Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод обнаружения сетевых атак с использованием многослойной нейронной сети»

Студент ИУ7-83Б _____ А. В. Криков
(Группа) (Подпись, дата) (И. О. Фамилия)

Руководитель ВКР _____ П. В. Клорикьян
(Подпись, дата) (И. О. Фамилия)

Нормоконтролер _____ _____
(Подпись, дата) (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Рассчетно-пояснительная записка содержит 61 страниц, 24 иллюстраций, 3 таблиц, 14 источников, 1 приложение.

Ключевые слова: сетевая атака, нейронная сеть, машинное обучение.

В работе представлена разработка метода обнаружения сетевых атак с использованием многослойной нейронной сети.

Рассмотрена задача обнаружения сетевых атак. Рассмотрены методы обнаружения сетевых атак. Рассмотрены существующие архитектуры нейронных сетей. Представлена реализация метода обнаружения сетевых атак, проведены исследования эффективности разработанного метода.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

MLP (от англ. Multilayer perceptron) — многослойный персептрон.

CNN (от англ. Convolutional neural network) — сверточная нейронная сеть.

RNN (от англ. Recurrent neural network) — рекуррентная нейронная сеть.

DDoS (от англ. Distributed Denial of Service) — распределённый отказ в обслуживании.

XSS (от англ. Cross-site scripting) — межсайтовый скриптинг.

PortScan (от англ. Port Scanning) — сканирование портов.

ВВЕДЕНИЕ

Быстрый рост информационных технологий вызывает ряд проблем, связанных с защитой сетевых ресурсов в глобальной сети. Согласно исследованиям, с начала 2022 года количество сетевых атак увеличилось на 15% по сравнению с 2021 годом, а доля массовых атак составляет 33% от всего числа. Также вырос интерес к интернет-ресурсам: доля атак на них увеличилась до 22% от общего числа в сравнении с наблюдаемыми 13% в 2021 году [1].

На основании данного исследования можно сделать вывод, что количество сетевых атак лишь растет, а следовательно, растет и потребность в защите от них.

Цель работы — разработать метод обнаружения сетевых атак с использованием многослойной нейронной сети.

Для достижения поставленной цели необходимо решить следующие задачи:

— описать термины предметной области и обозначить проблему;

— описать технологии, с помощью которых можно реализовать метод обнаружения сетевых атак;

— разработать метод обнаружения сетевых атак;

— разработать программный комплекс, реализующий интерфейс для взаимодействия с разработанным методом;

— исследовать разработанный метод на применимость.

1 Аналитический раздел

В данном разделе приводится классификация сетевых атак. Рассматривается технология нейронных сетей и принцип работы нейронных сетей. Описываются архитектуры нейронных сетей, их применение в задачах классификации. Проводится обзор существующих методов обнаружения сетевых атак.

1.1 Анализ предметной области

Сетевая атака [2] — это действие или последовательность связанных между собой действий, использующих уязвимости информационной системы и приводящих к нарушению политики безопасности. Под политикой безопасности подразумевается набор критериев и правил, описывающих информационные процессы в системе, выполнение которых обеспечивает необходимое условие безопасности системы.

1.1.1 Модели атак

Классическая модель атаки выстраивается по принципу «один к одному», как показано на рисунке 1.1, или «один ко многим», как показано на рисунке 1.2. Для защиты от таких атак разработчики внедряют сенсоры системы защиты, передающие информацию на центральный аппарат управления. Благодаря этому обеспечивается масштабируемость системы и простота удаленного управления. Однако такое решение не распространяется на модель с распределенными атаками [3].

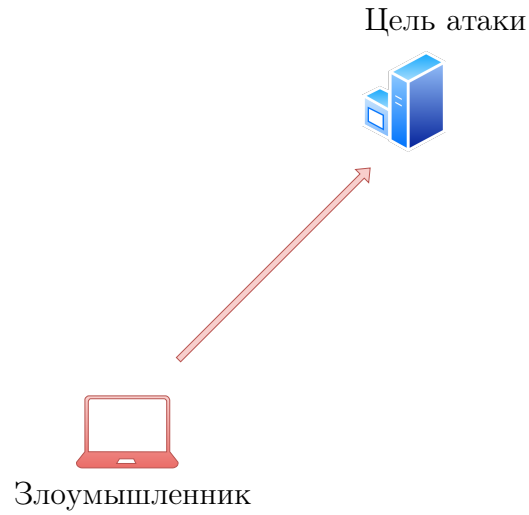

Рисунок 1.1 – Сетевая атака «один к одному»

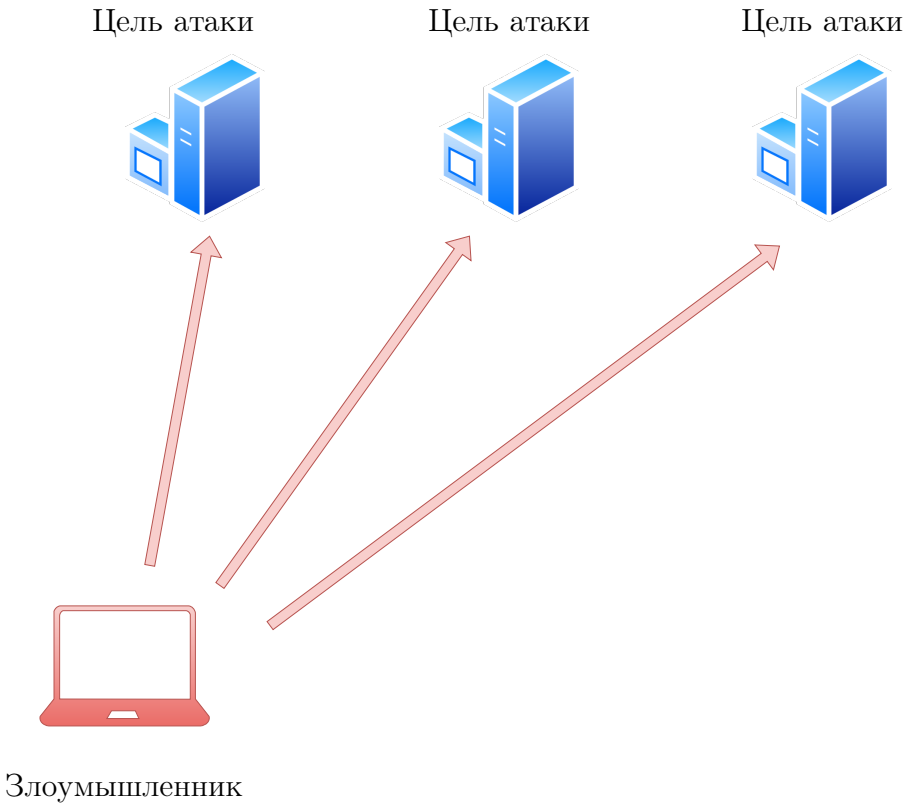

Рисунок 1.2 – Сетевая атака «один ко многим»

В **распределенной модели** используется принцип «многие к одному», как показано на рисунке 1.3, или «многие ко многим», как показано на рисунке 1.4. Данная атака осуществляется в два этапа. На первом этапе ищутся узлы сети, которые впоследствии задействуются для реализации распределенной атаки. Второй этап представляет собой посылку большого количества запросов на атакуемый сетевой узел. Отправка запросов осуществляется с помощью скомпрометированных систем-посредников, на которых установлены специальные агенты, реализующие распределенную атаку. Агенты делятся на два типа: «мастер» и «демон». Злоумышленник управляет небольшим числом «мастеров», а те управляют «демонами». Стоит отметить, что блокирование одного или нескольких «мастеров» или «демонов» не приводит к завершению атаки [3].

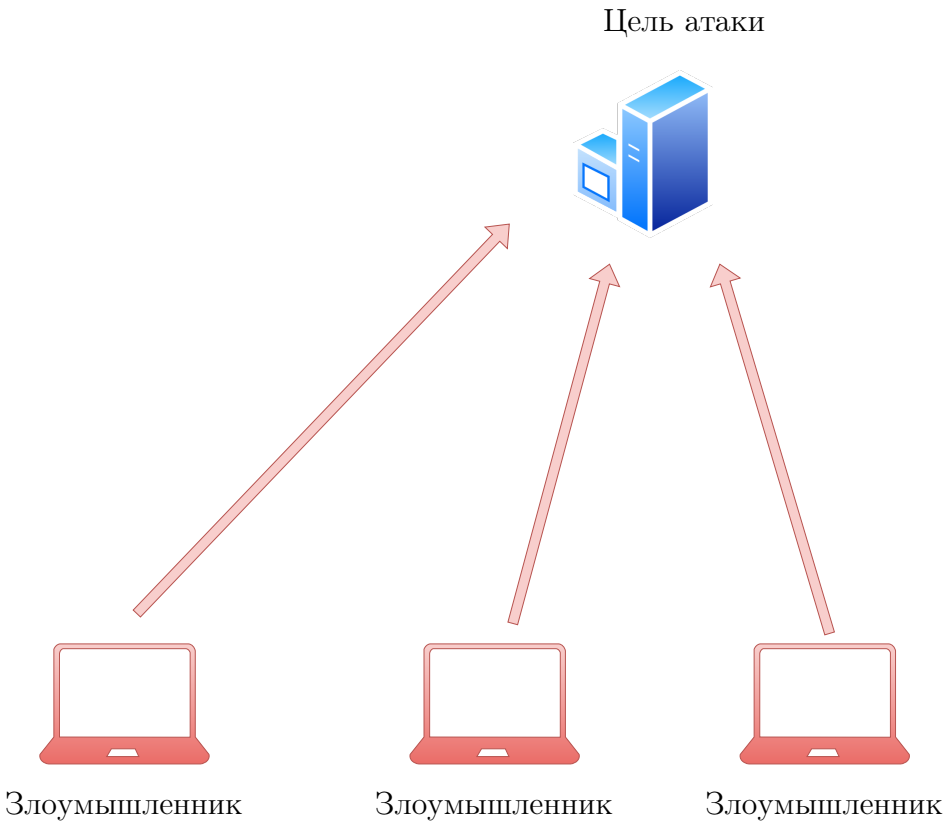

Рисунок 1.3 – Сетевая атака «многие к одному»

