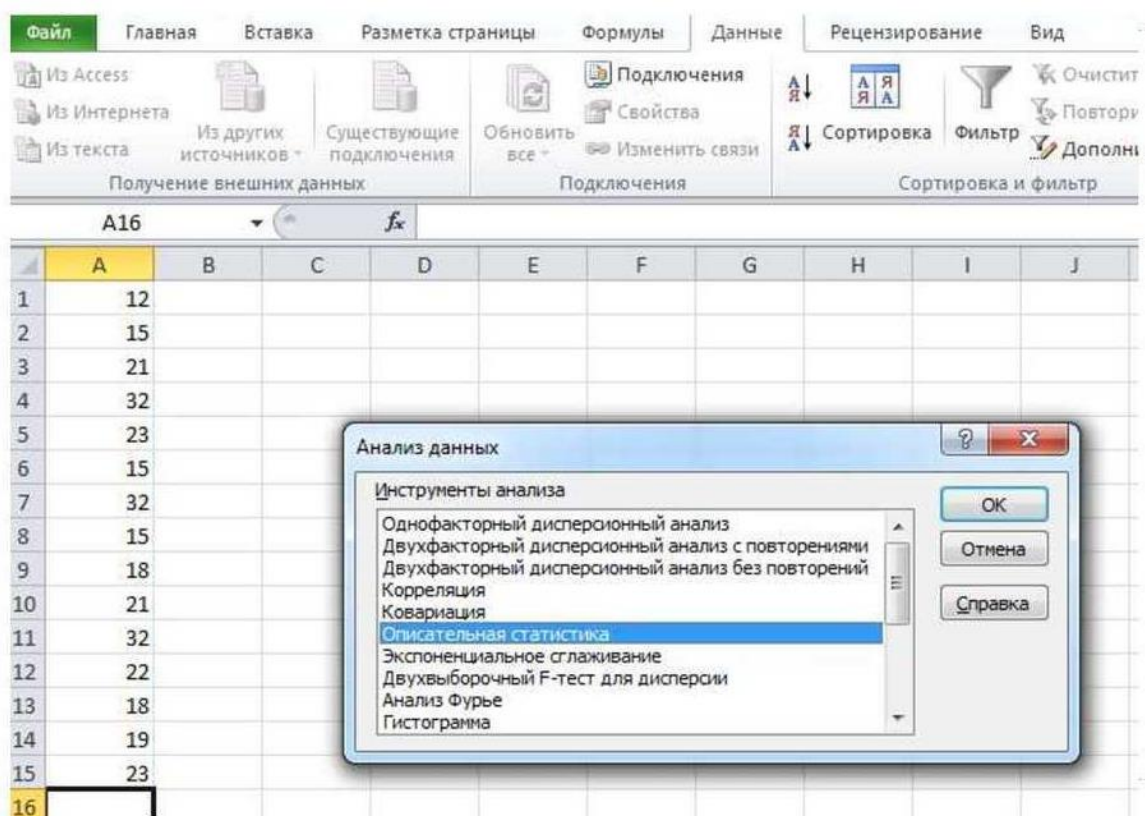


Рисунок 6 – Статистический пакет анализа данных в MS Excel

На рисунке 7 представлено, что с помощью надстройки MS Excel Neural Excel можно моделировать нейронные сети [5].

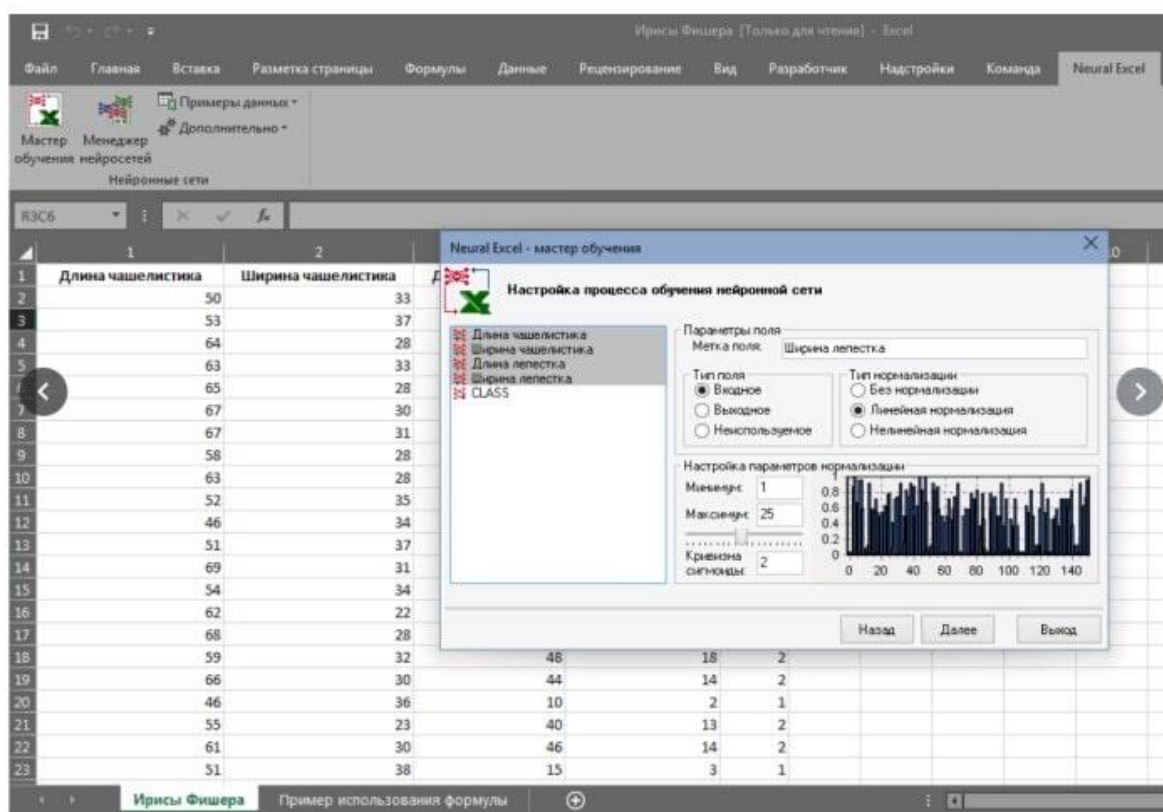


Рисунок 7 – Интерфейс MS Excel Neural Excel

Обработку экспериментальных данных можно проводить на графических калькуляторах от компании CASIO. Они могут проводить вычисления сложных и объемных выражений, а также содержат статистический аппарат, используемый для обработки экспериментальных данных, и графический анализ результатов эксперимента [6].

Языки программирования также используются для обработки и анализа данных в физике. Чаще всего используются языки программирования Python и C++ на базе пакетов pyROOT и ROOT. ПО ROOT – это пакет объектно-ориентированных программ и библиотек на языке программирования C++, разработанных для обработки экспериментальных данных физики высоких энергий. Взаимодействовать с пакетом ROOT на языке Python можно с помощью модуля расширений pyROOT посредством интерпретатора Python. Также язык Python является удобным средством разработки распределенных систем и сетевого программирования [7].

В одном из экспериментов на Большом адронном коллайдере (Large Hadron Collider, LCH), связанном с анализом данных, было разработано ПО OSTAP, обеспечивающее более понятное и удобное для пользователя представление интерфейса для программных пакетов ROOT и pyROOT. ПО OSTAP был основан на функциях языка программирования Python, что позволило расширить существенную функциональность и включить инструменты, необходимые для выполнения анализа [7].

Одним из преимуществ пакета OSTAP является возможность параллельного запуска нескольких задач на серверах, которые используются для обработки данных [7].

Все большую популярность в обработке и анализе данных физического эксперимента набирают нейронные сети. Одним из примеров является Гамильтоновы нейронные сети (ГНС) глубокого машинного обучения. Они реализуют гамильтонов путь в нейронной сети и образуют «закон сохранения энергии в физике» [8]. Модель применима к механике Гамильтона, моделированию фазового поля и теории Ландау.

Гамильтонова нейронная сеть предлагает способ изучения гамильтоновой динамики на основе данных путем обучения градиентов нейронной сети для соответствия производной по времени целевой системы контролируемым образом [9].

Гамильтонова нейронная сеть изучает дифференцируемую функцию $H(q, p)$, которая отображает состояние системы в скалярную величину.

$$L_{HNN} = \frac{1}{2} \left[\left(\frac{\partial H}{\partial p} - \frac{dq}{dt} \right)^2 + \left(\frac{\partial H}{\partial p} - \frac{dp}{dt} \right)^2 \right], \quad (8)$$

где $\frac{\partial H}{\partial p}$ и $\frac{dq}{dt}$ – производные вычисляются обратным распространением ошибки.

Такое обучение применима, известны истинное пространственное состояние и его производная по времени. Отсюда следует, что в большинстве экспериментов гамильтониан изучался не из наблюдений, а из пространства истинного состояния. Поэтому предлагается гамильтониан параметризовать с нейронной сетью [9].

Гамильтоновы нейронные сети включают в себя Лагранжевы нейронные сети. Они основаны на физико-информационном методе обучения при изучении динамики пространства состояния объектов управление и могут параметризовать произвольные лагранжианы с использованием нейронных сетей [9].

На рисунке 8 представлена схема лагранжевой нейронной сети с тремя скрытыми слоями:

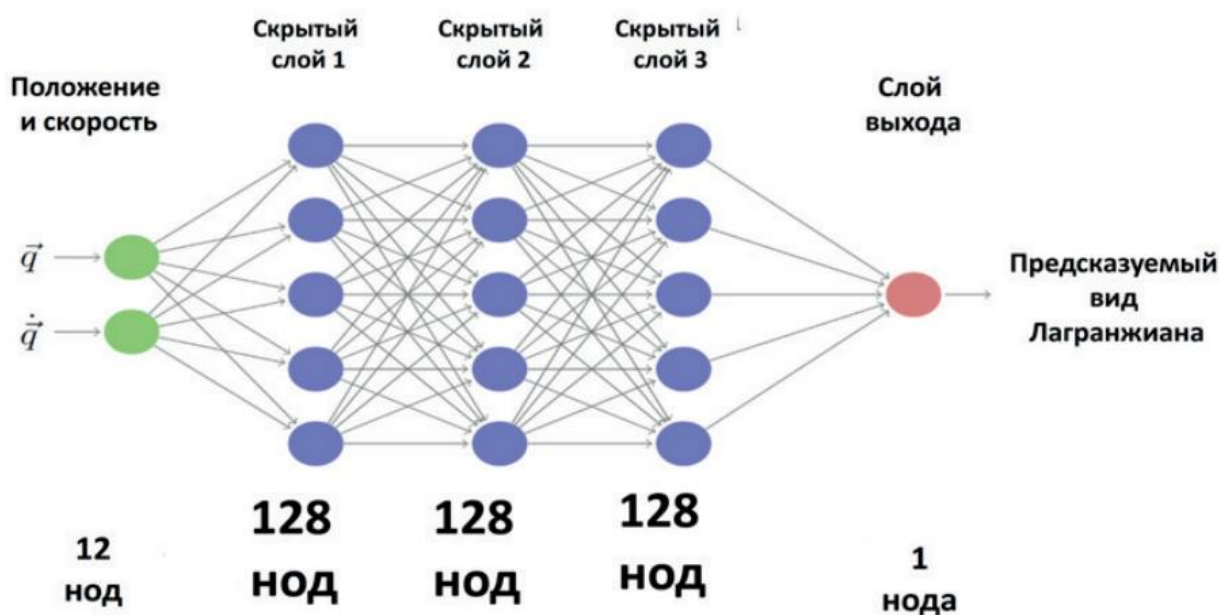


Рисунок 8 – Лагранжева нейронная сеть

Лагранжевы нейронные сети на вход получают положение и угловую ориентацию с соответствующими производными по времени. Результатом является лагранжиан системы [9].

В настоящее время наиболее популярным в области анализа данных является язык программирования Python. Он имеет ряд библиотек, которые позволяют эффективно анализировать данные, без обладания глубокими знаниями в области математической статистики и машинного обучения. Это является основным преимуществом перед другими аналогами. Также ПО для работы с языком программирования Python является бесплатным в отличие от Matlab.

2 Интеллектуальный анализ данных

С каждым годом интеллектуальный анализ данных становится более актуальным в разных сферах деятельности человека.

Интеллектуальный анализ данных – это совокупность математических моделей, численных методов, программных средств и информационных технологий, обеспечивающих обнаружение в эмпирических данных доступной для интерпретации информации [11].

Целью интеллектуального анализа данных является обнаружение неявных закономерностей в наборах данных.

К основным задачам интеллектуального анализа относят [10]:

- задача классификации – для каждого варианта определяется категория или класс, к которому он относится. Множество классов заранее известно,
- задача регрессии – производится поиск шаблонов для определения числового значения. Предсказываемый параметр – число из непрерывного диапазона,
- задача прогнозирования – прогнозируются новые значения на основе имеющихся значений числовой последовательности или нескольких последовательностей,
- задача кластеризации – деление множества объектов на кластеры схожих по параметрам. В отличие от классификации, число кластеров может быть заранее неизвестно. Характеристики будут определяться в ходе построения кластеров исходя из степени близости объединяемых объектов по совокупности параметров,
- задача определения взаимосвязей или задача поиска ассоциативных правил – определение часто встречающихся наборов объектов среди множества подобных наборов.

Методы классификации, моделирования и прогнозирования составляют основу интеллектуального анализа данных. Также к методам интеллектуального анализа данных относят статистические методы (дескриптивный анализ, дисперсионный анализ, дискриминантный анализ, корреляционный и регрессионный анализ, факторный анализ) [10].

Применение статистических методов анализа данных требует владение теорией вероятностей и математической статистики, но наглядное представление результатов данных позволяет людям, не имеющим специальной математической подготовки, использовать инструменты интеллектуального анализа данных.

Выбирая метод человек опирается на свойства и характеристики, такие как точность, интерпретируемость, быстрота, масштабируемость, гибкость и трудоемкость. Основная часть аналитических методов, используемых в интеллектуальном анализе данных – это математические методы и алгоритмы. Однако, благодаря новым возможностям технических и программных средств, большинство методов были разработаны в рамках теории искусственного интеллекта [10, 11].