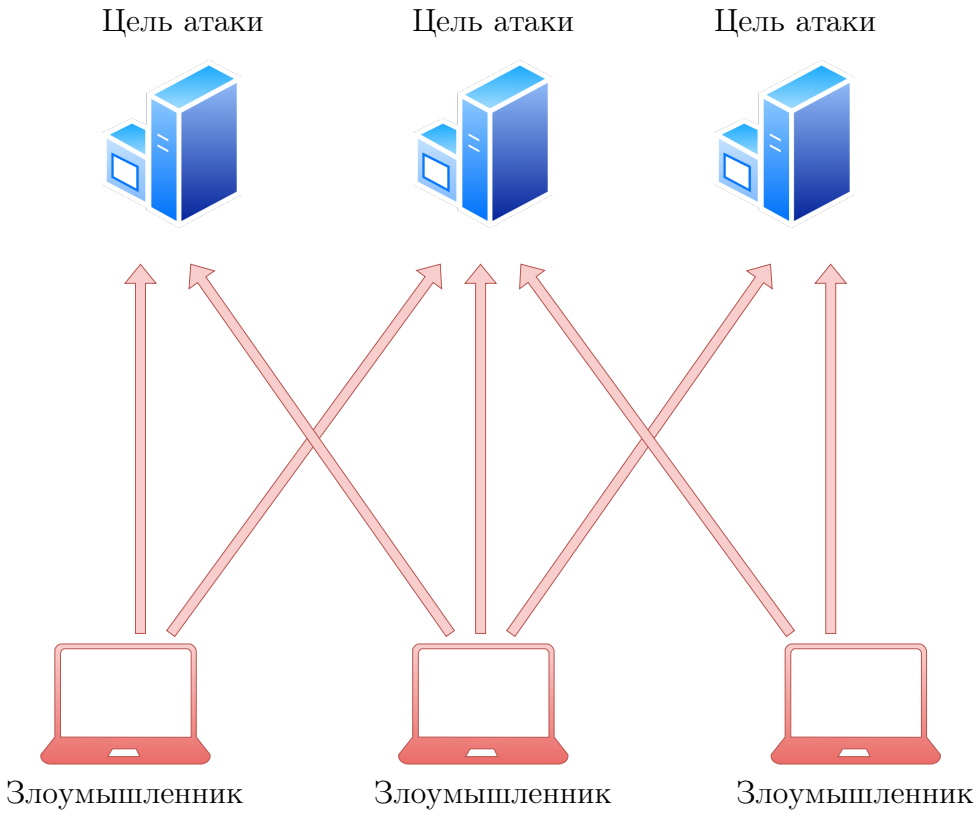

Рисунок 1.4 – Сетевая атака «многие ко многим»

1.1.2 Классификация сетевых атак

Большинство сетевых атак нацелены на изменение определенных параметров безопасности системы. Например, с помощью некоторых атак злоумышленник может получить возможность просматривать передаваемые сообщения, но не изменять их. Другие атаки могут позволить злоумышленнику выполнить останов некоторых компонент системы, при этом не предоставляя доступ к ресурсам, хранящимся в данной системе [3].

Существует множество различных типов классификации атак. Например, деление на внешние и внутренние, пассивные и активные, умышленные и неумышленные. Однако, все сетевые атаки можно разделить на два класса: пассивные и активные [3].

Пассивная атака [4] — это атака, при которой у злоумышленника нет доступа к модификации передаваемых сообщений и возможности добавления собственных сообщений в информационный канал между отправителем и получателем. Основная цель пассивной атаки — прослушивание передаваемых сообщений и анализ сетевого трафика.

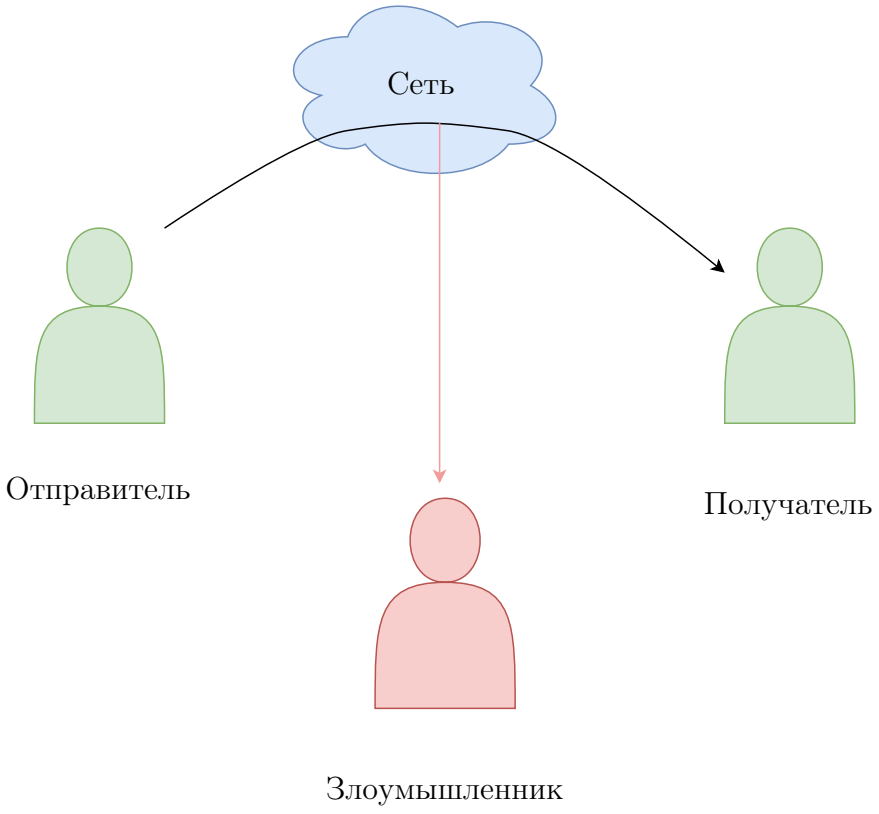

Рисунок 1.5 – Пассивная атака

Одной из разновидностей пассивных атак являются **атаки сканирова-**

ния [5]. Атаки сканирования не нацелены на проникновение в систему. Они помогают злоумышленнику определять:

— топологию сети;

— активные хосты в сети;

— выполняемое на хостах ПО сервера;

— номера версий обнаруженного ПО.

Наиболее популярной атакой сканирования является **PortScan-атака (от англ. Port Scanning)**. Сканируя порты, злоумышленник пытается определить следующие параметры системы: открытые порты, базовую операционную систему, поддерживаются ли неаутентифицированные входы в систему и т.д. Это выясняется путем отправки пакетов с флагами на различные порты целевой системы, чтобы выяснить, является ли порт уязвимым и может ли он быть использован для проникновения в систему.

Активная атака [6] — это атака, при которой у злоумышленника имеется возможность модифицировать передаваемые сообщения и добавлять собственные. Существуют следующие виды активных атак:

1. **Отказ в обслуживании — DoS-атака (англ. Denial of Service)** [5]. Отказ в обслуживании нарушает функционирование сетевых сервисов. Суть данной атаки заключается в следующем: на сетевой сервис поступает значительное количество запросов, в результате чего сетевой сервис перестает обрабатывать запросы реальных клиентов, а злоумышленник может перехватывать все сообщения направленные определенному адресату. DoS-атака базируется на классической модели атак.

2. **Распределенный отказ в обслуживании — DDoS-атака (от англ. Distributed Denial of Service)** [5]. Основным отличием DDoS-атаки от DoS-атаки является использование распределенной модели атак.

3. **DDoS Hulk (от англ. Distributed Denial of Service HTTP Unbearable Load King)** DDoS Hulk атака — это тип DDoS-атак, который основывается на использовании HTTP протокола, для усиления и ускорения атаки. Нападающий использует несколько ботнетов для

отправки огромного количества запросов на сервер, с целью перегрузить его и сделать недоступным для легитимных пользователей.

4. **DDoS Heartbleed** атака является сочетанием двух типов кибератак: DDoS-атака и атака на уязвимость Heartbleed в OpenSSL-шифровании. DDoS-атака направлена на перегрузку сети огромным количеством запросов, что делает сервис недоступным для легитимных пользователей. В то же время, атака на уязвимость Heartbleed позволяет злоумышленникам получить доступ к чувствительным данным, хранящимся в памяти сервера, в том числе логинам, паролям, кредитным картам и другим конфиденциальным данным. Комбинированное использование этих двух методов позволяет киберпреступникам делать не только сервис недоступным, но и кражу конфиденциальных данных, что является очень опасным для пользователей и компаний.

5. **DDoS Slowloris** атака — это тип DDoS атаки, который основывается на отправке множества неполных запросов на сервер с помощью открытых соединений HTTP. Неполные запросы не закрываются, поэтому соединение остается открытым и приводит к выделению большого количества ресурсов сервера. Это приводит к замедлению, а в итоге и к отказу в обслуживании сервера или веб-сайта.

6. **Атака полным перебором — Brute Force** [7]. Brute Force — это атака с угадыванием паролей, учетных данных для входа в систему, ключей шифрования и прочей информации. Основная цель атаки полным перебором — получить несанкционированный доступ к данным, системам или сетям.

7. **Межсайтовый скриптинг — XSS (от англ. Cross-Site Scripting)** [8]. XSS — тип атаки на веб-системы, заключающийся во внедрении в выдаваемую веб-системой страницу вредоносного кода (который будет выполнен на компьютере пользователя при открытии им этой страницы) и взаимодействии этого кода с веб-сервером злоумышленника. Специфика подобных атак заключается в том, что вредоносный код может использовать авторизацию пользователя в веб-системе для получения к ней расширенного доступа или для получения авторизационных данных

пользователя. Вредоносный код может быть вставлен в страницу как через уязвимость в веб-сервере, так и через уязвимость на компьютере пользователя.

8. **Внедрение SQL-кода — SQLi (от англ. SQL injection)** [8]. SQLi — один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода. Внедрение SQL, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.

1.2 Классификация методов обнаружения сетевых атак

Распознавание и реагирование на подозрительную деятельность, нацеленную на нарушение сетевых или вычислительных ресурсов организации, называется обнаружением сетевых атак. Эффективность этого процесса сильно зависит от применяемых методов анализа. На сегодняшний день основными методами обнаружения сетевых атак являются методы статистического анализа, методы на основе знаний и методы машинного обучения.

1.2.1 Методы статистического анализа

Для защиты системы используется подход, при котором для каждого субъекта определяется профиль, а любое отклонение от него считается несанкционированной деятельностью. Статистические методы универсальны, так как не требуют знания о возможных атаках на систему. Однако при их использовании могут возникнуть трудности, например, отсутствие возможности определить новые типы атак, которые не были включены в выборку данных, на основе которой обучается модель.

1.2.2 Методы на основе знаний

Данные методы широко используются и заключаются в формулировании информации об атаках в виде правил или сигнатур. Сигнатуры являются шаблонами, описывающими определенную атаку. Если происходящие действия совпадают с одной из сигнатур или срабатывает одно из правил, то такие действия считаются несанкционированными. Такие методы характеризуются практически полным отсутствием ложных тревог, однако для поддержания их актуальности необходимо постоянно обновлять базы данных атак. Методы не является устойчивым к модификациям атак и не могут автоматически адаптироваться.

1.2.3 Методы машинного обучения

Машинное обучение является эффективной технологией для обнаружения сетевых атак. Данная технология использует алгоритмы и методы обработки большого количества данных и создания моделей, которые могут определять наличие аномальной активности в сети.

Для обнаружения сетевых атак с помощью машинного обучения необходимо собрать большой объем данных о поведении сети. Данные включают в себя информацию о трафике, времени и местоположении источника трафика и других факторах. Эта информация затем анализируется с помощью методов машинного обучения, которые могут отслеживать паттерны поведения и обнаруживать аномальные события.