Примеры методов и алгоритмов интеллектуального анализа данных:

- искусственные нейронные сети,
- дерево решений,
- байесовские сети,
- линейная регрессия,
- метод опорных векторов,
- метод ближайшего соседа и k-ближайшего соседа,
- корреляционно-регрессионный анализ,
- иерархические методы кластерного анализа,
- метод ограниченного перебора,
- эволюционное программирование, генетические алгоритмы и другое [10].

Рассмотрим некоторые из них.

Метод k-ближайших соседей основан на поиске ближайших объектов с известными значениями целевой переменной [11], где k – это количество соседних объектов в пространстве признаков, сравниваемых с классифицируемым объектом.

Данный метод имеет как преимущества, так и недостатки. К достоинствам относят просто реализации, универсальность, возможность использования для классификации и регрессии. Недостатками метода являются замедленная работа при увеличении объема выборки [11].

Для обработки данных с размытыми значениями истинности используется нечеткая логика.

В задачах классификации реализуется метод «дерево решений». Суть метода – каждой внутренней вершине в соответствие ставится некоторый атрибут, каждая ветвь, выходящая из вершины – возможное значение атрибута, конкретный класс сопоставлен листу дерева. Цель дерева решений – создание модели, предсказывающей значение целевой переменной на основе нескольких переменных на входе [11].

Кластеризация применяется для моделирования и использует методы, обрабатывающие реляционные данные.

Этапы кластеризации:

- выделение характеристик. Выбираются свойства, характеризующие объекты (количественные, качественные характеристики). Выделяются наиболее важные свойства объектов и нормализуются,
- определение метрики. По выбранной метрике определяется близость объектов.

Сравнение разных методов интеллектуального анализа представлено в таблице 1.

Таблица 1 – Сравнение методов интеллектуального анализа [11]

Методы	Сложность	Точность	Размер обучаемых данных	Интерпретация
Дерево принятия решения	высокая	низкая	средний	низкая
Кластеризация	средняя	средняя	большой	низкая
Нейронные сети	высокая	высокая	большой	высокая
Нечеткая логика	средняя	низкая	-	высокая
Метод k-ближайших соседей	высокая	Очень низкая	большой	высокая

Важным инструментом для реализации аналитических задач и получения результатов интеллектуальном анализе данных является методология разработки проекта. Наиболее популярными методологиями являются:

- SEMMA (Sample, Explore, Modify, Model, and Assess),
- CRISP-DM (Cross-industry standard process for data mining),
- TDSP (Team Data Science Process).

Более подробно рассмотрим методологию CRISP-DM. Это стандартная методология разработки проектов интеллектуального анализа данных, предлагающая систематический итеративный подход к решению аналитических задач и охватывающая все основные этапы проекта [12].

CRISP-DM применяется в проектах, в которых используются компьютерные методы для сбора данных, обработки и анализа, визуализации больших объемов данных.

Business Understanding/ Бизнес-анализ	Data Understanding/ Анализ данных	Data Preparation/ Подготовка данных	Modeling/ Моделирование	Evaluation/ Оценка решения	Deployment/ Внедрение
Determine Business Objectives/ Определение бизнес-целей	Collect Initial Data/ Сбор данных	Select Data/ Выборка данных	Select Modeling Techniques/ Выбор алгоритмов	Evaluate Results/ Оценка результатов	Plan Deployment/ Внедрение
Assess Situation/ Оценка текущей ситуации	Describe Data/ Описание данных	Clean Data/ Очистка данных	Generate Test Design/ Подготовка плана тестирования	Review Process/ Оценка процесса	Plan Monitoring and Maintenance/ Планирование мониторинга и поддержки
Determine Data Mining Goals/ Определение целей аналитики	Explore Data/ Изучение данных	Construct Data/ Генерация данных	Build Model/ Обучение моделей	Determine Next Steps/ Определение следующих шагов	Produce Final Report/ Подготовка отчета
Produce Project Plan/ Подготовка плана проекта	Verify Data Quality/ Проверка качества данных	Integrate Data/ Интеграция данных	Assess Model/ Оценка качества моделей		Review Project/ Ревью проекта
		Format Data/ Форматирование данных			

Рисунок 9 – основные этапы методологии CRISP-DM

Интеллектуальный анализ данных по методологии CRISP-DM осуществляется в шесть этапов (рисунок 9, 10):

1. понимание бизнес-целей. На данном этапе определяются задачи исследования и основная цель работы,
2. анализ данных. Происходит сбор данных, их описание и изучение,
3. подготовка данных. Этот этап может занимать до 50-80% времени разработки. Он охватывает все действия, необходимые для создания окончательного набора данных, на которых будет происходить обучение модели. Формируются выборки, происходит очистка данных и их разделение на обучающую и тестовую выборку,
4. моделирование. Происходит обучение модели, оценка ее производительности по отношению к идеальной модели. Используются различные алгоритмы обучения,
5. оценка. Оцениваются модели, разработанные на предыдущем этапе – соответствуют ли они целям и полезны ли они для проекта,
6. реализация. Происходит внедрение модели и мониторинг производительности. По оценки эффективности разработанных решений могут вноситься корректировки при необходимости [12].

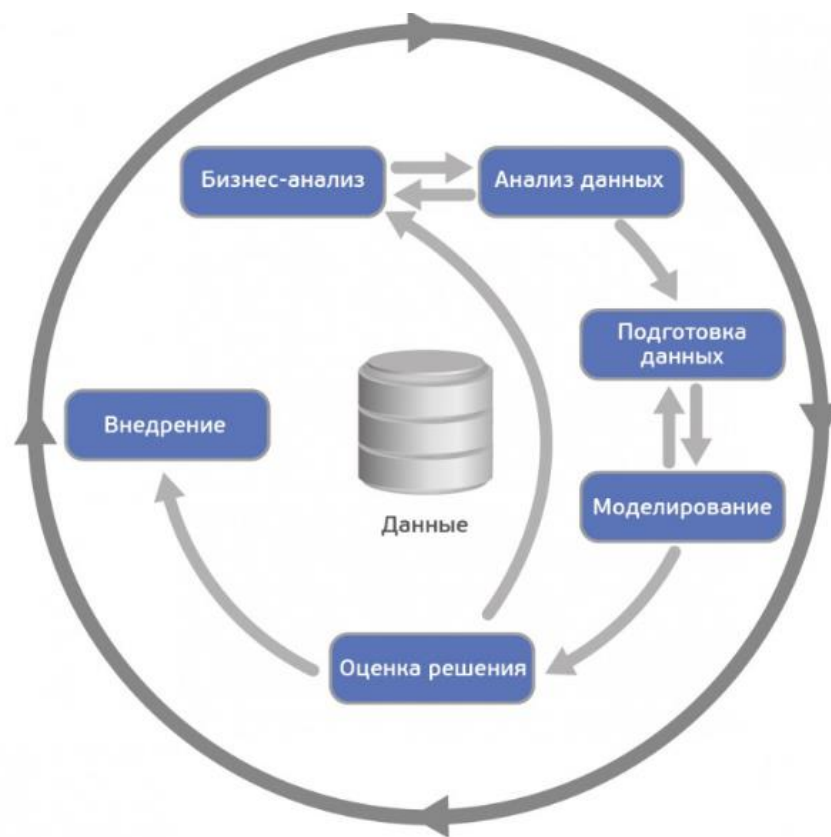


Рисунок 10 – цикл проекта

2.1 Предварительная обработка данных

Описательно-статистический анализ выявляет закономерные характеристики изменчивости отдельных признаков. Три ключевыми задачами описательной статистики являются:

1. оценка центральной тенденции распределения,
2. оценка отклонения от центра распределения,
3. анализ закона распределения переменной (только для параметрических переменных).

В описательной статистике используются количественный и визуальный подход. Количественный подход описывает общие численные характеристики; визуальный подход иллюстрирует данные с помощью диаграмм, гистограмм и графиков.

Если описывается и вычисляется характеристика одной переменной, то выполняется одномерный анализ. Двумерный анализ ищет статистические связи

между двумя переменными, а многомерный анализ применяется для нескольких переменных одновременно.

Виды метрик описательной статистики:

- центральные метрики – среднее, медиана и стандартное отклонение,
- метрики оценки вариативности данных – дисперсия и стандартное отклонение,
- метрики оценки корреляции или взаимозависимости.

В языке программирования Python существует множество библиотек для описательной статистики. Самыми популярными являются:

- Python statistics — встроенная библиотека Python для описательной статистики. Исполется, если наборы данных не слишком велики или если нельзя полагаться на импорт других библиотек,
- NumPy — сторонняя библиотека для численных вычислений, оптимизированная для работы с одномерными и многомерными массивами. Его основным типом является тип массива называется ndarray. Эта библиотека содержит множество подпрограмм для статистического анализа,
- SciPy — сторонняя библиотека для научных вычислений, основанная на NumPy. Она предлагает дополнительную функциональность по сравнению с NumPy, в том числе scipy.stats для статистического анализа,
- Pandas — сторонняя библиотека для численных вычислений, основанная на NumPy. Она отлично справляется с обработкой помеченных одномерных (1D) данных с объектами Series и двумерных (2D) данными объектами DataFrame,
- Matplotlib— сторонняя библиотека для визуализации данных.

Проведем статистический анализ экспериментальных данных для задачи «движение тела, брошенного под углом к горизонту».

Тело, брошенное под углом α к горизонту с начальной скоростью v_0 , без учета сопротивления воздуха будет лететь по параболе и через некоторое время достигнет земли. Решая данную задачу, разложим начальную скорость v_0 на горизонтальные и вертикальные составляющие:

$$v_x^{(0)} = v_0 \cos \alpha, v_y^{(0)} = v_0 \sin \alpha \quad (9)$$

Движение по горизонтали равномерным. Движение по вертикали является равнозамедленным до достижения верхней точки траектории и равноускоренным после ($v_y = v_y^{(0)} - gt$). В верхней точке скорость $v_y = 0$, тогда $v_y^{(0)} = gt$. Отсюда время достижения верхней точки будет:

$$\tau = \frac{v_y^{(0)}}{g} = \frac{v_0 \sin \alpha}{g} \quad (10)$$

Высота верхней точки h равна:

$$h = v_y^{(0)} \tau - \frac{g \tau^2}{2} = \frac{v_0^2 \sin^2 \alpha}{g} - \frac{g v_0^2 \sin^2 \alpha}{2g^2} = \frac{v_0^2 \sin^2 \alpha}{2g} \quad (11)$$

Полный путь тело пройдет за время 2τ , двигаясь равномерно вдоль оси x со скоростью $v_x^{(0)}$:

$$l = 2\tau v_x^{(0)} = v_0 \cos \alpha \cdot 2 \frac{v_0 \sin \alpha}{g} = \frac{v_0^2 \sin 2\alpha}{g} \quad (12)$$

Из выведенной формулы видим, что дальность полета зависит от начальной скорости и угла бросания.

Загрузим данным и посмотрим на основные колонки из таблицы (рисунок 11):

```
df=pd.read_csv('/content/experiment.csv')  
df.head(10)
```

	v	a	l
0	10	50	20.098117
1	16	1	1.823321
2	19	88	5.139201
3	26	80	47.184820
4	5	3	0.533308
5	12	4	4.089985
6	34	30	204.311299
7	14	27	32.360680
8	19	45	73.673469
9	29	7	41.521697

Рисунок 11 – Загрузка экспериментальных данных

Рассмотрим график зависимости дальности полета от начальной скорости. В качестве первой выборки возьмем начальную скорость полета тела, второй – дальность полета. На рисунке 12 представлен график, из которого видно, что чем больше начальная скорость, тем больше будет дальность полета тела .

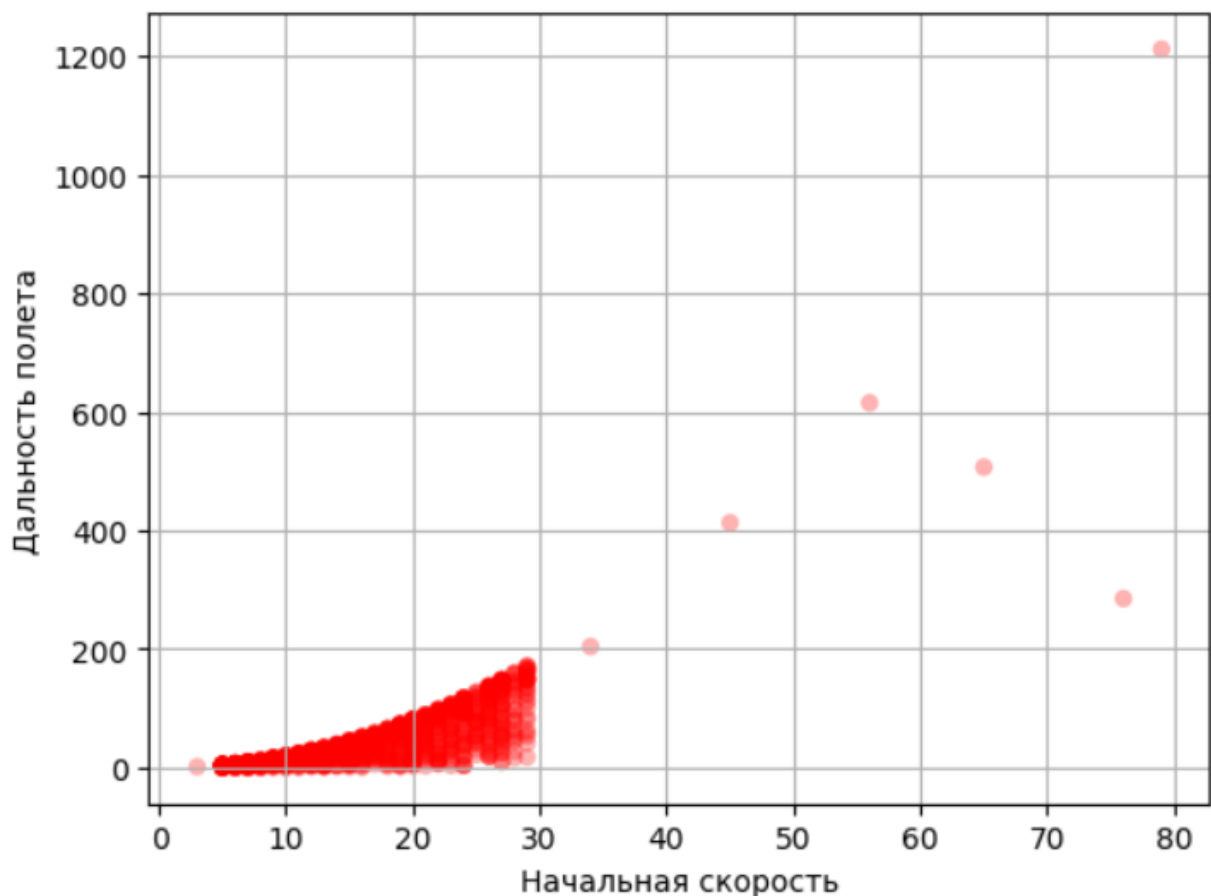


Рисунок 12 – график зависимости дальности полета от начальной скорости

Одним из важнейших понятий в теории вероятностей является математическое ожидание. Оно оценивается как среднее арифметическое наблюдаемых значений случайной величины. Значение математического ожидания для исходных данных эксперимента представлены на рисунке 13.

```
df.mean()
```

```
v    17.135000
a    43.091000
l    48.468712
dtype: float64
```

Рисунок 13 – математическое ожидание исходных данных

Числом, характеризующим выборку, является медиана. Если все элементы выборки будут различны, то половина элементов выборки будет больше

медианы, а другая половина меньше. Основное различие между математическим ожиданием и медианой – связь с выбросами и экстремальными значениями. Среднее значение сильно зависит от выбросов, а медиана либо зависит от выбросов незначительно, либо не зависит. Сравнение медианы и математического ожидания – один из способов обнаружения выбросов в данных. Значение медианы для исходных данных эксперимента представлены на рисунке 13.

```
df.median()

v      17.000000
a      42.000000
l      32.980969
dtype: float64
```

Рисунок 14 – медианы исходных данных

Из рисунков 13 и 14 видно, что между математическим ожиданием и медианой есть расхождения. Соответственно, в данных присутствуют выбросы.