Основными плюсами использования машинного обучения в задачах обнаружения сетевых атак являются:

— гибкость. Методы машинного обучения позволяют анализировать данные сетевого трафика в условиях искажения или неполноты;

— быстрая скорость обработки информации. Скорость обработки информации с помощью машинного обучения достаточна для реагирования в режиме реального времени на проводимые атаки до того, как в системе появятся непоправимые повреждения;

— возможность прогнозирования. Выходные данные в методах машинного обучения могут быть интерпретированы в форме вероятности. Это

предоставляет возможность прогнозирования дальнейшего развития атаки.

1.3 Нейронные сети

Направление исследований в области искусственного интеллекта, называемое нейронными сетями, основано на том, чтобы воссоздать работу нервной системы человека. Одним из ключевых аспектов является способность обучаться и исправлять ошибки, что в конечном итоге позволяет имитировать работу человеческого мозга, хотя и достаточно грубо [9].

Нейронные сети состоят из слоев узлов: входных данных, скрытых и выходных данных. У каждого узла есть веса и пороговое значение. Если вывод узла превышает порог, он активируется и пересылает данные на следующий уровень. Если нет, данные не передаются.

На рисунке 1.6 представлена модель работы нейронной сети, где x_i — входные данные, а Y_j — выходные данные.

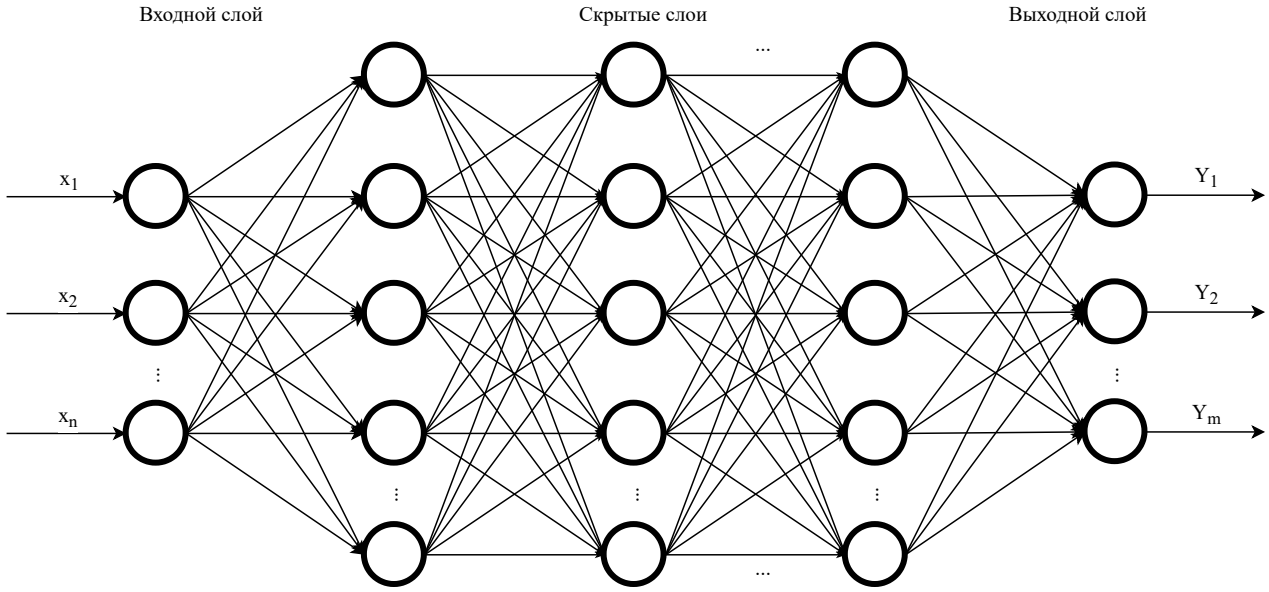

Рисунок 1.6 – Модель нейронной сети

1.3.1 Принцип работы нейронных сетей

Модель каждого отдельного узла нейронной сети может быть описана как модель линейной регрессии [10] состоящая из входных данных, весовых коэффициентов, смещения (или порогового значения) и выходных данных. Эту модель можно описать следующей формулой:

$$\hat{y} = \sum_{i=1}^{m} w_i x_i + bias, \tag{1.1}$$

где $\hat{y}$ —прогнозируемое значение модели;

m —количество факторов;

w —весовой коэффициент фактора;

x —единица входных данных;

$bias$ —значение ошибки.

Выходное значение для узла описывается по формуле:

$$f(x_i) = \begin{cases} 1 & \text{если } \hat{y} \geq 0 \\ 0 & \text{иначе.} \end{cases} \tag{1.2}$$

Необходимо назначить весовые коэффициенты после определения слоя входных данных. Они определяют важность каждого фактора: чем выше коэффициент, тем больший вклад данный фактор вносит в выходные данные. Затем произведения входных данных и соответствующих им весовых коэффициентов суммируются. Выходные данные передаются через функцию активации 1.2, которая вычисляет результат. Если полученный результат превышает установленное пороговое значение, узел срабатывает (активируется), передавая данные на следующий слой сети.

Для оценки точности модели необходимо использовать функцию стоимости (среднеквадратическая ошибка). Эта функция рассчитывается по формуле:

$$CostFunction = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2, \tag{1.3}$$

где i —индекс выборки;

$\hat{y}$ —прогнозируемое значение;

y —фактическое значение;

m —число выборок.

Конечная цель — минимизировать функцию стоимости, чтобы обеспечить корректность для каждого отдельно взятого наблюдения. Для достижения точки сходимости или локального минимума в модели используется

функция стоимости и обучение с подкреплением, которые корректируют весовые коэффициенты и смещение. Алгоритм градиентного спуска позволяет определить стратегию уменьшения количества ошибок или минимизации функции стоимости. Каждый шаг обучения корректирует параметры модели, пока не будет достигнут минимум. Рисунок 1.7 показывает процесс минимизации функции стоимости.

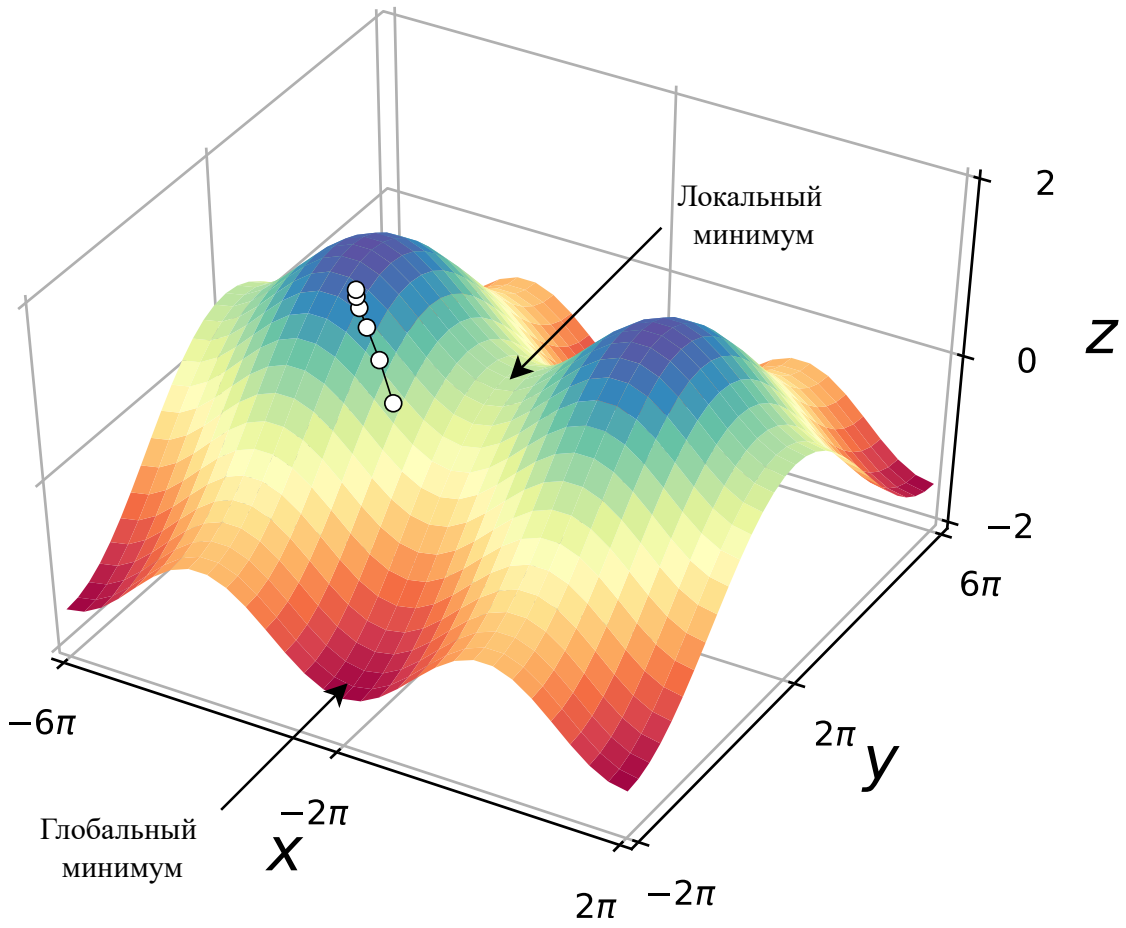

Рисунок 1.7 – Минимизация функции стоимости

1.3.2 Архитектуры нейронных сетей

Многослойная нейронная сеть (от англ. multilayer perceptron, MLP) — это архитектура нейронной сети, состоящая из нескольких слоев нейронов, каждый из которых полностью связан со следующим слоем. Нейроны в каждом слое получают на вход значения активации от всех нейронов предыдущего слоя, производят линейную комбинацию этих значений с определенными весами и применяют к результату функцию активации для получения своего выходного значения. Эта операция выполняется для каждого нейрона в

каждом слое. Такие сети обычно используются для задач классификации и регрессии, где входные данные имеют множество признаков.

Сверточная нейронная сеть (от англ. Convolutional Neural Network, CNN) используется для обработки и классификации изображений, видео и других пространственных данных. CNN состоит из нескольких слоев: сверточных, объединяющих, полносвязных и слоя выхода. Сверточные слои обрабатывают входные данные с помощью взвешивания и объединения признаков из различных частей изображения. Объединяющие слои уменьшают размерность данных, снижая число параметров и значительно ускоряя вычисления. Полносвязные слои связывают все объединенные признаки в выходной слой. CNN активно применяется в многих областях, включая компьютерное зрение, робототехнику, автоматическое управление транспортом, медицину и другие.

Рекуррентная нейронная сеть, (от англ. Recurrent neural network, RNN) — используется для обработки последовательностей данных, например, временных рядов, текстов и аудиофайлов. RNN имеет входной слой, скрытый слой и выходной слой. Скрытый слой в RNN имеет рекуррентные связи, которые позволяют сохранять информацию о предыдущих состояниях и использовать ее в следующих шагах. Это позволяет RNN обрабатывать переменную длину последовательности данных и изучать контекст и зависимости между элементами последовательноси

1.4 Постановка задачи

Формально задача обнаружения сетевых атак с использованием многослойной нейронной сети может быть описана с помощью IDEF0-диаграммы, представленной на рисунке 1.8.

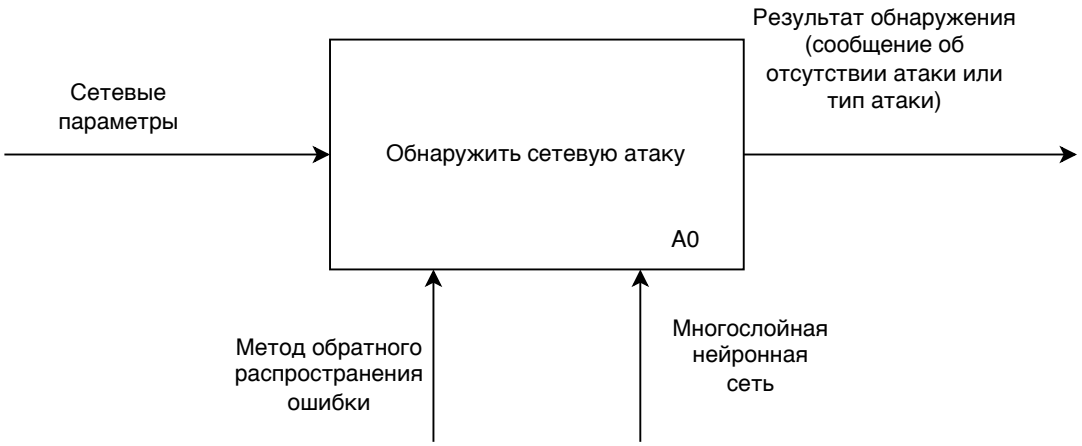

Рисунок 1.8 – Постановка заадчи

1.5 Применение машинного обучения в качестве классификатора сетевых атак

Задача обнаружения сетевых атак сводится к задаче классификации сетевых атак. Процесс создания системы классификации сетевых атак схож с процессами создания других систем с применением машинного обучения:

1. Необходимо собрать набор данных для обучения классификатора. Обучение модели нейронной сети для системы обнаружения сетевых атак будет производиться на наборе данных CICIDS2017.

2. Каждую запись из обучающей коллекции нужно представить в виде вектора признаков.

3. Для каждой записи нужно указать «правильный ответ», т.е. тип сетевой атаки, по этим ответам и будет обучаться классификатор.

4. Выбрать алгоритм классификации и обучить классификатор.

5. Использовать полученную модель для обнаружения сетевых атак.

1.5.1 Метод опорных векторов

SVM (от англ. Support vector machine) — метод машинного обучения, применяемый в задачах классификации, основанный на построении гиперплоскости и ее анализе [11].

Основной целью SVM как классификатора является поиск уравнения разделяющей гиперплоскости $w_1x_1+w_2x_2+\ldots+w_nx_n+w_0=0$ в пространстве R^n, которая бы разделила два класса оптимальным образом. В формуле 1.4 представлен общий вид преобразования F объекта x в метку класса Y:

$$F(x) = sign(w^T x - b). \tag{1.4}$$

При рассмотрении обозначим $w = (w_1, w_2, \ldots, w_n), b = -w_0$. Один из классов будет предсказываться для всех объектов, расположенных на одной стороне гиперплоскости, в то время как для объектов, находящихся на другой стороне, будет предсказываться другой класс.

В SVM веса w и b выбираются таким образом, чтобы объекты классов лежали как можно дальше от разделяющей гиперплоскости. Иначе говоря, алгоритм максимизирует зазор (англ. margin) между гиперплоскостью и объектами классов, которые расположены ближе всего к ней. Такие объекты называются опорными векторами.

На рисунке 1.9 изображен принцип работы метода.

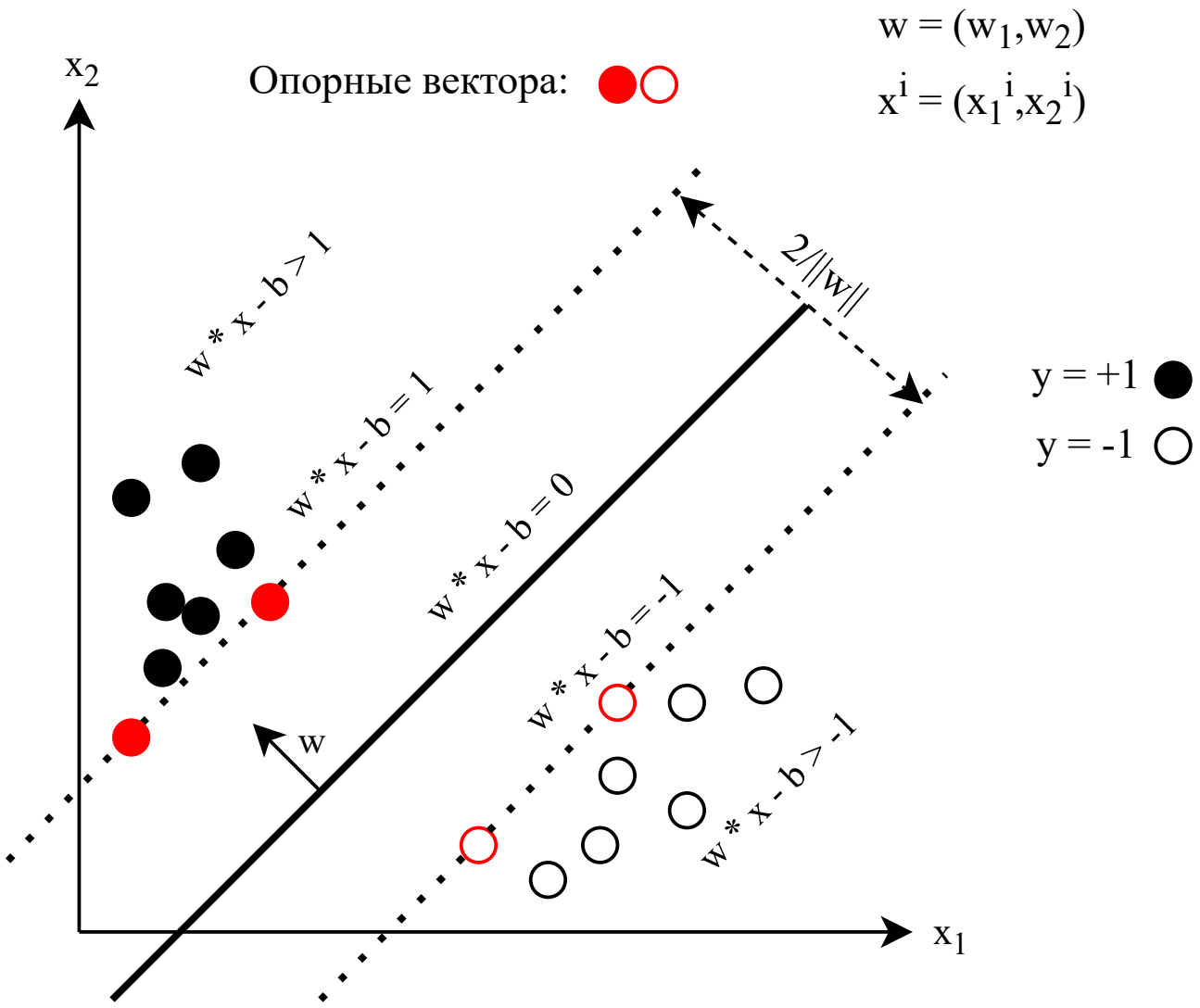

Рисунок 1.9 – Визуализация работы SVM

Здесь и далее скалярное произведение двух векторов будет обозначаться как $\langle a, b\rangle$ или $a^T b$. Ширину разделяющий полосы будет показывать проекция вектора w, концами которого являются опорные вектора разных классов.

На рисунке 1.10 представлен вывод правил настройки весов.

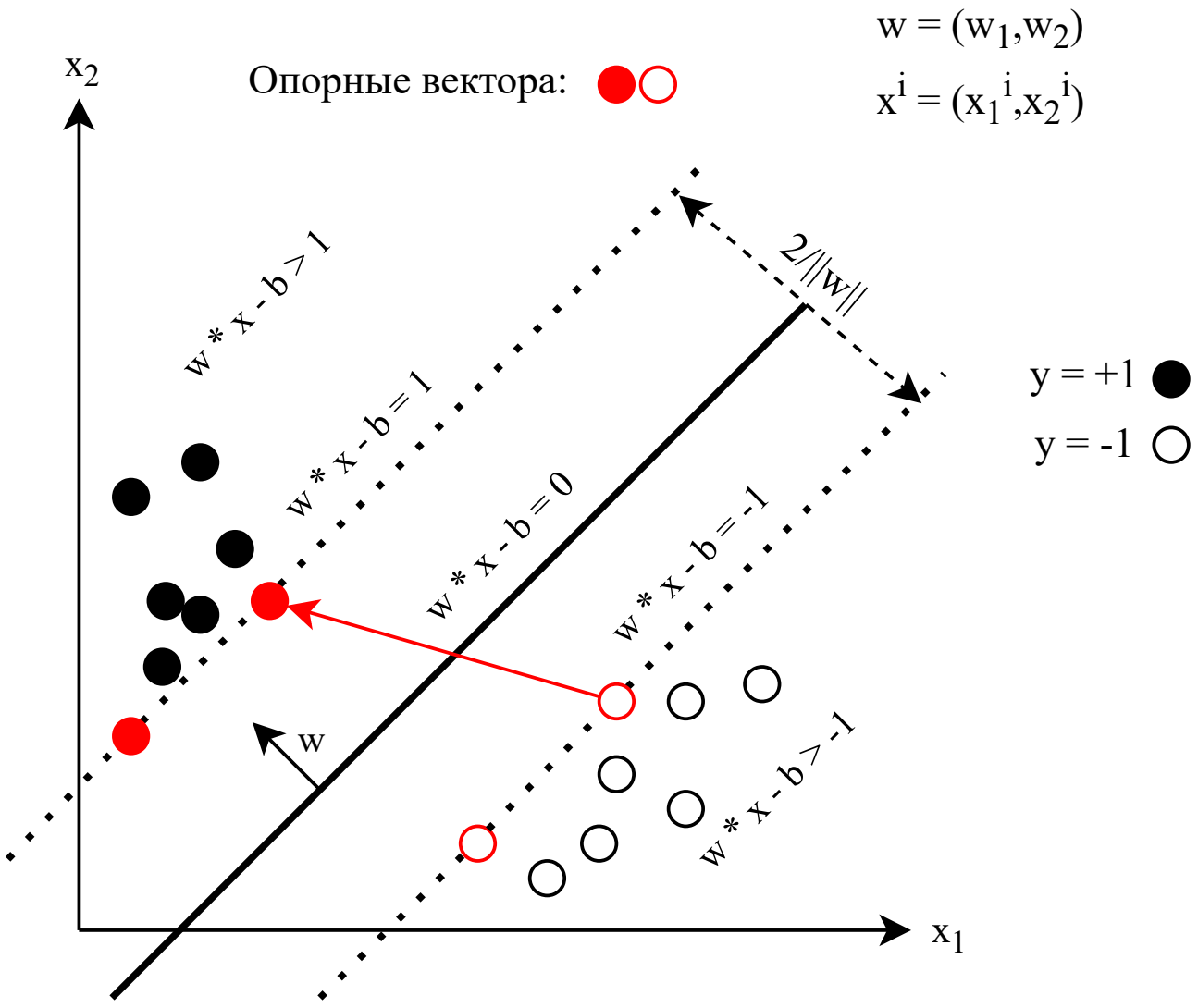

Рисунок 1.10 – Вывод правил настройки весов

В формулах 1.5 — 1.8 представлен вывод правил настройки весов.