Еще одной центральной метрикой является мода, показывающая наиболее часто встречаемое значение во множестве наблюдений. Чаше мода вычисляется для нечисловых данных. С ее помощью определяют наиболее популярные типы продукта при планировании производства и прогнозе продаж. Мода для исходных данных представлена на рисунке 15.

```
print('Значение моды начальной скорости: ', st.mode(df.v))
print('Значение моды угла бросания: ', st.mode(df.a))
print('Значение моды дальности полета: ', st.mode(df.l))
```

```
Значение моды начальной скорости:  20
Значение моды угла бросания:  31
Значение моды дальности полета:  3.60516536576193
```

Рисунок 15 – значение моды для исходных данных

Для описания данных центральных метрик недостаточно. Необходимы метрики оценки вариантности данных, которые количественно определяют разброс точек данных.

Насколько сильно значения отклоняются от среднего позволяет понять дисперсия – квадрат стандартного отклонения. Для каждого значения вычисляется разница между имеющимся значением и средним значением, которая возводится в квадрат, так как она может быть положительной или отрицательной. Все полученные значения суммируются и делятся на количество наблюдений минус.

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^2 (x_i - \bar{x})^2 \quad (13)$$

В языке программирования Python в библиотеке statistic есть функция `pvariance()`, которая вычисляет дисперсию (рисунок 16).

```
print('Дисперсия:')  
print('Начальная скорость', st.pvariance(v))  
print('Угол бросания', st.pvariance(a))  
print('Дальность полета', st.pvariance(l))
```

```
Дисперсия:  
Начальная скорость 63.072775  
Угол бросания 638.802719  
Дальность полета 3857.818751231164
```

Рисунок 16 – дисперсия для исходных данных

На рисунке 17 представлен еще один показатель разброса данных – стандартное отклонение, связанное с оценкой дисперсии.

```
print('Стандартное отклонение:')  
print('Начальная скорость', st.pstdev(v))  
print('Угол бросания', st.pstdev(a))  
print('Дальность полета', st.pstdev(l))
```

Стандартное отклонение:

Начальная скорость 7.941837004119387

Угол бросания 25.274546860428575

Дальность полета 62.1113415668279

Рисунок 17 – стандартное отклонение для исходных данных

С помощью графика на рисунке 18 можно посмотреть разброс значений начальной скорости:

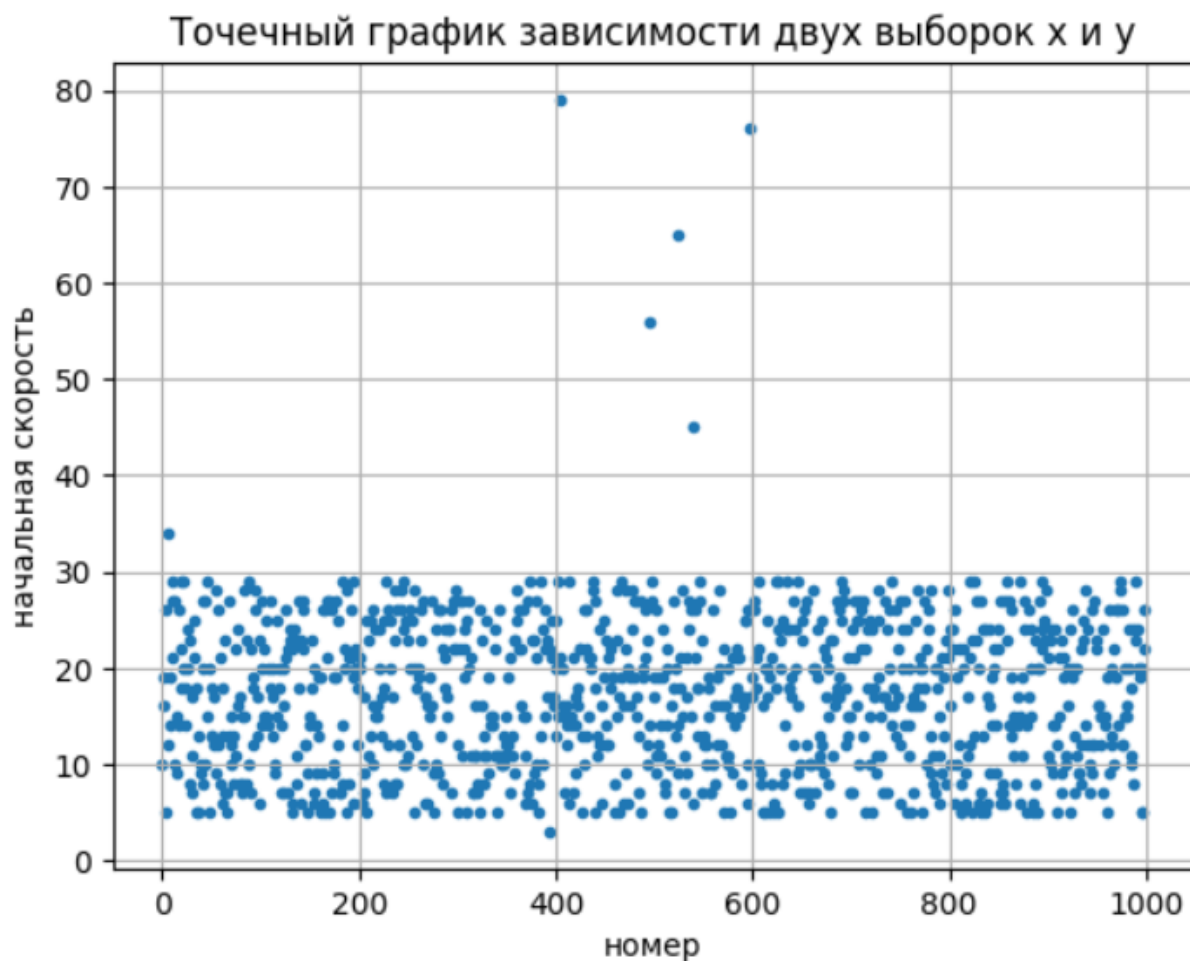


Рисунок 18 – разброс значений начальной скорости

Третье метрикой описательной статистики является оценка корреляции или взаимосвязи. Корреляция – это согласованные изменения двух или большего количества признаков. Коэффициент корреляции принимает значение от -1 до 1.

$$r_{XY} = \frac{cov_{XY}}{\sigma_X \sigma_Y} = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum (X - \bar{X})^2 \sum (Y - \bar{Y})^2}} \quad (14)$$

где cov_{XY} – ковариация, показывающая как связаны две переменные. Положительная ковариация означает, что переменные связаны прямой зависимостью, а отрицательная ковариация указывает на обратную зависимость.

Для порядковых переменных используются коэффициенты корреляции:

- коэффициент ранговой корреляции Спирмена,
- коэффициент ранговой корреляции Кендалла,
- коэффициент ранговой корреляции Гудмена–Краскела.

С помощью коэффициента Пирсона можно определить только силу линейной взаимосвязи между переменными, другие виды взаимосвязей выявляются методами регрессионного анализа. Чем ближе коэффициент к 0, тем более независимы переменные. Для начальной скорости тела и дальности полета коэффициент Пирсона близок к 1, что показывает их взаимосвязь.

На рисунке 19 видим, что для начальной скорости и дальности полета корреляция Пирсона близка к 1, что показывает их взаимосвязь.

```
x = df['v']
y = df['l']
corr, p_value = pearsonr(x, y)
print('Коэффициент корреляции Пирсона:', corr)
```

Коэффициент корреляции Пирсона: 0.7705631034194182

Рисунок 19 – коэффициент корреляции Пирсона

Данные корреляционного анализа можно представить с помощью матрицы корреляции. Из рисунка 20 видно, что дальность полета слабо взаимосвязана с углом бросания.

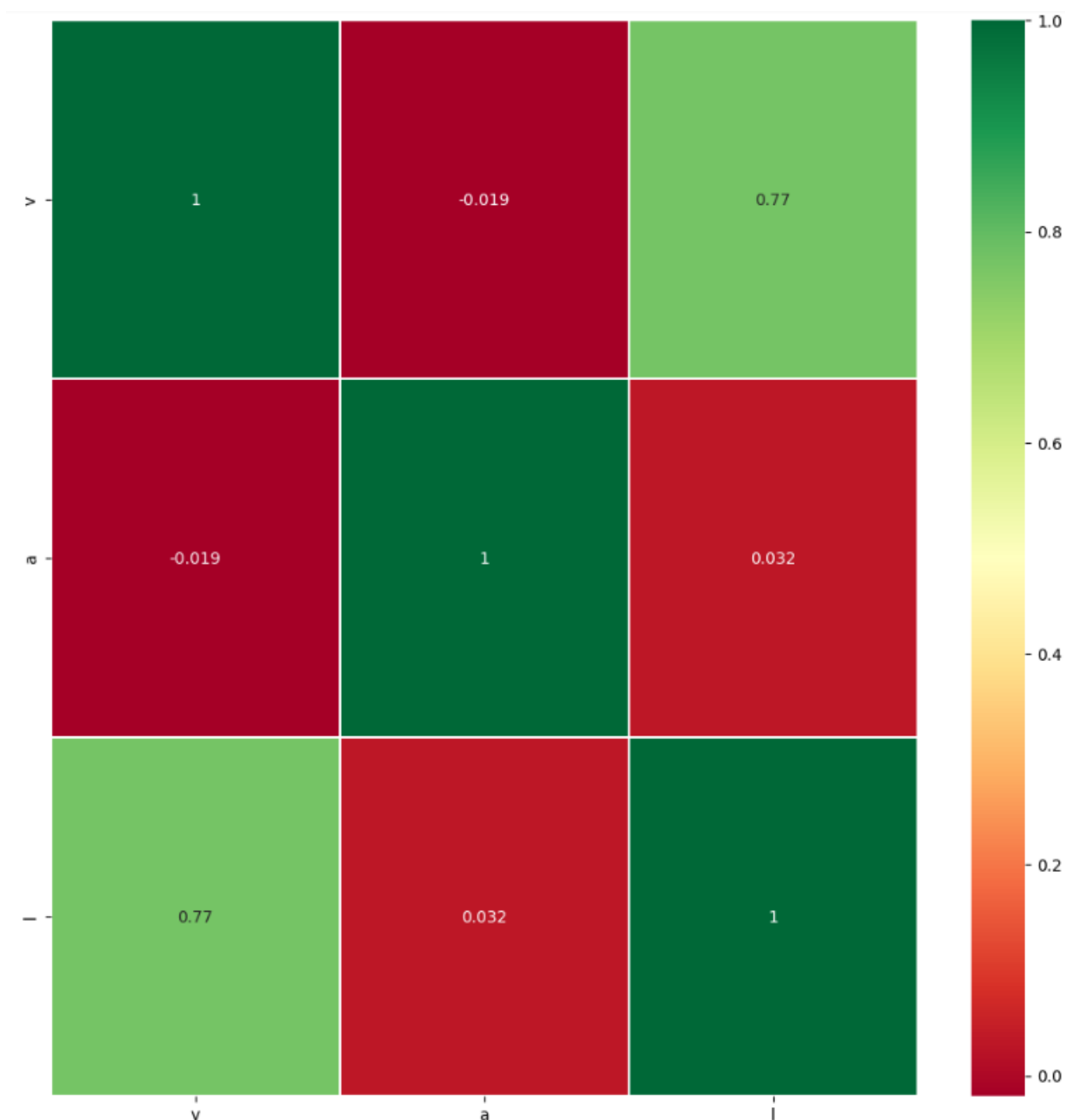


Рисунок 20 – матрица корреляции

Визуализировать набор данных в описательной статистике можно с помощью коробчатой диаграммы «ящик с усами». Она показывает диапазон, межквартильный диапазон, медиану, моду, выбросы и все квартили. В языке программирования Python график можно построить с помощью функции `boxplot()` из библиотеки Matplotlib. Функция имеет параметры:

- `x` – данные (начальная скорость),
- `vert` – устанавливает ориентацию графика (горизонтальная),
- `showmeans` – показывает среднее значение данных,
- `meanline` – представляет среднее в виде линии, когда значение `True`.

Представлением по умолчанию является точка,

- `labels` – метки данных,
- `patch_artist` – определяет, как рисовать график,
- `medianprops` – обозначает свойства линии, представляющей медиану (фиолетовая линия),
- `meanprops` – указывает свойства линии или точки, представляющей среднее значение (красная пунктирная линия).

Из рисунка 21 видно, что в выборке имеются выбросы с значением больше 35. Записи с выбросами можно удалить с помощью функции `drop()`.

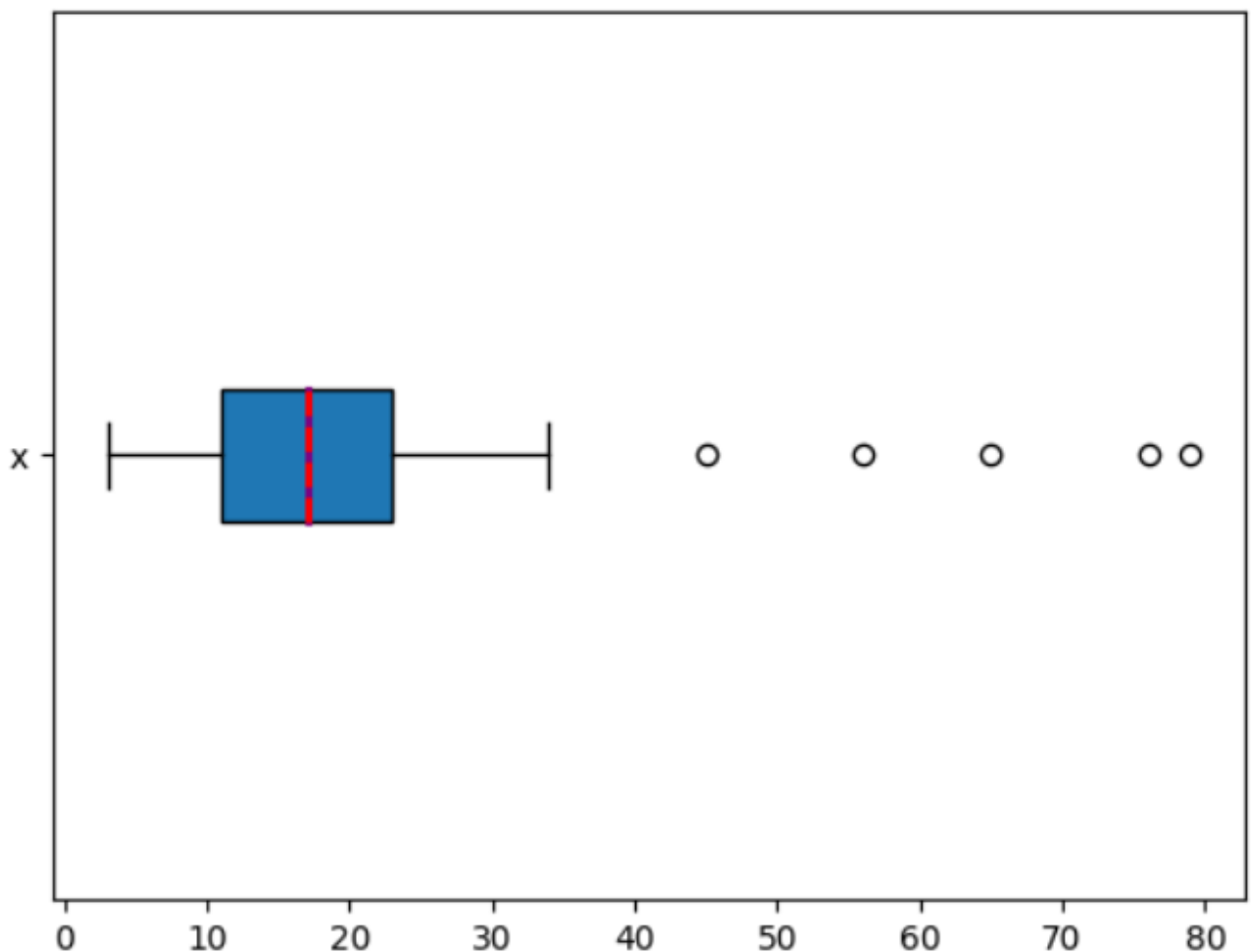


Рисунок 21 – коробчатая диаграмма для начальной скорости

На рисунке 22 представлена коробчатая диаграмма для начальной скорости без выбросов:

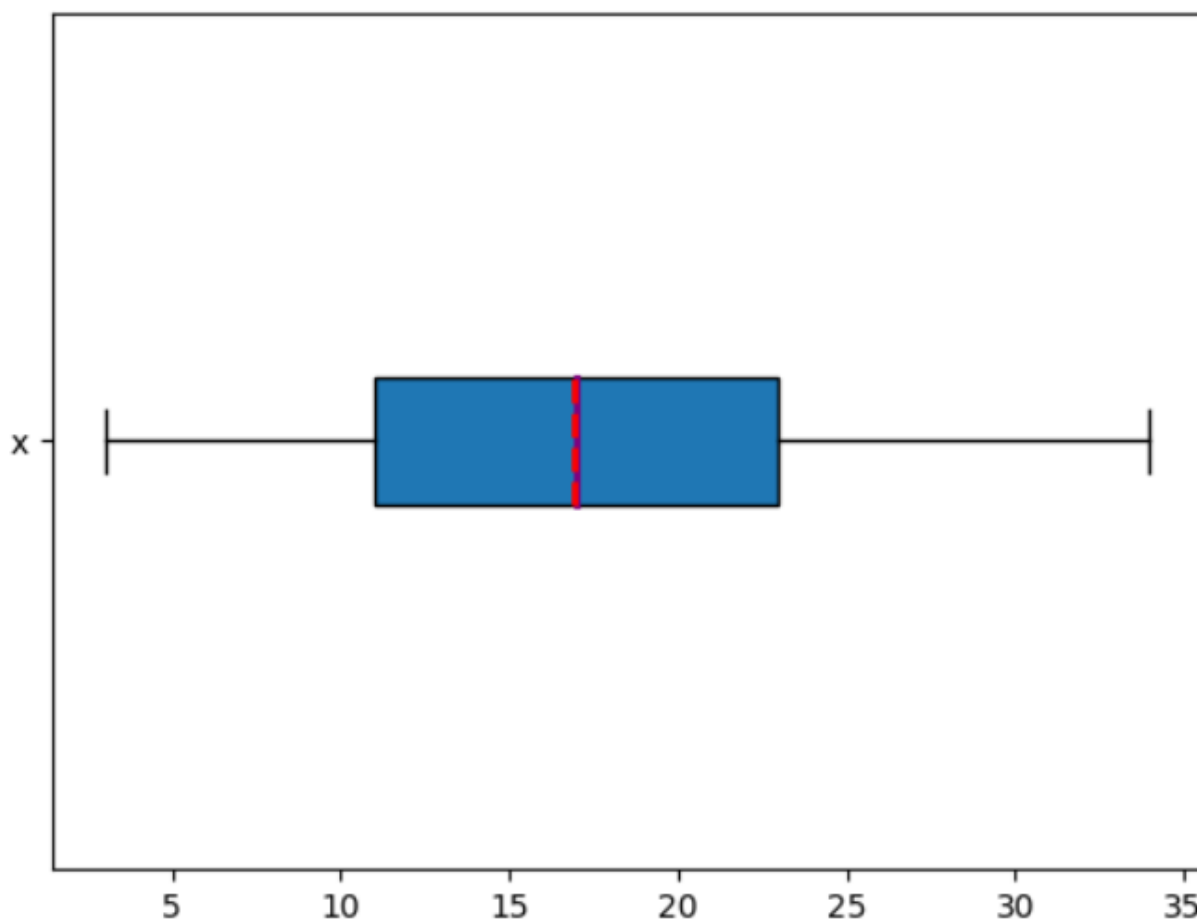


Рисунок 22 – коробчатая диаграмма для начальной скорости без выбросов

2.2 Модель в виде нейронной сети

2.2.1 Выбор нейронной сети

Все большую популярность в обработке и анализе данных физического эксперимента набирают нейронные сети. Это связано с их применением в самых разных научных областях – в физике, медицине, химии и т.д. Нейронные сети используются в решении задач, связанных с решением задач, касающихся прогнозирования, управления, классифицирования [14]. Схема простой нейронной сети представлена на рисунке 23.

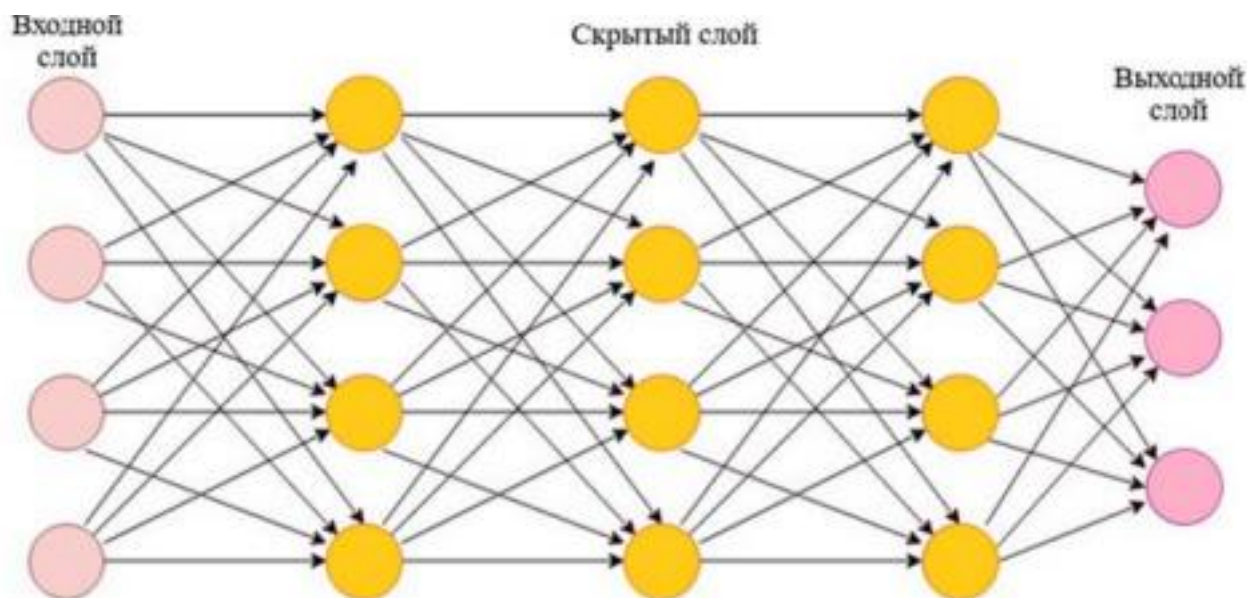


Рисунок 23 – Нейронная сеть

Главной особенностью нейронных сетей считается их нелинейность. На протяжении многих лет линейное моделирование было основным методом моделирования. Но не все задачи можно решить с помощью линейной аппроксимации. Нейронные же сети позволяют решить проблему невозможности моделирования линейных зависимостей в случае большого числа переменных [14].

Нейронная сеть – это математическая модель, построенная по принципу нейронной сети живого организма. Главной составляющей нейронной сети является нейрон. Нейрон – вычислительная единица, которая получает информацию, производит на ней простые вычисления и передает ее дальше [15]. Структура искусственного нейрона представлена на рисунке 24.

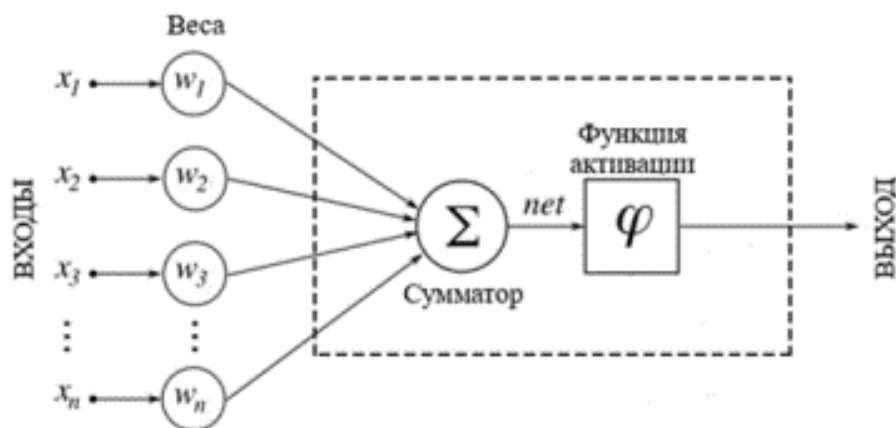


Рисунок 24 – структура искусственного нейрона

Нейроны делятся на три типа: входной, скрытый и выходной. Если нейронная сеть состоит из большего количества нейронов, то вводится понятие слоя:

- входной слой, получающий информацию,
- n скрытых слоев, обрабатывающих информацию,
- выходной слой, выводящий результат,

Каждый нейрон имеет два параметра: входные данные и выходные данные. Все нейроны связаны между собой – синапсами. У синапсов есть один параметр – вес w_i . Когда сигнал проходит через связь он умножается на ее вес. После того как сигнал прошел через все слои он попадает в сумматор, где произведения складываются. В результате получается взвешенная сумма net (рисунок 24) [16].

$$x_1 w_1 + x_2 w_2 + \dots + x_n w_n = \sum_{i=1}^n x_i w_i \quad (15)$$

Каждый слой обрабатывает полученную информацию с помощью функции активации. Функция активации – это способ нормализации входных данных. Функции активации бывают разных видов [15]:

- линейная,
- сигмоида (логистическая),
- гиперболический тангенс (relu).

Классификация нейронных сетей основана на их структуре, функциональности и способе обработки данных [17]. Она представлена на рисунке 25.