$$\langle (x_+ - x_-), \frac{w}{\|w\|} \rangle = \frac{\langle x_+, w \rangle - \langle x_-, w \rangle}{\|w\|} = \frac{(b+1) - (b-1)}{\|w\|} = \frac{2}{\|w\|}, \tag{1.5}$$

$$\frac{2}{\|w\|} \to max, \tag{1.6}$$

$$\|w\| \to min, \tag{1.7}$$

$$\frac{w^T w}{2} \to min. \tag{1.8}$$

Если $M > 1$, то объект x классифицируется правильно, и находится на некотором удалении от разделяющей полосы. Алгоритм будет правильно классифицировать объекты, если выполняется условие, представленное в формуле 1.9. Величина $M = y(w^T x - b)$ называется отступом объекта x от границы классов. В случае отрицательного отступа алгоритм допускает ошибку на объекте. Если $M \in (0, 1)$, то объект попадает внутрь разделяющей полосы. Если $M > 1$, то объект x классифицируется корректно, и находится на удалении от разделяющей полосы. Алгоритм будет корректно классифицировать объекты, если выполняется условие, представленное в формуле 1.9.

$$y(w^T x - b) \geq 1. \tag{1.9}$$

1.5.2 Алгоритм наивной Байесовской классификации

Простой вероятностный классификатор, известный как наивный Байесовский классификатор, использует теорему Байеса с предположением о независимости [12].

Уравнение, описывающее взаимосвязь условных вероятностей статистических величин, известно как теорема Байеса. В байесовской классификации главный интерес заключается в определении вероятности отнесения к определенной категории по некоторым наблюдаемым характеристикам.

Целью классификации является поиск наилучшего класса, то есть имеющего наибольшую апостериорную вероятность $P(c_i|d_j)$:

$$c^* = arg_{c_j \in C} max P(c_i|d_j), \tag{1.10}$$

где $d_i \in \Omega, c_j \in C$.

По формуле Байеса:

$$P(c_i|d_j) = \frac{P(c_i)P(d_i|c_j)}{P(d_i)} \approx P(c_j)P(d_i|c_j), \tag{1.11}$$

где $P(c_j)$ — априорная вероятность, что объект принадлежит c_j, а $P(d_i|c_j)$ — вероятность встретить объект типа d_i среди объектов класса c_j.

В прикладных приложениях для оценки параметров наивных Байесовских моделей часто применяется метод максимального правдоподобия,

который позволяет работать с моделью, не обращая внимания на Байесовскую вероятность и байесовские методы.

Хотя классификатор имеет очень простые условия и наивный вид, он успешно справляется с обработкой самых разнообразных данных.

Достоинством наивного Байесовского классификатора является малое количество данных, необходимых для обучения, оценки параметров и классификации.

1.5.3 Метод деревьев принятия решений

Метод деревьев принятия решений — машинное обучение, при котором данные непрерывно разделяются в соответствии с определенным параметром [13].

Дерево состоит из следующих элементов:

— узлы — сетевые параметры;

— листья — метки классов;

— рёбра — веса сетевых параметров.

Пример дерева принятия решения представлен на рисунке 1.12

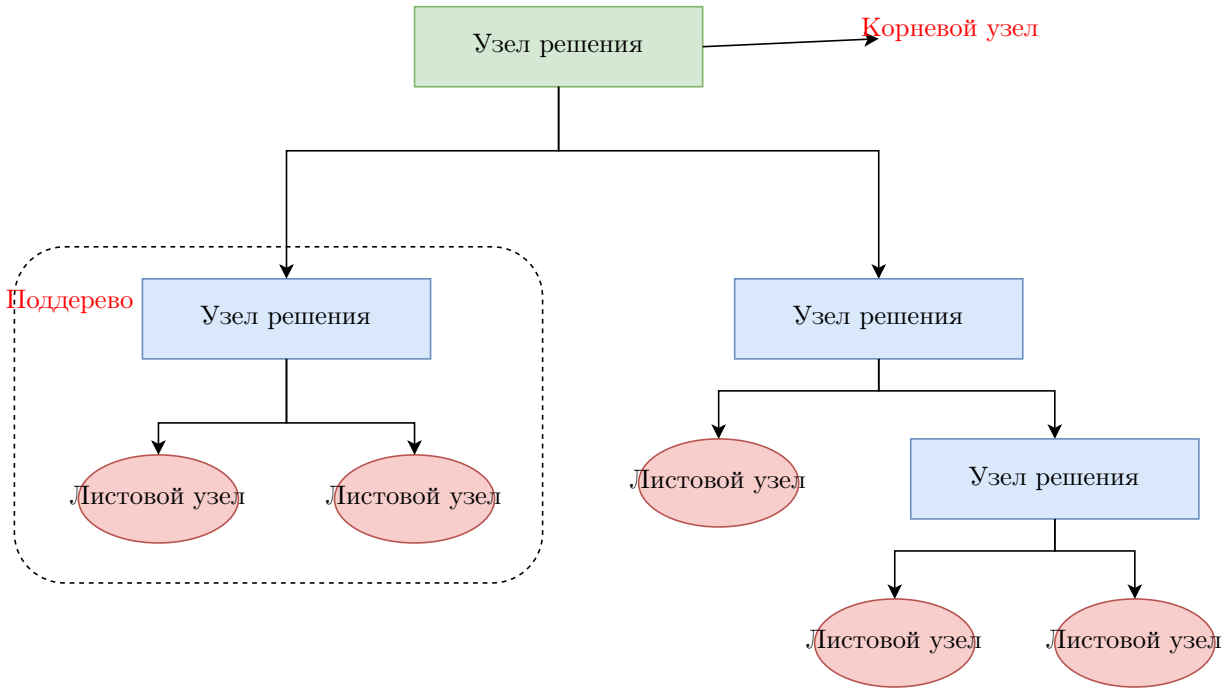

Рисунок 1.11 – Дерево принятия решений

Процесс классификации заключается в поочередных перемещениях между узлами дерева в зависимости от определенных признаков данного объекта. Процесс классификации подходит к концу в тот момент, когда достигнут некий конечный лист дерева. Лист, до которого дошел процесс классификации, определяет определенный класс и, следовательно, рассматриваемый объект является объектом данного класса. В задачах классификации чаще всего встречаются бинарные деревья решений, и результат перехода по ребрам определяется наличием или отсутствием некого признака в объекте.

Одним из главных преимуществ такого метода классификации является легкость в интерпретации результата, а также меньший объем фильтрации. Однако деревья решений имеют неустойчивый характер предсказаний, то есть небольшое изменение входных данных влечет за собой значительное изменение структуры дерева.

1.5.4 Метод обратного распространения ошибки для обучения нейронных сетей

Метод обратного распространения ошибки (от англ. backpropagation, BP) — метод вычисления градиента, который используется при обновлении весов в многослойной нейронной сети [14].

Идея BP состоит в распространении сигналов ошибки от выходов многослойной нейронной сети к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы.

— на вход нейрона поступает кортеж чисел вида $(x^1, ..., x^n) \in \mathbb{R}^n$,

— на выходе нейрон выдает число $a = \sigma(\langle x, w\rangle - w_0)$, где σ — функция активации.

При заданной совокупности w значений весовых коэффициентов w_{ij} многослойная нейронная сеть определяет функцию a_w, отображающую каждый входной вектор $x \in \mathbb{R}^M$. Если задана обучающая выборка $S \subseteq \mathbb{R}^n \times \mathbb{R}^M$, то ошибкой на паре $(x, y) \in S$ называется число

$$Q(x, y, w) = \frac{1}{2}(a_w(x) - y)^2. \tag{1.12}$$

Задача алгоритма BP заключается в нахождении такой совокупности w весовых коэффициентов, которые делают ошибки (1.3) как можно меньше. Алгоритм BP решает эту задачу путем выполнения нескольких итераций, каждая из которых состоит из двух частей:

— выбор пары $(x, y) \in S$,

— нахождение ошибки 1.12 на выбранной паре (x, y) при текущем наборе весовых коэффициентов w путем вычисления в «прямом направлении» выходов всех нейронов,

— коррекция весовых коэффициентов w_{ij} путем вычисления в «обратном направлении» (сначала корректируются весовые коэффициенты последнего слоя, затем — предпоследнего, и т.д.).

Алгоритм BP имеет следующий вид:

1. Инициализация весов небольшими случайными значениями.

2. Производятся итерации (до тех пор пока Q не стабилизируется), каждая итерация заключается в вычислении по текущему набору w весовых коэффициентов нового набора w', который будет текущим в следующей итерации, и имеет следующий вид:

   — случайно выбираются $(x, y) \in S, x = (x^1, ..., x^n), y = (y^1, ..., y^M)$,

   — прямой выход: вычисляются выходы нейронов, и

$$\begin{aligned} Q(x, y, w) &= \frac{1}{2}\sum_{m=1}^{M}(a^m - y^m)^2 \\ \forall m = 1, ..., M \quad \frac{\partial Q}{\partial a^m} &= a^m - y^m = \xi^m, \end{aligned} \tag{1.13}$$

   — обратный ход (модификация весов в направлении $-\nabla$):

$$\begin{aligned} w'_{hm} &= w_{hm} - \frac{\partial Q}{\partial w_{hm}}\eta, \\ w'_{jh} &= w'_{jh} - \frac{\partial Q}{\partial w_{jh}}\eta \end{aligned} \tag{1.14}$$