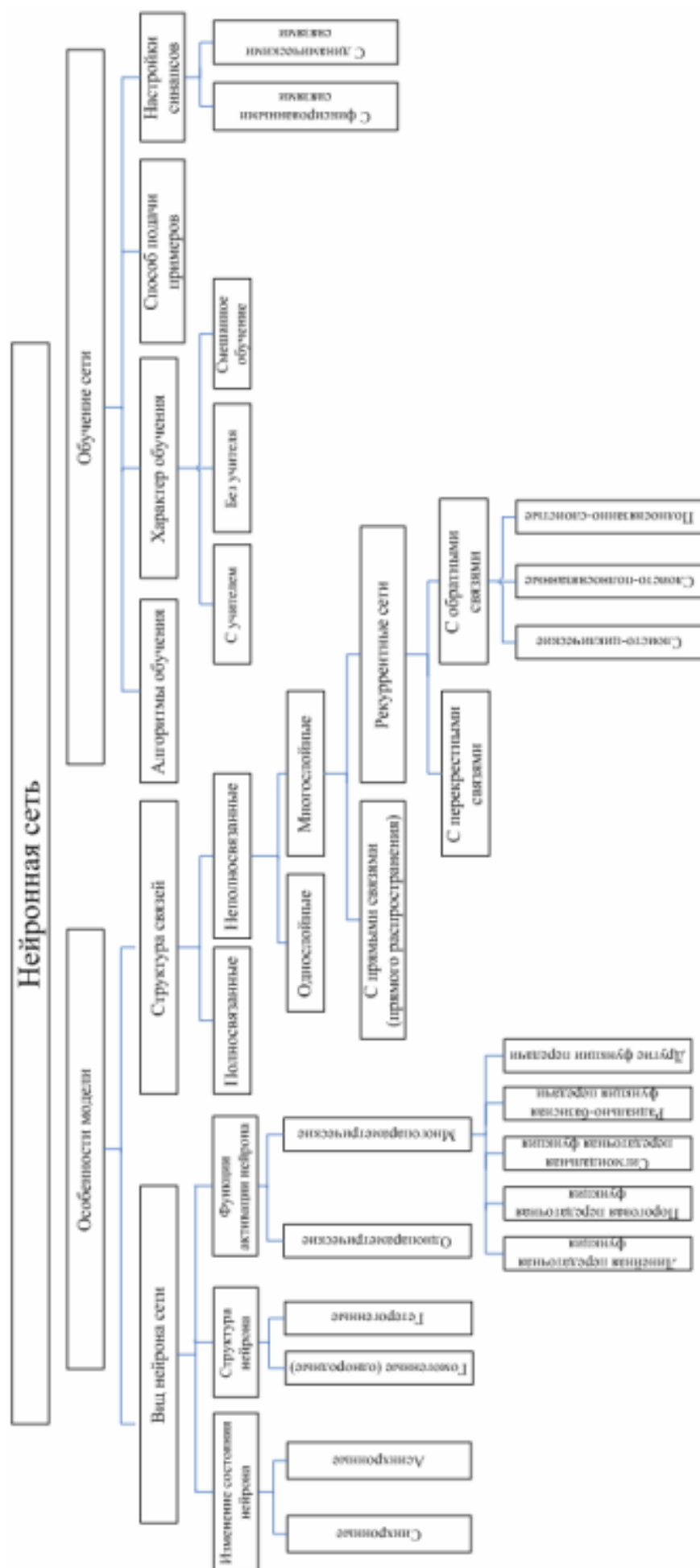


Рисунок 25 – Классификация нейронных сетей

На рисунке 26 представлена полносвязная нейронная сеть, где каждый нейрон передает выходной сигнал остальным нейронам, в том числе самому себе. Все входные сигналы подаются всем нейронам. Выходными сигналами могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети [18].

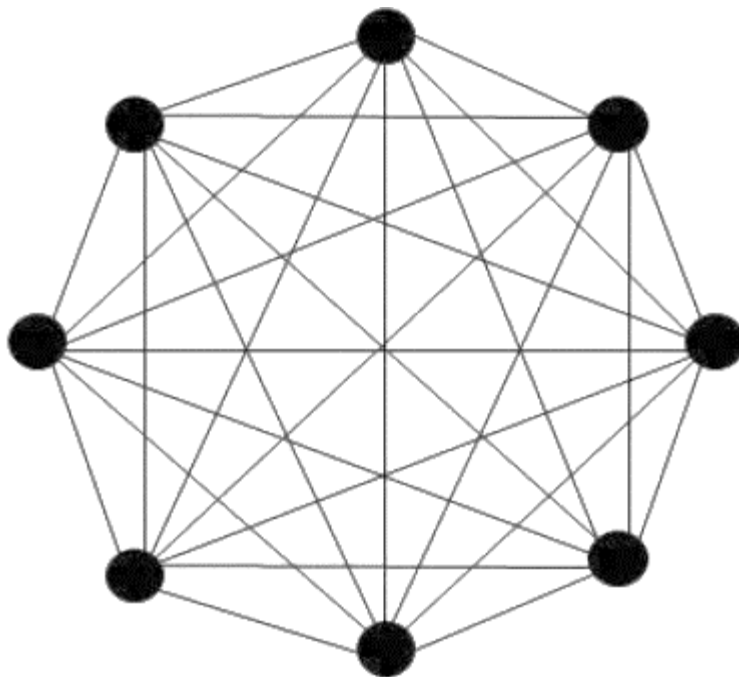


Рисунок 26 – Полносвязная нейронная сеть

Неполносвязные нейронные сети подразделяются на однослойные и многослойные, с прямыми, перекрестными и обратными связями. В нейронных сетях с прямыми связями нейроны j -ого слоя по входам могут соединяться только с нейронами нижележащих слоев. В нейронных сетях с перекрестными связями допускаются связи внутри одного слоя [18].

Среди многослойных нейронных сетей выделяют следующие типы:

- **монотонные.** Каждый слой кроме выходного разбит на два блока: возбуждающий и тормозящий. Связи между блоками тоже разделяются на тормозящие и возбуждающие. Если от нейронов блока А к нейронам блока В ведут только возбуждающие связи, то это означает, что любой выходной сигнал блока является монотонной неубывающей функцией любого выходного сигнала блока А. Если же эти связи только тормозящие, то любой выходной сигнал блока В является невозрастающей функцией любого выходного сигнала блока А [18];

– сети без обратных связей. Нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам первого скрытого слоя, и так далее вплоть до выходного, который выдает сигналы для интерпретатора и пользователя;

– сети с обратными связями. В сетях с обратными связями информация с последующих слоев передается на предыдущие. Среди них выделяют следующие:

1) слоисто-циклические, отличающиеся тем, что слои замкнуты в кольцо: последний слой передает свои выходные сигналы первому; все слои равноправны и могут как получать входные сигналы, так и выдавать выходные;

2) слоисто-полносвязанные состоят из слоев, каждый из которых представляет собой полносвязную сеть, а сигналы передаются как от слоя к слою, так и внутри слоя; в каждом слое цикл работы распадается на три части: прием сигналов с предыдущего слоя, обмен сигналами внутри слоя, выработка выходного сигнала и передача к последующему слою;

3) полносвязанно-слоистые, по своей структуре аналогичные слоисто-полносвязанным, но функционирующим по-другому: в них не разделяются фазы обмена внутри слоя и передачи следующему, на каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и последующих [18].

Схема классификации многослойных нейронных сетей представлена на рисунке 27.



Рисунок 27 – классификация многослойных нейронных сетей

Принцип работы рекуррентной нейронной сети - сохранение выходных данных слоя и передачи их обратно на входные данные, чтобы помочь в прогнозировании результата слоя [19]. Используются в задачах обработки последовательностей, анализе текста и других. Структура рекуррентной нейронной сети представлена на рисунке 28.

Первый слой формируется аналогично нейронной сети с прямой связью. После того как все вычисления пройдены, начинается повторяющийся процесс нейронной сети. То есть, от одного временного шага к следующему нейрон будет помнить информацию, которую он имел в предыдущем временном шаге. Каждый нейрон действует как ячейка памяти при выполнении вычислений [19].

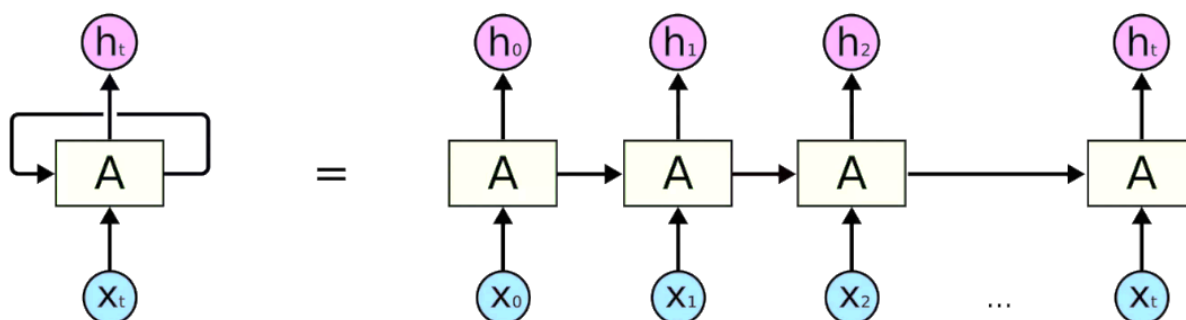


Рисунок 28 – Рекуррентная нейронная сеть

Сверточные нейронные сети состоят из разных видов слоев:

- сверточные слои,
- субдискретизирующие слои,
- слои персептрона.

Сверточные нейронные сети могут быстро обучаться за счет распараллеливания процесса свертки на каждом шаге, а также обратной свертки при распространении ошибок по сети (рисунок 29). Они были разработаны для работы с изображениями и видеоданными [18].

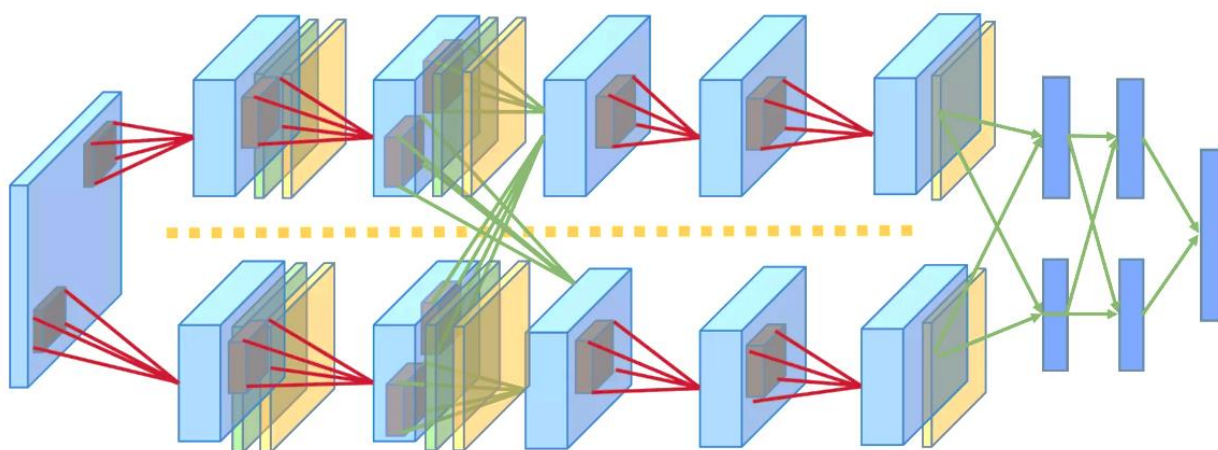


Рисунок 29 – Сверточная нейронная сеть

На рисунке 30 представлена самоорганизующиеся карты Кохонена. Это нейронная сеть без учителя, используемые для кластеризации и визуализации данных. В картах самоорганизации нейроны помещаются в узлах одно- или двухмерной решетки. В ходе конкурентного процесса нейроны избирательно настраиваются на различные входные сигналы. Особенностью самоорганизующихся карт является простая реализация при большой сложности анализа его свойств с использованием математического аппарата [20].

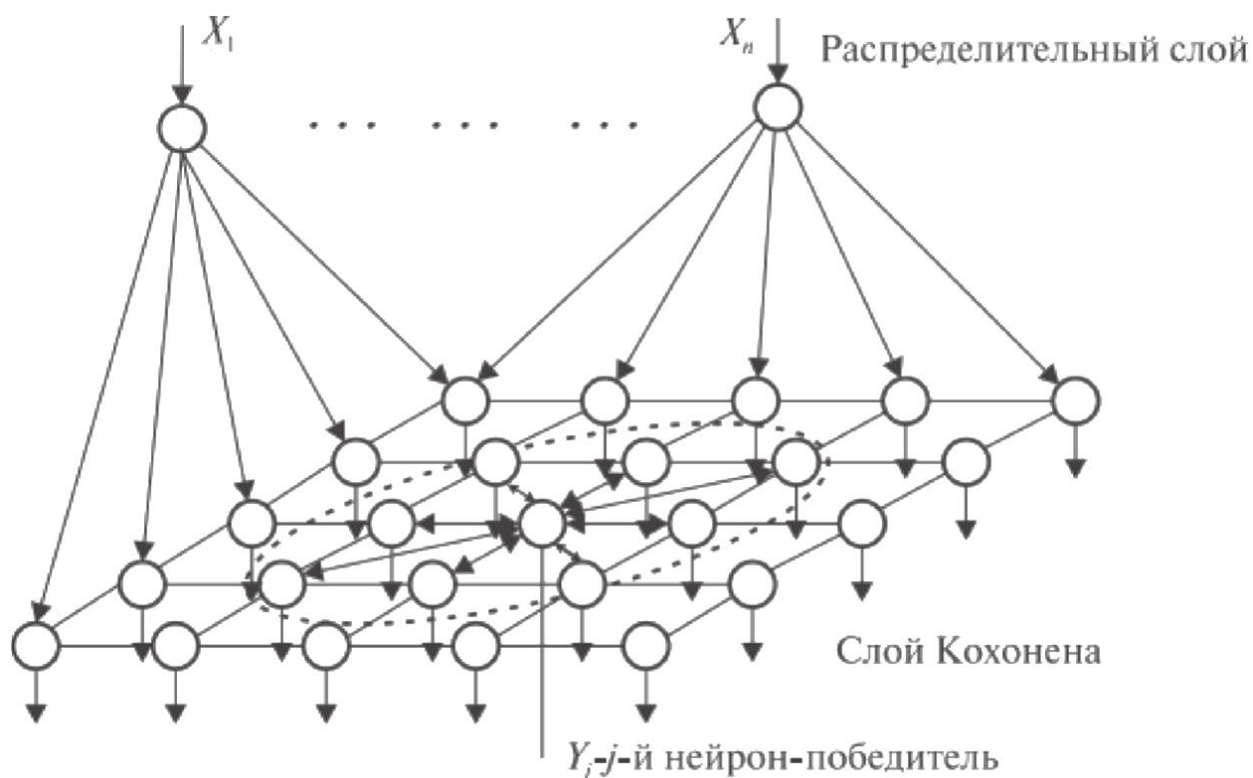


Рисунок 30 - Самоорганизующаяся карта Кохонена

Многослойный персептрон (MLP) является наиболее часто используемой архитектурой нейронной сети. Он состоит из входного слоя, одного или нескольких скрытых слоев и одного выходного слоя (рисунок 31). Данный вид нейронной сети применяется для решения разнообразных сложных задач. При этом обучение с учителем выполняется помощью алгоритма обратного распространения ошибки [20].