где $\eta \in (0,1)$ — подбираемый параметр (темп обучения), и частные производные $\frac{\partial Q}{\partial w_{hm}}$, $\frac{\partial Q}{\partial w_{jh}}$ вычисляются следующим образом: пусть $u^1, ..., u^H$ — выходы первого слоя, тогда $\forall m = 1, ..., M \quad \forall h = 1, ..., H, \quad \forall j = 1, ...n$,

$$a^m = \sigma(\sum_{h=1}^{H} w_{hm} u^h - w_{0m}),$$

$$\frac{\partial Q}{\partial w_{hm}} = \frac{\partial Q}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \xi^m \sigma'(\sum_{h=1}^{H} w_{hm} u^h - w_{0m}) u^h = \xi^m a^m (1 - a^m) u^h,$$

$$\frac{\partial Q}{\partial w_{0m}} = \frac{\partial Q}{\partial a^m}\frac{\partial a^m}{\partial w_{0m}} = \xi^m \sigma'(\sum_{h=1}^{H} w_{hm} u^h - w_{0m})(-1) = -\xi^m a^m (1 - a^m),$$

$$u^h = \sigma(\sum_{j=1}^{n} w_{jh} x^j - w_{0h}),$$

$$\frac{\partial Q}{\partial u^h} = \sum_{m=1}^{M} \frac{\partial Q}{\partial a^m}\frac{\partial a^m}{\partial u^h} = \sum_{m=1}^{M} \xi^m \sigma'(\sum_{h=1}^{H} w_{hm} u^h - w_{0m}) w_{hm} =$$

$$= \sum_{m=1}^{M} \xi^m a^m (1 - a^m) w_{hm} = \zeta^h$$

$$\frac{\partial Q}{\partial w_{jh}} = \frac{\partial Q}{\partial u^h}\frac{\partial u^h}{\partial w_{jh}} = \zeta^h \sigma'(\sum_{j=1}^{n} w_{jh} x^j - w_{0h}) x^j = \zeta^h u^h (1 - u^h) x^j,$$

$$\frac{\partial Q}{\partial w_{0h}} = \frac{\partial Q}{\partial u^h}\frac{\partial u^h}{\partial w_{0h}} = \zeta^h \sigma'(\sum_{j=1}^{n} w_{jh} x^j - w_{0h})(-1) = -\zeta^h u^h (1 - u^h)$$

Основным преимуществом является универсальность, поскольку данный метод работает с любой конфигурацией многослойной нейронной сети. К недостаткам можно отнести долгий процесс обучения, а также вероятность получить большие значения весов. Это может привести к тому, что большинство нейронов будут функционировать при очень больших значениях весовых коэффициентов, в области, где производная функции активации очень мала. Так как обратно распространяемая в процессе обучения ошибка пропорциональна этой производной, то процесс обучения может стать парализованным.

1.5.5 Метод k-ближайших соседей

Пусть задана обучающая выборка $S \subseteq X \times Y$ [12]. Если на множестве X объектов задана метрика p, то $\forall x \in X$ объекты из множества X_s можно упорядочить в соответствии с их близостью к x, т.е. расположить в последовательность

$$x_1, ..., x_{|S|}, \tag{1.15}$$

удовлетворяющую условию:

$$p(x, x_1) \leq ... \leq p(x, x_{|S|}) \tag{1.16}$$

т.е. первым в 1.15 расположен ближайший к x объект, затем — следующий по близости к x объект, и т.д. $\forall i = 1, ..., l$ объект x_i из последовательности 1.16 называется i–м ближайшим соседом к x. Метод ближайших соседей для построения аппроксимирующей функции a_S заключается в том, что выбираются

— натуральное число $k \geq 1$ (число ближайших к x объектов, ответы на которых учитываются при вычислении ответа $a_S(x)$),

— действительные числа $w_1 \geq ... \geq w_k > 0$, которые имеют смысл весов, определяющих вклад ближайших k соседей объекта x из обучающей выборки S в вычисление ответа для объекта x, например,

$$\forall i = 1, ..., k \quad w_i = \frac{k + 1 - i}{k}. \tag{1.17}$$

$\forall x \in X$ значение $a_S(x)$ определяется как такой ответ $y \in Y$, который максимизирует значение выражения

$$\sum_{i=1}^{k} (y_{x_i} = y) w_i, \tag{1.18}$$

т.е. как такой ответ y, который наиболее характерен среди ответов на k ближайших соседей x из обучающей выборки S. Оптимальным является такое

k, которое минимизирует риск

$$\sum_{(x,y_x)\in S}(a^k_{S\setminus\{(x,y_x)\}}(x) \neq y_x), \tag{1.19}$$

где a^k_S — построенная аппроксимирующая функция в соответствии с приведенным выше определением.

1.6 Сравнение рассмотренных методов

Существуют различные способы оценки модели классификации:

— F-мера (f1-score);

— точность (precision);

— время обучения (training time);

— аккуратность (accuracy);

— полнота (recall).

1.6.1 Аккуратность

Наиболее простая метрика, определяющая долю правильных ответов результатов модели. Рассчитывается следующим образом:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{1.20}$$

где TN — истинно отрицательные ответы, TP — истинно положительные ответы, FN — ложно отрицательные ответы, FP — ложно положительные ответы. Однако данная метрика не подходит в случае, если объем данных между классами распределен неравномерно.

1.6.2 Точность

Данная метрика показывает долю правильных ответов относительно всех положительных ответов в пределах одного класса. Формула вычисления приведена ниже:

$$precision = \frac{TP}{TP + FP}. \tag{1.21}$$

Учитывая данную метрику, можно сказать о количестве ложно положительных предсказаний и сделать вывод о достаточности объема данных других классов.

1.6.3 Полнота

Полнота — это доля истинно положительных классификаций. С её помощью можно сказать какую долю объектов, которые действительно относятся к данному классу, модель предсказала верно.

$$recall = \frac{TP}{TP + FN}. \tag{1.22}$$

Полнота показывает способность алгоритма обнаруживать данный класс.

1.6.4 F-мера

F-мера — это гармоническое среднее между точностью и полнотой. Данная метрика позволяет найти баланс между этими метриками, так как на практике зачастую невозможно достичь максимальной точности и полноты одновременно.

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{1.23}$$

Метрика достигает своего максимума при наибольших значениях полноты и точности, и стремится к нулю, если один из аргументов близок к нулю.

1.6.5 Результаты сравнений

Результаты сравнений на наборе данных UNSW-NB15 [15] приведены в таблице 1.1.

Таблица 1.1 – Сравнение методов

	F-мера	**Аккуратность**	**Точность**	**Полнота**	**Время**
K-ближайших соседей	0.96	0.97	0.96	0.97	Высокое
BP	0.9	0.92	0.89	0.92	Среднее

Деревья принятия решений	0.92	0.9	0.91	0.93	Низкое
SVM	0.88	0.9	0.87	0.88	Среднее
Наивная Байесовская классификация	0.79	0.8	0.78	0.78	Низкое

1.7 Вывод

Были рассмотрены модели сетевых атак, а также была приведена их классификация. Дано пределение понятия нейронной сети, описаны архитектуры нейронных сетей и принцип их работы. Рассмотрены методы машинного обучения для решения задачи классификации сетевых атак. В качестве выбранной технологии для разрабатываемого метода выбрана многослойная нейронная сеть, так как она широко используется в задачах классификации. Исходя из результатов сравнительного анализа для решения задачи классификации сетевых атак был выбран метод обратного распространения ошибки.

2 Конструкторский раздел

В данном разделе описываются требования к разрабатываемому методу и программному комплексу. Рассматривается архитектура нейронной сети и структура программного обеспечения. Описываются данные для обучения модели.

2.1 Требования к разрабатываемому методу

Метод распознавания сетевых атак должен:

— принимать на вход данные о сетевых параметрах в формате CSV;

— определять наличие сетевой атаки во входных данных;

— определять вероятность принадлежности атаки к определенному классу.

2.2 Требования к разрабатываемому программному комплексу

Программный комплекс, реализующий интерфейс для разработанного метода, должен предоставлять:

— возможность загрузки параметров сетевого трафика через графический интерфейс;

— возможность просмотра информации о сетевом трафике;

— возможность просмотра информации о количестве атак и их типах в исходных данных.

2.3 Набор данных CICIDS2017

Набор данных CICIDS2017 содержит доброкачественные и самые современные распространенные атаки. Он также включает в себя результаты анализа сетевого трафика с маркированными потоками на основе временной метки, портов источника и назначения, протоколов и атак. Описание параметров приведено в таблице 2.1.

Таблица 2.1 – Описание параметров датасета CICIDS2017

№	**Название параметра**	**Описание**
1	Flow Duration	Продолжительность потока
2	Source Port	Порт источника
3	Destination Port	Порт назначения
4	Protocol ID	ID протокола
5	Total Fwd Packets	Всего пакетов в прямом направлении
6	Total Backward Packets	Всего пакетов в обратном направлении
7	Fwd Packet Length Min	Минимальный размер пакета в прямом направлении
8	Fwd Packet Length Max	Максимальный размер пакета в прямом направлении
9	Fwd Packet Length Mean	Средний размер пакета в прямом направлении
10	Fwd Packet Length Std	Размер стандартного отклонения пакета в прямом направлении
11	Fwd IAT Total	Общее время между двумя пакетами, отправленными в прямом направлении
12	Fwd IAT Mean	Среднее время между двумя пакетами, отправленными в прямом направлении
13	Fwd IAT Std	Время стандартного отклонения между двумя пакетами, отправленными в прямом направлении
14	Fwd IAT Max	Максимальное время между двумя пакетами, отправленными в прямом направлении
15	Fwd IAT Min	Минимальное время между двумя пакетами, отправленными в прямом направлении
16	Fwd IAT Mean	Среднее время между двумя пакетами, отправленными в прямом направлении
17	SYN Flag Count	Количество пакетов с SYN
18	ACK Flag Count	Количество пакетов с ACK
19	FIN Flag Count	Количество пакетов с FIN

20	CWE Flag Count	Количество пакетов с CWE
21	Bwd Packet Length Min	Минимальный размер пакета в обратном направлении
22	Bwd Packet Length Max	Максимальный размер пакета в обратном направлении
23	Bwd Packet Length Mean	Средний размер пакета в обратном направлении
24	Bwd Packet Length Std	Размер стандартного отклонения пакета в обратном направлении
25	Bwd IAT Total	Общее время между двумя пакетами, отправленными в обратном направлении
26	Bwd IAT Mean	Среднее время между двумя пакетами, отправленными в обратном направлении
27	Bwd IAT Std	Время стандартного отклонения между двумя пакетами, отправленными в обратном направлении
28	Bwd IAT Max	Максимальное время между двумя пакетами, отправленными в обратном направлении
29	Bwd IAT Min	Минимальное время между двумя пакетами, отправленными в обратном направлении
30	Bwd IAT Mean	Среднее время между двумя пакетами, отправленными в обратном направлении
31	Attack Label	Метка атаки

2.4 Структура разрабатываемого программного комплекса

На рисунке 2.1 представлена диаграмма IDEF0 обучения модели.