Три отличительных черты многослойного персептрона:

- 1) каждый нейрон имеет нелинейную функцию активации,
- 2) нейроны скрытых слоев позволяют обучаться нейронной сети решению сложных задач, последовательно извлекая наиболее важные признаки из входного вектора,
- 3) обладает высокой степенью связанности, реализуемой за счет синаптических соединений.

Два типа сигнала для MLP:

1) функциональный сигнал – входной сигнал сети, который передается в прямом направлении и достигает конца в виде выходного сигнала. Каждый нейрон, в который передается данный сигнал, вычисляет некоторую функцию от взвешенной суммы его входов,

2) сигнал ошибки – распространяется от выхода сети в обратном направлении от слоя к слою. Вычисляется нейронами на основе функции ошибки.

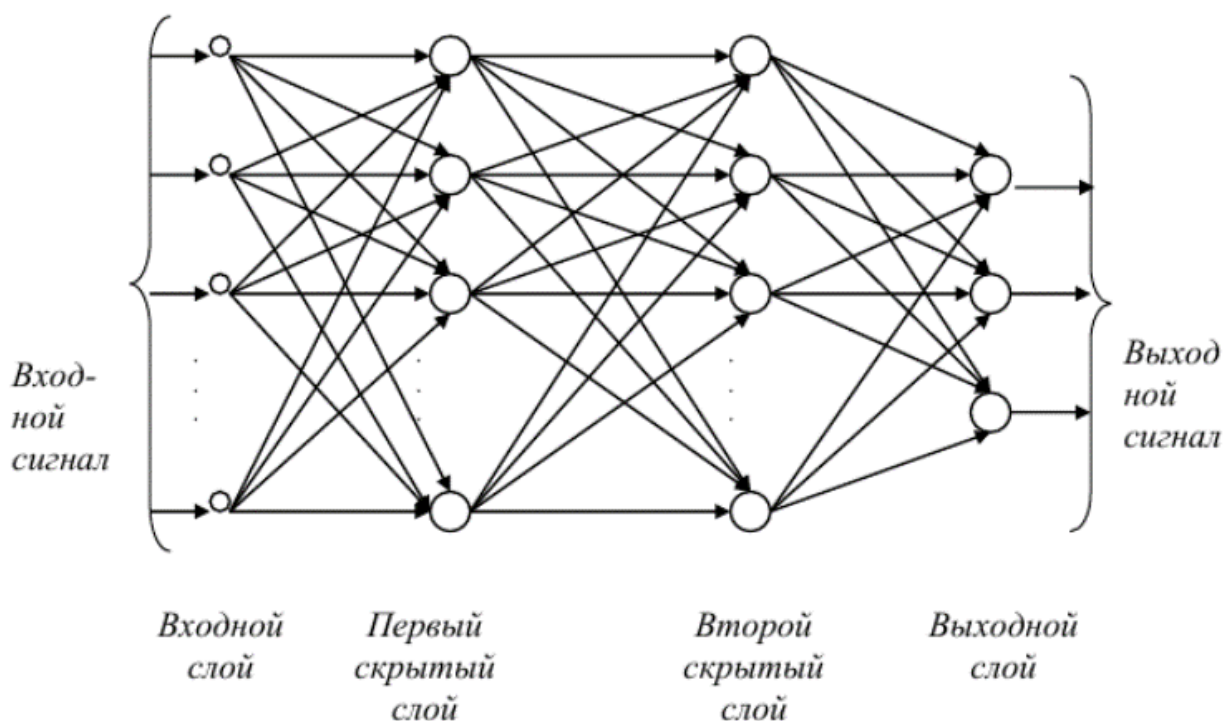


Рисунок 31 – Структура многослойного персептрона с двумя скрытыми слоями

К достоинствам многослойного персептрона относят:

- широко применяемые алгоритмы обучения,
- гарантированное получение ответа,
- моделирование функций разных степеней сложности,
- простота в использовании.

2.2.2. Разработка нейронной сети

Для разработки использовались библиотеки:

- `from sklearn.linear_model import LinearRegression:`

1) `sklearn` - библиотека машинного обучения, которая поддерживает обучение с учителем и без учителя, реализован расчет выхода, ошибки и другие функции,

2) `linear_model` – создает модели линейной регрессии. В ней реализованы методы `fit()` для обучения модели, `predict()` - расчет выходов модели при заданных входах и сохраненных в модели параметрах, `score()` - расчет значения функции ошибки для заданных примеров,

3) `LinearRegression` – метод линейной регрессии;

– `from sklearn.model_selection import train_test_split` – из модуля `model_selection` подключается функция `train_test_split()`, которая случайно разделяет данные на обучающие и тестовые в заданных долях,

– `import torch` – библиотека глубокого обучения,

– `import torch.nn as nn` – модуль для создания нейронной сети с возможностью выбора типа слоев,

– `import numpy as np` – библиотека, состоящая из объектов многомерных массивов и методов работы с ними,

– `import math` – библиотека, содержащая математические функции и константы.

Созданный класс многослойного персептрона `MLPtorch` представлен на рисунке 32.

```

class MLPptorch(nn.Module):
    def __init__(self, in_size, hidden_size, second_hidden_size, out_size):
        nn.Module.__init__(self)
        self.layers = nn.Sequential(nn.Linear(in_size, hidden_size),
                                    nn.ReLU(),
                                    nn.Linear(hidden_size, second_hidden_size),
                                    nn.ReLU(),
                                    nn.Linear(second_hidden_size, out_size),
                                    nn.ReLU(),
                                    )

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

```

Рисунок 32 – программный код класса MLPptorch

Для инициализации на вход подается размер входного, скрытого выходного слоя.

- nn.Sequential – контейнер модулей, позволяющий объединять их и запускать одновременно,
- nn.Linear – соединяет слой нейронной сети, где первый аргумент - количество узлов на i-ом слое, второй аргумент – количество узлов на i+1 слое,
- nn.Linear(in_size, hidden_size) – входной и первый скрытые слои,
- nn.Linear(hidden_size, second_hidden_size) – первый и второй скрытые слои,
- nn.Linear(second_hidden_size, out_size) – второй скрытый и выходной слои,
- nn.ReLU() – кусочно-линейная функция активации.

Данные физического эксперимента «движение тела, брошенного под углом к горизонту». Дальность полета рассчитывается по формуле (12)

Важным этапом в оценке модели является разделение исходных данных на обучающую и тестовую выборки. Для обучения отводится 70-80% исходных данных, на тестирование 20-30% данных. Тестовая выборка служит для оценки производительности модели. На языке программирования Python для этого

используется функция `train_test_split` из библиотеки `sklearn`. Указываются переменные `X` и `y`, которые представляют собой исходные данные и метки классов. Пример реализации разделения данных на обучающую и тестовую выборку представлен на рисунке 33:

```
from sklearn.model_selection import train_test_split # подключаем функцию для разделения данных
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # отдаем 30% на тест
```

Рисунок 33 – Пример разделения данных на обучающую и тестовую выборку

Переменная `X` – объединённый массив данных скорости полета и угол бросания, `y` – массив, содержащий данные о дальности полета брошенного тела (рисунок 34).

```
X = np.stack((v, a), axis=-1)
y = 1
```

Рисунок 34 – значения переменных `X` и `y`

Для обучения необходимо указать количество входных сигналов, равное количеству признаков задачи, выходных сигналов, равное количеству классов задачи, и число нейронов скрытых слоев. По умолчанию количество нейронов в скрытых слоях равно 100 (рисунок 35).

```
inputSize = X.shape[1] # количество входных сигналов равно количеству признаков задачи
hiddenSizes = 100 # задаем число нейронов скрытого слоя
second_hidden_size = 50
outputSize = 1 # количество выходных сигналов равно количеству классов задачи
```

Рисунок 35 – Программный код. Задаем количество нейронов

Задачей обучения многослойного персептрона является задача оптимизации функции по весам. Для обучения нейронных сетей и многослойного персептрона используется градиентный спуск или его модификации. Сам градиентный спуск имеет ряд проблем и недостатков. Такие как: если у функции ошибки есть несколько локальных минимумов, то в одном из них производная станет равна 0 и параметр больше не будет изменяться; необходимо задавать величину шага: если шаг будет слишком маленький, то нужно будет долго искать минимум, если большой, то можно его пропустить.

Поэтому используются различные модификации градиентного спуска, в которых можно менять шаг, ограничивать параметры и прочее.

Для обучения данного многослойного персептрона был выбран стохастический градиентный спуск `torch.optim.SGD()`. В качестве аргументов передаются параметры сети и скорость обучения (рисунок 36).

```
net = MLPptorch(inputSize,hiddenSizes,second_hidden_size,outputSize)
lossFn = nn.MSELoss()

optimizer = torch.optim.SGD(net.parameters(), lr=0.000001)
```

Рисунок 36 – программный код реализации стохастического градиентного спуска

В функцию обучения многослойного персептрона, представленной на рисунке 37, передаются данные обучающей выборки `X_train` и `y_train` и количество итераций. На каждой 500-ой итерации выводится ошибка.

```
def train(x, y, num_iter):
    for i in range(0,num_iter):
        pred = net.forward(x)
        loss = lossFn(pred, y)
        loss.backward()
        optimizer.step()
        if i%(num_iter/10)==0:
            print('Ошибка на ' + str(i) + ' итерации: ', loss.item())
    return loss.item()
```

Рисунок 37 – программный код функции обучения

```
loss_ = train(torch.Tensor(X_train),
               torch.Tensor(np.transpose(y_train))), 5000)
```

Рисунок 38 – обучение многослойного персептрона

Вывод ошибок для разного количества нейронов на скрытых слоях представлен на рисунке 39:

Ошибка на 0 итерации: 7324.35595703125
Ошибка на 500 итерации: 7324.35595703125
Ошибка на 1000 итерации: 7324.35595703125
Ошибка на 1500 итерации: 7324.35595703125
Ошибка на 2000 итерации: 7324.35595703125
Ошибка на 2500 итерации: 7324.35595703125
Ошибка на 3000 итерации: 7324.35595703125
Ошибка на 3500 итерации: 7324.35595703125
Ошибка на 4000 итерации: 7324.35595703125
Ошибка на 4500 итерации: 7324.35595703125

Рисунок 39 – Ошибка при количестве нейронов на скрытых слоях 100 и 50.

Из рисунка 39 видим, что нейронная сеть в процессе обучения заикливается и перестает обучаться. На рисунке 40 показана матрица неточности для данного количества нейронов. Точность обучения многослойного персептрона при таком количестве нейронов на скрытых слоях составляет 2%

Неверно классифицированные образцы: 293

Точность: 0.02

Text(0.5, 1.1, 'Матрица неточности Персептрон\n')

Матрица неточности Персептрон

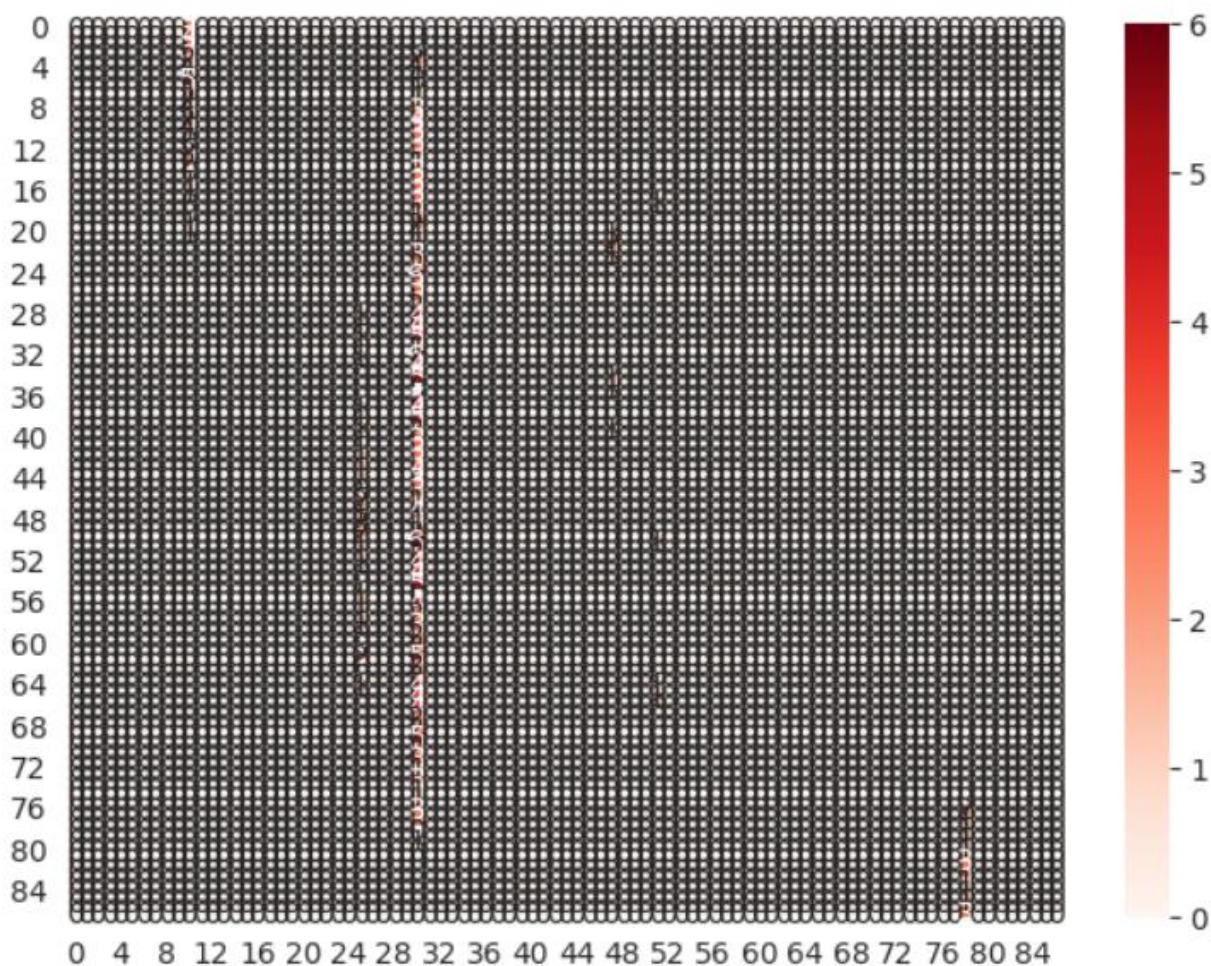


Рисунок 40 - Обучаемость многослойного персептрона. Количество нейронов на скрытых слоях 100 и 50

Наибольшую точность предсказания результатов эксперимента получилось достичь при 150 нейронах на первом скрытом слое и 100 нейронах на втором скрытом слое. Точность составила 98% (рисунок 42). Значение ошибки на итерациях представлены на рисунке 41.