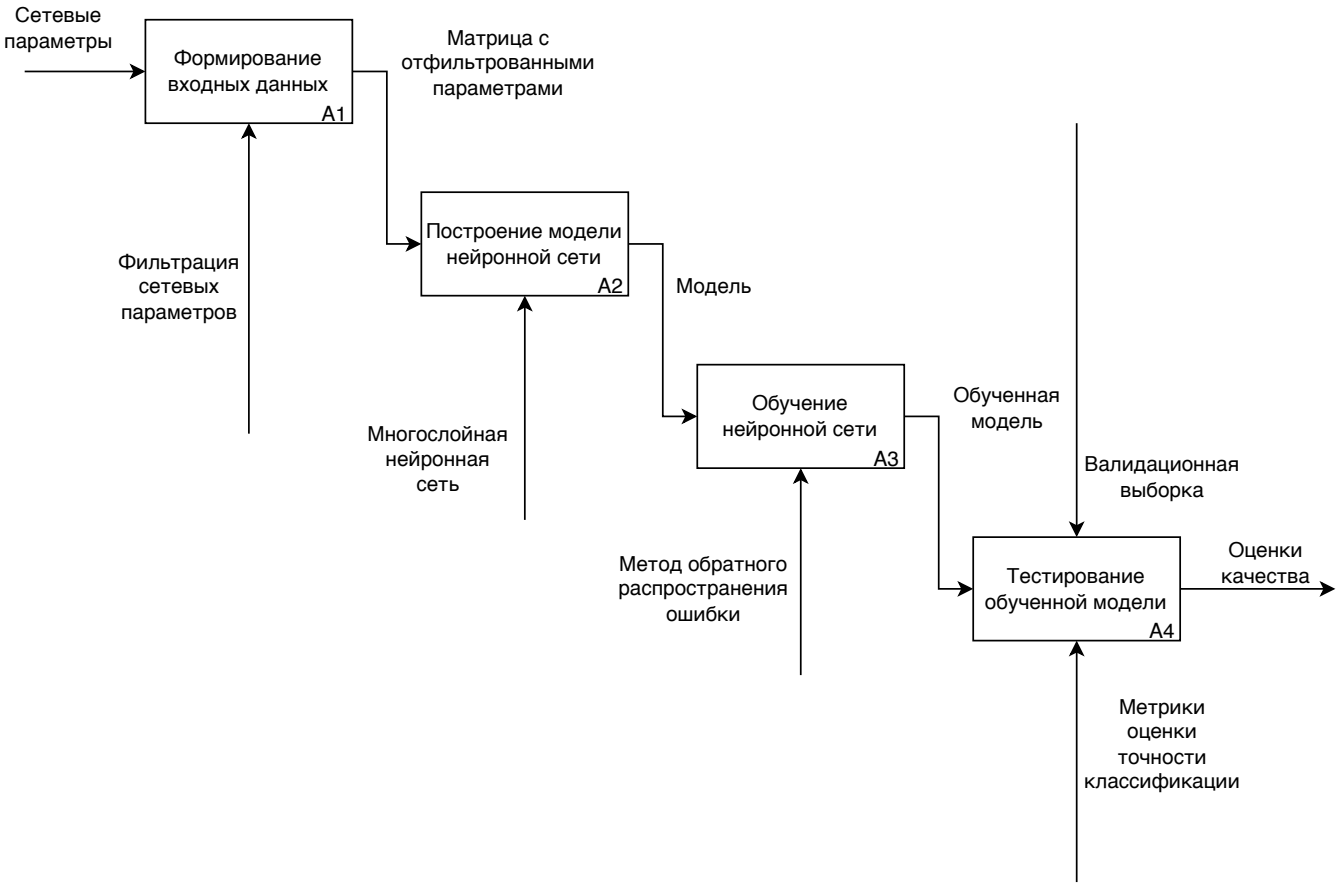

Рисунок 2.1 – IDEF0–диаграмма обучения модели

2.5 Структура многослойной нейронной сети

Поскольку некоторые семейства атак в сети проще обнаружить, чем другие, архитектура многослойной нейронной сети будет модифицирована так, чтобы для легко классифицируемых образцов сеть не должна была оценивать все слои.

Такая архитектура основана на предположении о том, что нейронные сети с большим количеством слоев могут обучаться все более сложным функциям, которые, в свою очередь, требуются только для классификации некоторых конкретных, заведомо трудноклассифицируемых образцов.

В результате сеть строится таким образом, что к каждому слою подключается дополнительный набор нейронов (копия выходных нейронов), позволяющий осуществлять предсказания на каждом уровне.

Данная архитектура представлена на рисунке 2.2

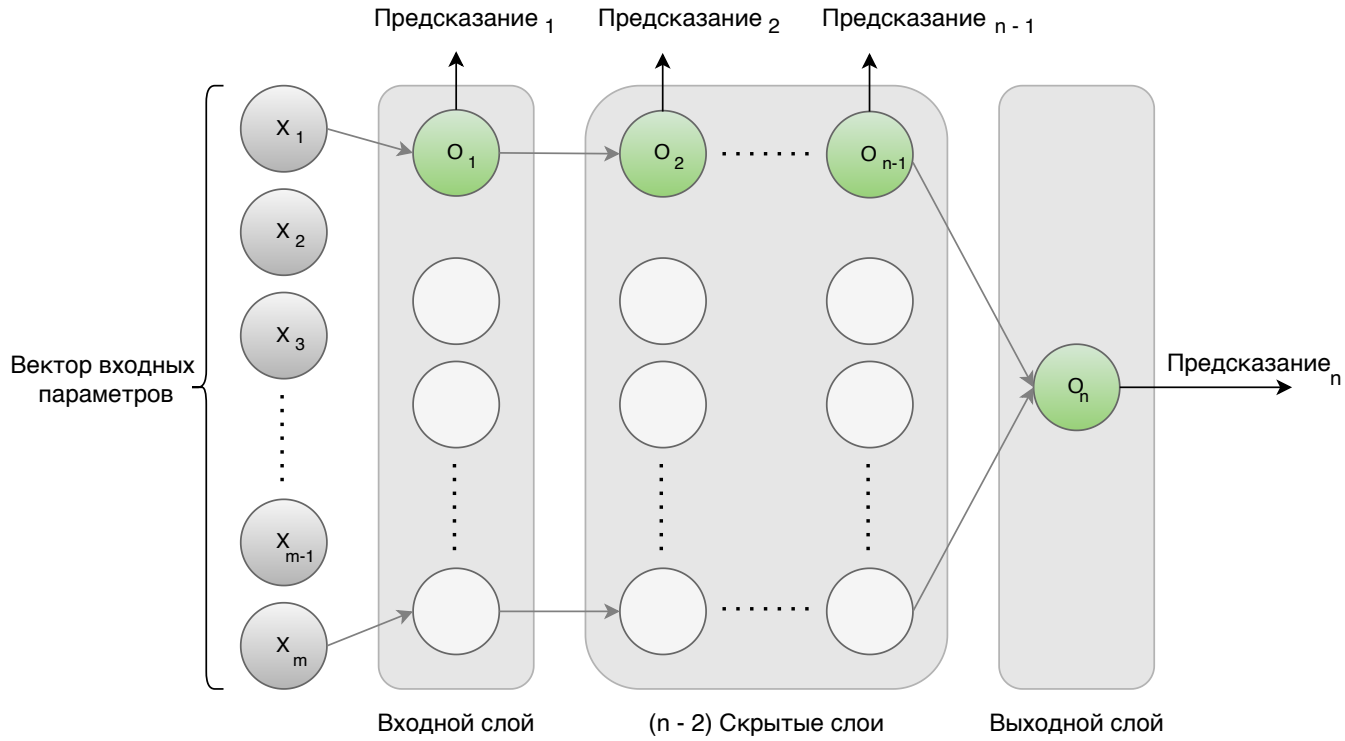

Рисунок 2.2 – Модификация многослойной нейронной сети

Исходные нейроны показаны серым цветом, а выходные нейроны (для каждого слоя) — зеленым. Нейронная сеть продолжает оценивать слои один за другим и выдает значение достоверности прогноза на каждом слое. Значение достоверности определяется как величина, которая отражает, насколько сеть уверена в принадлежности образца к определенному классу. Это число от 0,5 до 1, которое получается путем применения сигмовидной функции к выходному нейрону.

Оценивая значение достоверности, нейронная сеть определяет, следует ли обрабатывать дополнительные слои или полученное в данный момент значение достоверности достаточно высоко. Эта модификация гарантирует, что более простые образцы будут классифицироваться на ранних слоях, а более сложные образцы будут передаваться в более глубокие слои.

2.6 Вывод

Были представлены требования к разрабатываемому методу обнаружения и программному комплексу.

Представлены схемы работы с разрабатываемыми программными модулями.

Представлена модификация архитектуры многослойной нейронной сети.

Представлен выбор датасета для обучения модели и описаны входные данные.

3 Технологический раздел

В данном разделе описываются средства, используемые для разработки программного обеспечения. Приводятся детали реализации программных компонентов и процесс обучения разрабатываемой нейронной сети.

3.1 Средства реализации программного комплекса

3.1.1 Выбор языка программирования

Для написания программного обеспечения будет использоваться язык программирования Python [16] версии 3.11.

Данный выбор обусловлен следующими факторами:

— наличие большого количества библиотек для работы с нейронными сетями;

— кроссплатформенность;

— автоматическое управление памятью.

3.1.2 Используемые библиотеки

При разработке программного обеспечения использовались следующие библиотеки:

— PyTorch [17] — библиотека, для создания и обучения модели нейронной сети. Основным преимуществом PyTorch является гибкость. PyTorch предоставляет множество инструментов для разработки своих моделей, включая выбор оптимизаторов, функций потерь и слоев;

— PyQt5 [18] — библиотека для создания графического интерфейса;

— Scikit-learn [19] — библиотека включает в себя подготовку данных для последующей классификации, а также различные алгоритмы машинного обучения и поддерживает взаимодействие с NumPy;

— Numpy [20] — это библиотека, которая предоставляет функциональность для работы с многомерными массивами и матрицами. Она используется для научных вычислений, обработки данных и машинного обучения;

— Matplotlib [21] — библиотека для визуализация результатов экспериментов и исследований, создание диаграмм и др.

3.2 Минимальные требования к вычислительной системе

Для того, чтобы воспользоваться программным обеспечением, потребуется ЭВМ с предустановленным интерпретатором Python версии 3.11. Также необходимо скачать и установить все использованные в проекте библиотеки.

Требования к использованию определенной операционной системы отсутствуют поскольку ПО реализовано на языке Python, который является кроссплатформенным.

3.3 Формат входных данных

Входными данными являются текстовые файл в формате CSV. CSV — формат текстовых данных в котором каждая строка — это отдельная строка таблицы, а столбцы отделены один от другого специальными символами-разделителями — запятыми. Файл CSV содержит информацию о сетевых параметрах. Файлы данного формата могут быть открыты на любой операционной системе в любом программном обеспечении для работы с текстом.

3.4 Формат выходных данных

Выходные данные модуля классификации можно условно разделить на два типа:

— прогнозирование класса атаки для каждой строки из входного файла;

— визуализация количества атак принадлежащих к различным классов сетевых атак.

Полученные выходные данные выводятся на графический интерфейс программного обеспечения

3.5 Структура разработанного ПО

Структура разработанного ПО изображена на рисунке 3.1.