При большем количестве время обучения многослойного персептрона увеличивалось, но точность становилась меньше.

Ошибка на 0 итерации: 7320.38134765625
 Ошибка на 500 итерации: 4775.7265625
 Ошибка на 1000 итерации: 4835.6513671875
 Ошибка на 1500 итерации: 4870.79833984375
 Ошибка на 2000 итерации: 5153.81591796875
 Ошибка на 2500 итерации: 4933.068359375
 Ошибка на 3000 итерации: 4755.1005859375
 Ошибка на 3500 итерации: 5060.541015625
 Ошибка на 4000 итерации: 5093.15185546875
 Ошибка на 4500 итерации: 4772.19287109375

Рисунок 41 - Ошибка при количестве нейронов на скрытых слоях 150 и 100

Неверно классифицированные образцы: 6

Точность: 0.98

Text(0.5, 1.1, 'Матрица неточности Персептрон\n')

Матрица неточности Персептрон

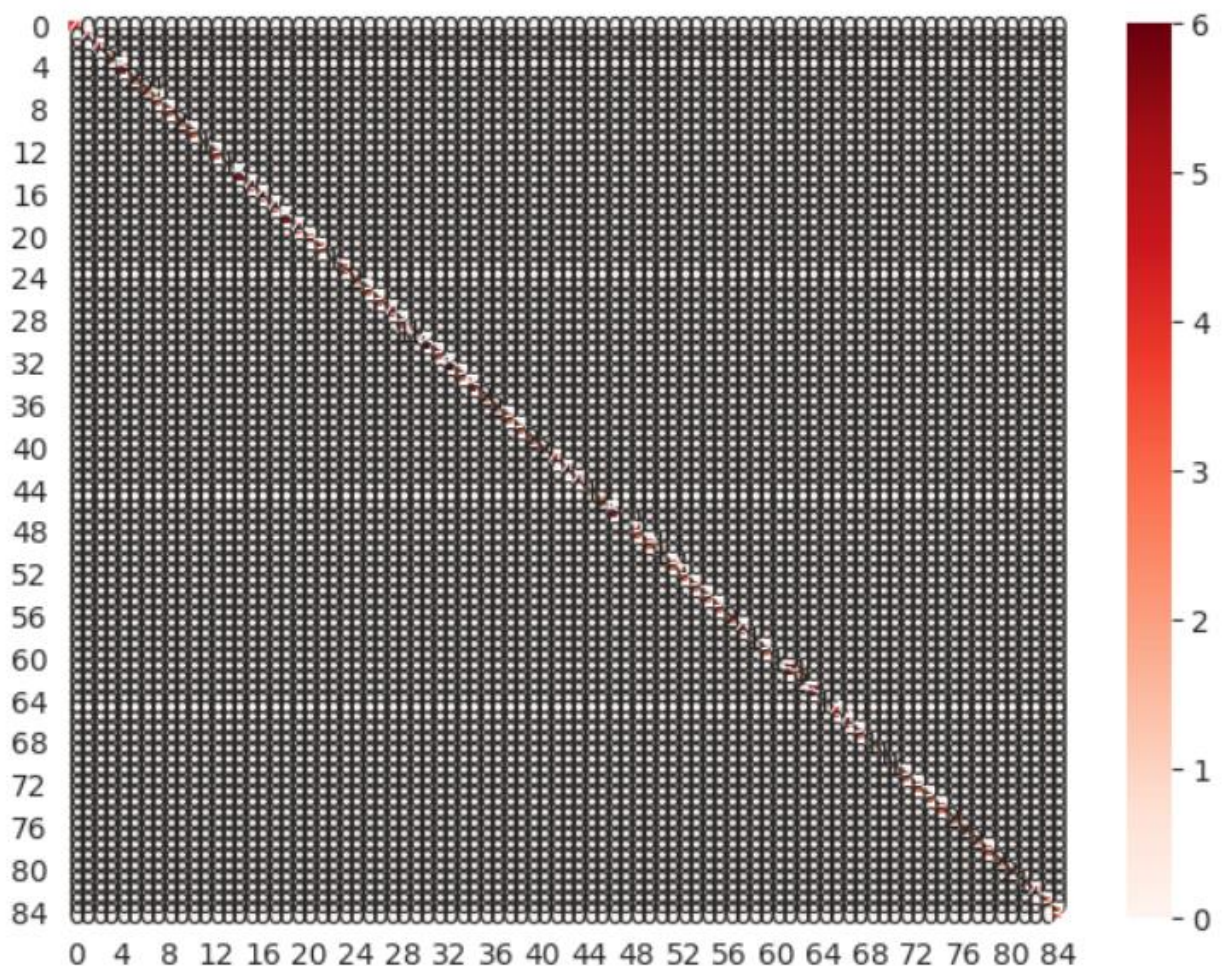


Рисунок 42 - Обучаемость многослойного персептрона. Количество нейронов на скрытых слоях 100 и 50

ЗАКЛЮЧЕНИЕ

Многослойный персептрон является одним из наиболее популярных видов нейронных сетей, который может быть использован в обработке данных физического эксперимента. MLP состоит из нескольких слоев нейронов, где каждый нейрон в слое связан с каждым нейроном в следующем слое.

В контексте физического эксперимента MLP может быть использован для анализа данных, выполнения классификации и прогнозирования. Например, в эксперименте по физике частиц MLP может быть использован для определения свойств элементарных частиц на основе данных, собранных детектором.

MLP может быть обучен на основе исходных данных и затем использован для предсказания результатов эксперимента. Он может помочь исследователям обнаружить скрытые закономерности в данных, которые могут быть недоступны для человека. Кроме того, MLP может быть использован для оптимизации процесса экспериментов и улучшения точности результатов.

Итак, многослойный персептрон применяется в обработке физического эксперимента для анализа данных, классификации, прогнозирования и оптимизации экспериментов. Он является мощным инструментом, который может помочь исследователям извлечь ценную информацию из экспериментальных данных.

В ходе выполнения данной работы было исследовано и проанализировано использование искусственного интеллекта для интеллектуального анализа данных физических экспериментов, проанализированы существующие методы и программы для анализа данных, выявлены их преимущества и недостатки.

В результате выполнения данной выпускной квалификационной работы были достигнуты следующие результаты:

- проанализированы существующие программы, способные выполнять анализ и обработку данных физических экспериментов, сравнены и выявлены сильные и слабые стороны текущих решений;
- разработана полносвязная нейронная сеть для прогнозирования результатов эксперимента;
- произведено обучение разработанной нейронной сети;
- реализовано тестирование и сравнение полученной модели с аналогами;
- показана применимость разработанной нейронной сети.

Определена актуальность данной темы и поставлены цели выполнения для дальнейшего выполнения научно-исследовательской работы. Также был проанализирован описательно-статистический анализ данных, методология CRISP-DM интеллектуального анализа. Был реализован многослойный персептрон для прогнозирования результатов физических экспериментов. Изучены различные виды интеллектуального анализа. Нейронная сеть была протестирована на корректность обработки.

Для обучения был выбран физический эксперимент «движение тела, брошенного под углом к горизонту». На основе простой задачи было показано, что применение нейронных сетей для анализа данных физических экспериментов возможна и может быть использована для анализа более сложных экспериментов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фадеев Р.И., Бирюков Д.В. Использование искусственного интеллекта для анализа сетевого трафика // Современные инновации. 2022. №1 (41). С. 39-42.
2. Семейкин В.Д., Скупченко А.В. Моделирование искусственных нейронных сетей в среде Matlab // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. 2009. №1. С. 159-164.
3. Спиридонова К.С. Существующий программный комплекс первичной обработки и анализа данных // Символ науки. 2021. №6. С. 34-35.
4. Ершова Н.М. Методология планирования и реализация эксперимента в обработке средств данных Excel // Вісник ПДАБА. 2009. №2 (133). С. 7-18.
5. Пономарева К.А. Применение искусственных нейронных сетей при решении задач прогнозирования // Наука без границ. 2020. №1 (41). С. 42-47.
6. Вострокнутов И.Е. Обработка и анализ результатов физического эксперимента средствами современных графических калькуляторов Casio // Вестник Нижегородского университета им. Н. И. Лобачевского. Серия: Социальные науки. 2017. №1 (45). С. 146-149
7. Егорычев А.В., Беляев И.М., Овсянникова Т.А. Распараллеливание в задачах анализа физических данных эксперимента ЛНСб // Программные продукты и системы. 2021. №1. С. 189-194.
8. Шананин В.А., Чаругин В.М., Архипова Е.М., Андрианова А.И. Применение искусственного интеллекта в физическом эксперименте // Инновации и инвестиции. 2022. №4. С. 117-119.
9. Зрелова Д.П., Ульянов С.В. Модели физически информированных / осведомленных классических лагранжевых / гамильтоновых нейронных сетей в глубоком обучении // Современные информационные технологии и ИТ-образование. 2022. №2. С. 310-325.
10. Дядичев, В. В. Задачи и методы интеллектуального анализа данных // Геополитика и экогеодинамика регионов. 2015. – Т. 1 (11), № 3. С. 23-29.

11. Прокопец, Т. Н. Методы интеллектуального анализа данных // Устойчивое развитие экономики: состояние, проблемы, перспективы: Сборник трудов XVI международной научно-практической конференции. В 2-х частях, Пинск, 29 апреля 2022 года / Редколлегия: В.И. Дунай [и др.]. Том Часть I. – Пинск: Полесский государственный университет, 2022. С. 301-304.
12. Выдолоб, Д. Е. Методология разработки проектов data science. CRISP-DM // Тенденции развития науки и образования. 2023. № 99-7. С. 21-24.
13. Дерябин, А. А. Дата-грамотность как новая цифровая компетенция // Информационное общество. 2020. № 5. С. 39-47
14. Макаров Д.А., Шибанова А.Д. СТРУКТУРА НЕЙРОННЫХ СЕТЕЙ // Теория и практика современной науки. 2019. №1 (43). С. 618-621
15. Качков М.С. Создание нейронной сети для решения различных прикладных задач // Известия ТулГУ. Технические науки. 2023. №2. С. 339-343.
16. Панарин В.М., Гришаков К. Вл., Маслова А. А., Гришакова О. В., Архипов А. В. Нейроны в нейронных сетях // Известия ТулГУ. Технические науки. 2023. №2. С. 438-443.
17. Сущеня Р.В., Кокаев А.Э. Нейронные сети и их классификация. Основные виды нейронных сетей // Вестник науки. 2023. №8 (65). С. 186-189.
18. Горбачевская Е. Н. Классификация нейронных сетей // Вестник Волжского университета им. В.Н. Татищева. – 2012. – № 2(19). – С. 128-134.
19. Львов Ф. А. Типы и применение нейронных сетей // Кронос: естественные и технические науки. 2019. №2 (24). С. 25-31.
20. Воронов И.В., Политов Е.А., Ефременко В.М. Обзор типов искусственных нейронных сетей и методов их обучения // Вестник КузГТУ. 2007. №3. С. 38-42.
21. Хоченков А.Е., Галиаскаров Э.Г. Автоматизированная система сбора данных физического эксперимента // Объектные системы. 2014. №1 (8). С. 57-63.
22. Григораш О.В. Методология экспериментальных исследований // Научный журнал КубГАУ. 2017. №127. С. 849-864.

23. Аннаева М., Атанепесова А. Однофакторный дисперсионный анализ: методы и применение в статистике // Всемирный ученый. 2023. №9. С. 267-271.
24. Ершова Е.А. Методы статистического анализа // European research. 2016. №12 (23). С. 27-29.
25. Новиков С. П., Зайцева Е. Ю. Применение корреляционного анализа для исследования экспериментальных данных // Молодой исследователь Дона. 2019. №4 (19). С. 57-64.
26. Харитонов Дарья Евгеньевна Корреляционно-регрессионный анализ в экономике // Контентус. 2016. №8 (49). С. 176-179.

Приложение А

(обязательное)

Интеллектуальный анализ данных физических экспериментов

Листинг программы

Файл main.ipnb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sts
%matplotlib inline
import statistics as st

df=pd.read_csv('/content/experiment.csv')
df.head(10)
df.mean()
v = df['v']
l = df['l']
fig, ax = plt.subplots()
ax.scatter(v, l, c='red', alpha=0.3, edgecolors='none')

ax.grid(True)
plt.xlabel('Начальная скорость')
plt.ylabel('Дальность полета')
plt.show()

a = df['a']
l = df['l']
fig, ax = plt.subplots()
ax.scatter(a, l, c='red', alpha=0.3, edgecolors='none')

ax.grid(True)
plt.xlabel('Угол бросания')
plt.ylabel('Дальность полета')
plt.show()

print('Медиана: ', st.median(v))
print('Медиана: ', st.median_low(v))
print('Медиана: ', st.median_high(v))

print('Медиана: ', st.median(l))
print('Медиана: ', st.median_low(l))
print('Медиана: ', st.median_high(l))

print('Медиана: ', st.median(a))
print('Медиана: ', st.median_low(a))
print('Медиана: ', st.median_high(a))
df.median()

df.mode()

print('Дисперсия:')
print('Начальная скорость', st.pvariance(v))
print('Угол бросания', st.pvariance(a))
```

```

print('Дальность полета', st.pvariance(l))

print('Стандартное отклонение:')
print('Начальная скорость', st.pstdev(v))
print('Угол бросания', st.pstdev(a))
print('Дальность полета', st.pstdev(l))

x = [i for i in range(0, len(v))]
y = df.loc[:, 'v']

plt.xlabel('номер')
plt.ylabel('начальная скорость')
plt.title('Точечный график зависимости двух выборок x и y')
plt.grid(True)
plt.scatter(x, y, marker='.')
plt.show()

x = [i for i in range(0, len(a))]
y = df.loc[:, 'a']

plt.xlabel('номер')
plt.ylabel('угол бросания')
plt.title('Точечный график зависимости двух выборок x и y')
plt.grid(True)
plt.scatter(x, y, marker='.')
plt.show()

df.loc[df['v'] > 76, :]

df.describe()

v = df['v']
l = df['l']
plt.xlabel('выборка v')
plt.ylabel('выборка l')
plt.title('Точечный график зависимости двух выборок x и y')
plt.grid(True)
plt.scatter(v, l, marker='.')

df = df.loc[df['v']!=405, :]

v = df['v']
l = df['l']
a = df['a']
plt.xlabel('выборка v')
plt.ylabel('выборка l')
plt.title('Точечный график зависимости двух выборок x и y')
plt.grid(True)
plt.scatter(v, l, marker='.')

from scipy.stats import pearsonr, spearmanr, kendalltau, norm

```

```

x = df['v']
y = df['l']
corr, p_value = pearsonr(x, y)
print('Коэффициент корреляции Пирсона:', corr)
cor, p_value = spearmanr(v, l)
print('Коэффициент корреляции Спирмена', cor)

corr, p_value = pearsonr(l, a)
print(corr)

cor, p_value = spearmanr(v, l)
print(cor)

cor, p_value = kendalltau(v, l)
print(cor, p_value)

import seaborn as sns
sns.heatmap(df.corr(), annot=True, cmap='RdYlGn', linewidths=0.2)
#data.corr()-->correlation matrix
fig=plt.gcf()
fig.set_size_inches(12,12)
plt.show()

plt.scatter(x, y)

slope, intercept = np.polyfit(x, y, deg = 1)

x_vals = np.linspace(0, 80, num = 1000)
y_vals = intercept + slope * x_vals
plt.plot(x_vals, y_vals, 'r')

plt.show()

plt.boxplot(x=df['v'], vert=False, showmeans=True, meanline=True,
labels=('x'), patch_artist=True, medianprops={'linewidth': 2,
'color': 'purple'}, meanprops={'linewidth': 2, 'color': 'red'})

df_del = df.drop(np.where(df['v'] >= 35)[0])
plt.boxplot(x=df_del['v'], vert=False, showmeans=True,
meanline=True, labels=('x'), patch_artist=True,
medianprops={'linewidth': 2, 'color': 'purple'},
meanprops={'linewidth': 2, 'color': 'red'})

```

Файл MLP.ipynb

```

import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import torch
import torch.nn as nn

```

```

import numpy as np
import math

class MLPtorch(nn.Module):
    def __init__(self, in_size, hidden_size,
second_hidden_size,out_size):
        nn.Module.__init__(self)
        # nn.Sequential - контейнер модулей
        # он последовательно объединяет и позволяет запускать их
одновременно
        self.layers = nn.Sequential(nn.Linear(in_size,
hidden_size), # слой линейных сумматоров
nn.ReLU(),
# функция активации
nn.Linear(hidden_size,
second_hidden_size),
nn.ReLU(),
nn.Linear(second_hidden_size,
out_size),
nn.ReLU(),
        )

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

def train(x, y, num_iter):
    for i in range(0,num_iter):
        pred = net.forward(x)
        loss = lossFn(pred, y)
        loss.backward()
        optimizer.step()
        if i%(num_iter/10)==0:
            print('Ошибка на ' + str(i) + ' итерации: ',
loss.item())
    return loss.item()

df=pd.read_csv('/content/experiment.csv')
df.head(10)

v = df['v']
l = df['l']
a = df['a']

X = np.stack((v, a), axis=-1)
y = l
from sklearn.model_selection import train_test_split

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3)

inputSize = X.shape[1] # количество входных сигналов равно
количеству признаков задачи
hiddenSizes = 150 # задаем число нейронов скрытого слоя
second_hidden_size = 100
outputSize = 1 # количество выходных сигналов равно количеству
классов задачи

net =
MLPptorch(inputSize,hiddenSizes,second_hidden_size,outputSize)
lossFn = nn.MSELoss()

optimizer = torch.optim.SGD(net.parameters(), lr=0.00001)

loss_ = train(torch.Tensor(X_train),
               torch.Tensor(np.transpose(y_train)), 5000)

mlp.fit(X_train,y_train)
mlp_pred = mlp.predict(X_test)
print('Неверно классифицированные образцы: %d' % (y_test !=
mlp_pred).sum())
print('Точность: %.2f' % accuracy_score(y_test, mlp_pred))

# матрица неточности

cnf_matrix_log = confusion_matrix(y_test, mlp_pred)

sns.heatmap(pd.DataFrame(cnf_matrix_log), annot=True, cmap="Reds" ,
fmt='g')
plt.tight_layout()
plt.title('Матрица неточности Персептрон\n', y=1.1)

```

Приложение Б

(обязательное)

Интеллектуальная обработка данных физических экспериментов

Презентация

Министерство науки и высшего образования РФ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет вычислительной техники
Кафедра «Информатика и вычислительная техника»

Выпускная квалификационная работа
на тему

ИНТЕЛЛЕКТУАЛЬНАЯ ОБРАБОТКА ДАННЫХ ФИЗИЧЕСКИХ ЭКСПЕРИМЕНТОВ

Смирнова Алина Дмитриевна

НАУЧНЫЙ РУКОВОДИТЕЛЬ:

доктор технических наук,
профессор Зинкин С.А.

Пенза 2024

Цель магистерской диссертации заключается в исследовании нейросетевых алгоритмов и методов анализа данных физических экспериментов, разработка нейронной сети для прогнозирования результатов эксперимента.

Задачи магистерской диссертации:

- обзор и анализ существующих исследований и алгоритмов, применяемых и направленных на анализ данных физических экспериментов;
- разработка нейросетевой модели для прогнозирования результатов физических экспериментов;
- обучение модели для прогнозирования результатов экспериментов;
- тестирование и сравнение полученной модели с аналогами.

Тело, брошенное под углом α к горизонту с начальной скоростью v_0 , без учета сопротивления воздуха будет лететь по параболе и через некоторое время достигнет земли.

Полный путь тело пройдет за время 2τ , двигаясь равномерно вдоль оси x со скоростью $v_x^{(0)}$:

$$l = 2\tau v_x^{(0)} = v_0 \cos \alpha \cdot 2 \frac{v_0 \sin \alpha}{g} = \frac{v_0^2 \sin 2\alpha}{g} \quad (1)$$

Многослойный персептрон (MLP) - это название современной искусственной нейронной сети с прямой связью, состоящей из полностью связанных нейронов с нелинейным типом функции активации, организованной по меньшей мере в три слоя, примечательной способностью различать данные, которые нельзя разделить линейным образом.

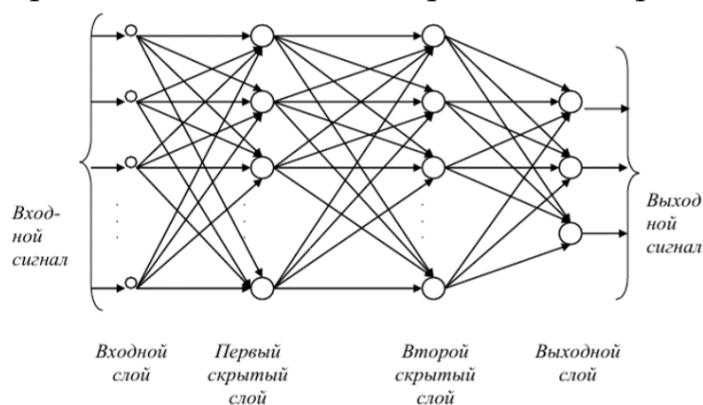


Рисунок 1 – Схема многослойного персептрона с двумя скрытыми слоями.

Описательно-статистический анализ выявляет закономерные характеристики изменчивости отдельных признаков. Три ключевыми задачами описательной статистики являются:

- 1. оценка центральной тенденции распределения;
- 2. оценка отклонения от центра распределения;
- 3. анализ закона распределения переменной (только для параметрических переменных).

	v	a	l
0	10	50	20.098117
1	16	1	1.823321
2	19	88	5.139201
3	26	80	47.184820
4	5	3	0.533308
5	12	4	4.089985
6	34	30	204.311299
7	14	27	32.360680
8	19	45	73.673469
9	29	7	41.521697

Рисунок 2 – Набор данных.

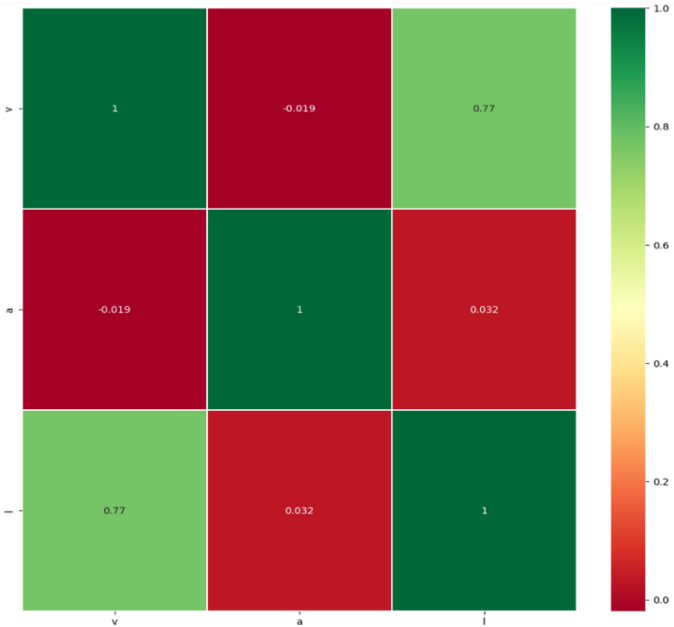


Рисунок 3 – Матрица корреляции

Пример библиотек языка программирования Python для анализа данных:

1. **Python statistics** — встроенная библиотека Python для описательной статистики. Используется, если наборы данных не слишком велики или если нельзя полагаться на импорт других библиотек.
2. **NumPy** — сторонняя библиотека для численных вычислений, оптимизированная для работы с одномерными и многомерными массивами. Его основным типом является тип массива называется ndarray. Эта библиотека содержит множество подпрограмм для статистического анализа.
3. **SciPy** — сторонняя библиотека для научных вычислений, основанная на NumPy. Она предлагает дополнительную функциональность по сравнению с NumPy, в том числе scipy.stats для статистического анализа.
4. **Pandas** — сторонняя библиотека для численных вычислений, основанная на NumPy. Она отлично справляется с обработкой помеченных одномерных (1D) данных с объектами Series и двумерных (2D) данными объектами DataFrame.
5. **Matplotlib** — сторонняя библиотека для визуализации данных.

```
class MLPtorch(nn.Module):
    def __init__(self, in_size, hidden_size, second_hidden_size, out_size):
        nn.Module.__init__(self)
        self.layers = nn.Sequential(nn.Linear(in_size, hidden_size),
                                     nn.ReLU(),
                                     nn.Linear(hidden_size, second_hidden_size),
                                     nn.ReLU(),
                                     nn.Linear(second_hidden_size, out_size),
                                     nn.ReLU(),
                                     )

    # прямой проход
    def forward(self, x):
        return self.layers(x)
```

Рисунок 4 - Программный код класса MLPtorch

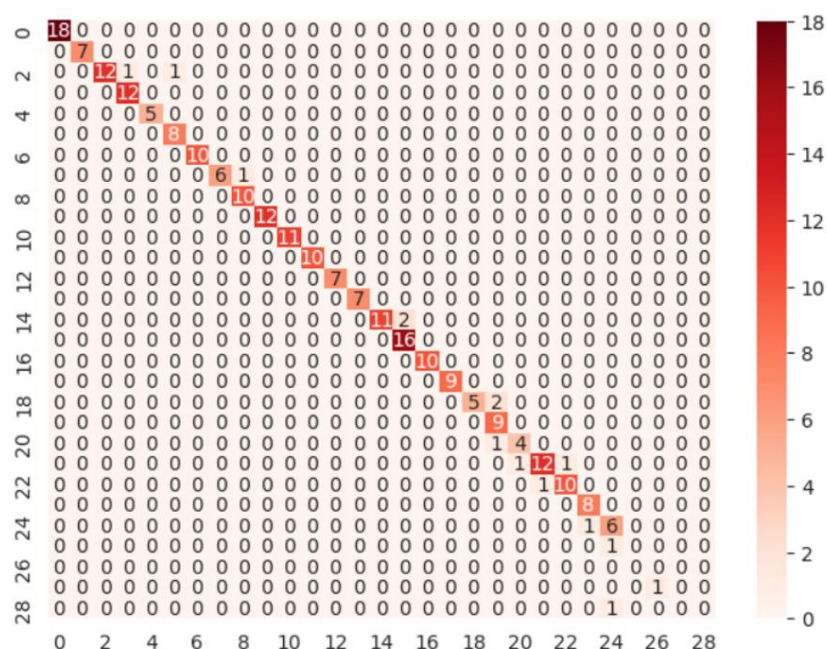


Рисунок 5 – Матрица неточности многослойного персептрона.

Основные результаты и выводы

- 1) Проанализированы существующие нейронные сети, способные осуществлять анализ данных физических экспериментов.
- 2) Разработана нейросетевая модель, способная обрабатывать физические данные и прогнозировать результаты эксперимента.
- 3) Реализовано тестирование и сравнение полученной модели с аналогами.
- 4) Показана применимость разработанной нейросетевой модели для интеллектуального анализа данных физического эксперимента.