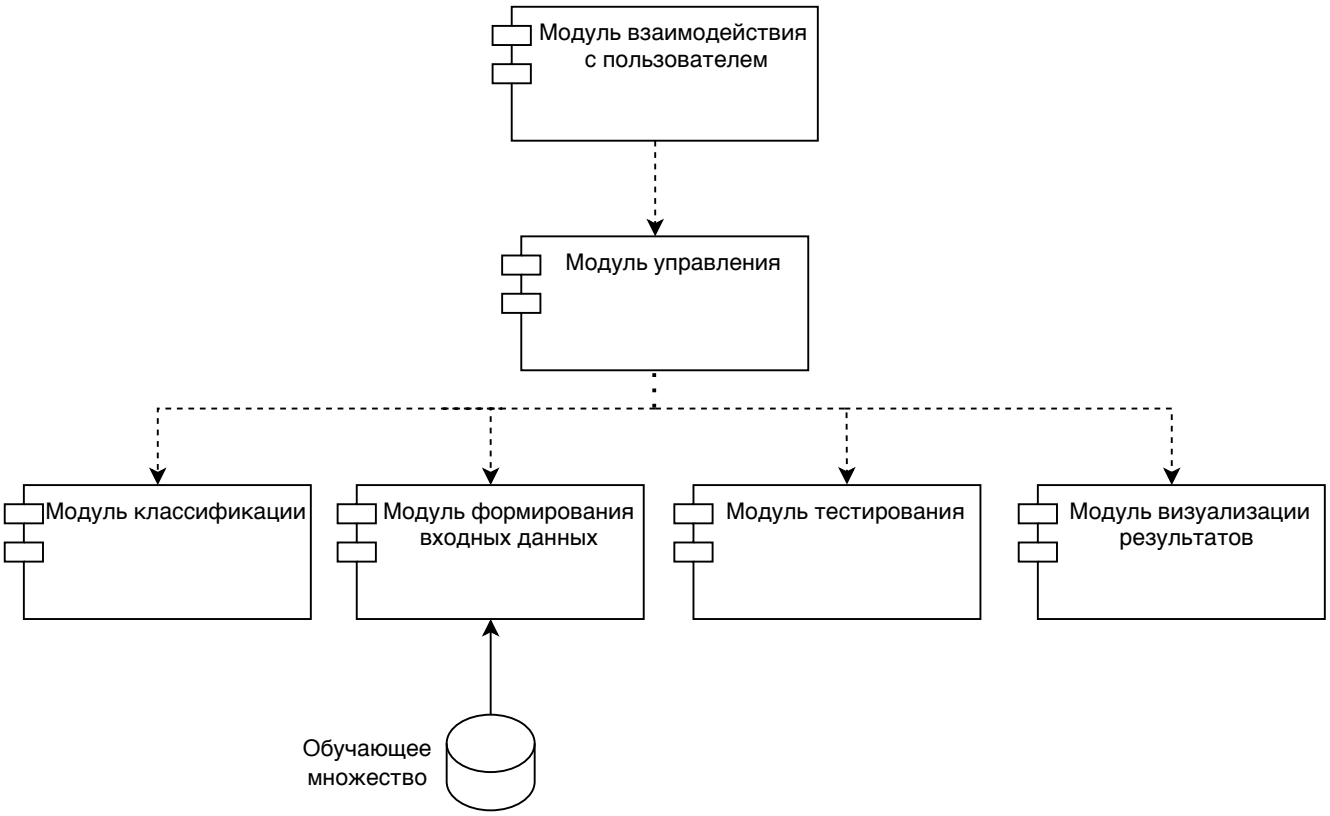

Рисунок 3.1 – Диаграмма структуры ПО

Каждый модуль, который изображен на диаграмме, содержит в себе сгруппированные по функциональному значению соответствующие функции и классы. Модуль взаимодействия с пользователем отвечает за пользовательский интерфейс. Модуль управления связывает модули классификации, формирования данных, визуализации результатов и тестирования а также отвечает за координацию их работы.

3.6 Установка программного обеспечения

Для запуска разработанного программного продукта требуется установить на ЭВМ интерпретатор для Python 3.11 и создать виртуальное окружение для ПО. В листинге 3.1 приведены команды, которые необходимо выполнить для создания виртуального окружения.

Листинг 3.1 – Команды для установки виртуального окружения

```
1    python3 -m pip install -user virtualenv
2    python3 -m venv env
3    source env/bin/activate
```

Используемые в разработке библиотеки, которые необходимы для запуска ПО, приведены в файле **requirements.txt**, который находится в корневой

директории проекта. С помощью пакетного менеджера pip все зависимости нужно установить, запустив в терминале команду, приведенную в листинге 3.2.

Листинг 3.2 – Команда для установки необходимых библиотек

```
1    pip3 install -r requirements.txt
```

3.7 Пользовательский интерфейс

Графический интерфейс разработан при помощи библотеки PyQt5, предоставляющей набор классов и методов для работы с компонентами интерфейса. На рисунке 3.2 представлен пользовательский интерфейс.

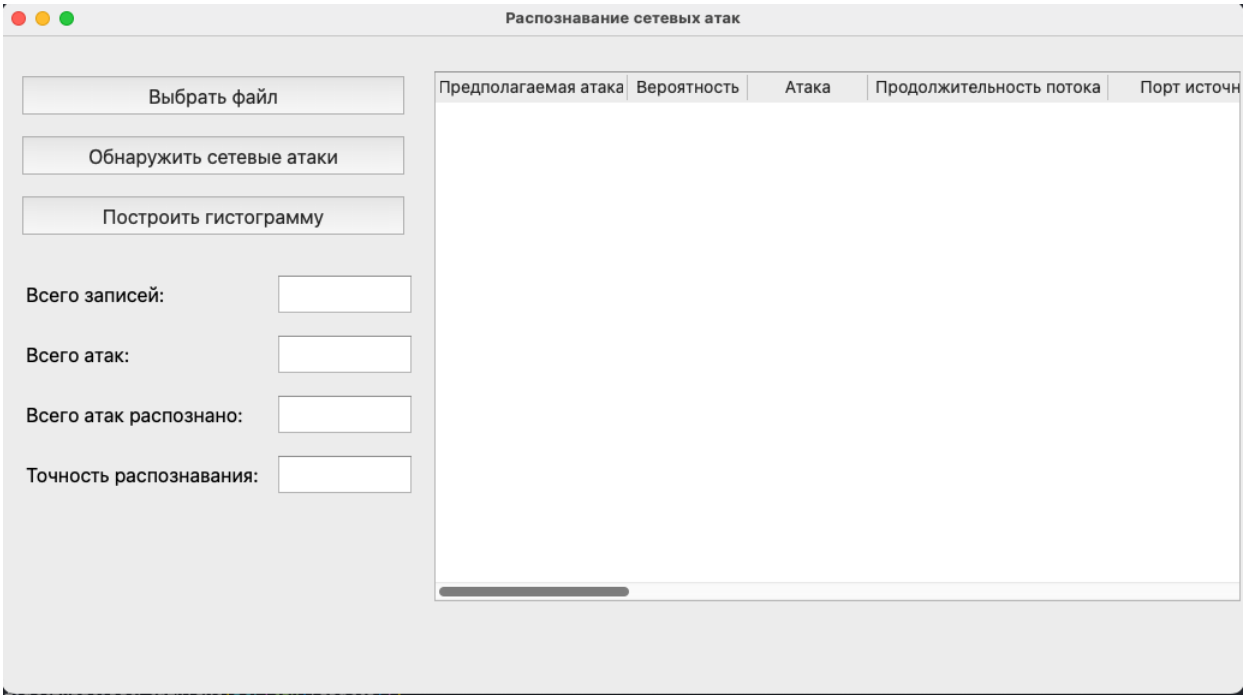

Рисунок 3.2 – Пользовательский интерфейс

В колонке «Предполагаемая атака» отображаются результаты обнаружения сетевых атак во входном CSV-файле. В колонке «Вероятность» отображается вероятнсть принадлежности атаки к соответствующему классу.

Если пользователь, не выбрав входной файл, нажмет на кнопку «Обнаружить сетевые атаки» или на кнопку «Построить гистограмму», то на экран выведется сообщение об ошибке, как показано на рисунке 3.3.

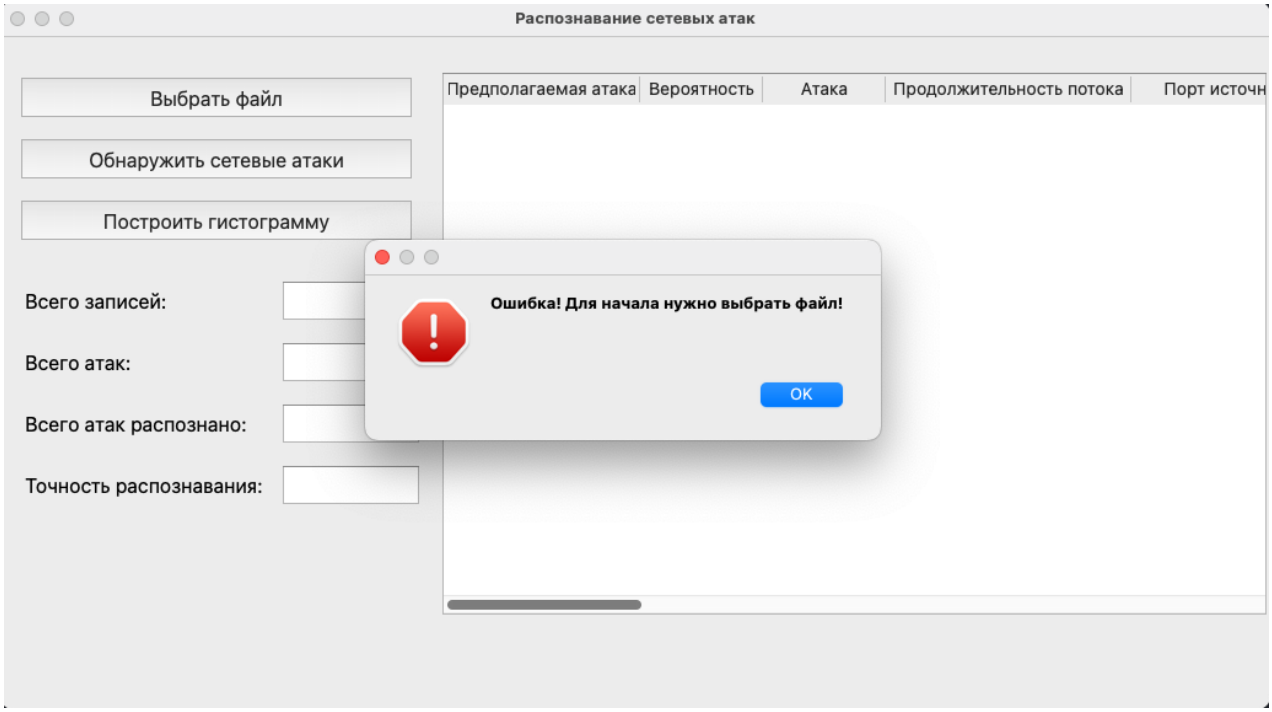

Рисунок 3.3 – Сообщение об ошибке

При нажатии на кнопку «Выбрать файл» открываются файловая система устройства, после чего необходимо выбрать интересующий CSV-файл. После выбора файла, сетевые параметры отобразятся в таблице «Исходные данные».

Если файл выбран, то при нажатии на кнопку «Обнаружить сетевые атаки», запустится алгоритм обнаружения сетевых атак для каждой строки из входного CSV-файла. Результат работы алгоритма представлен на рисунке 3.4.

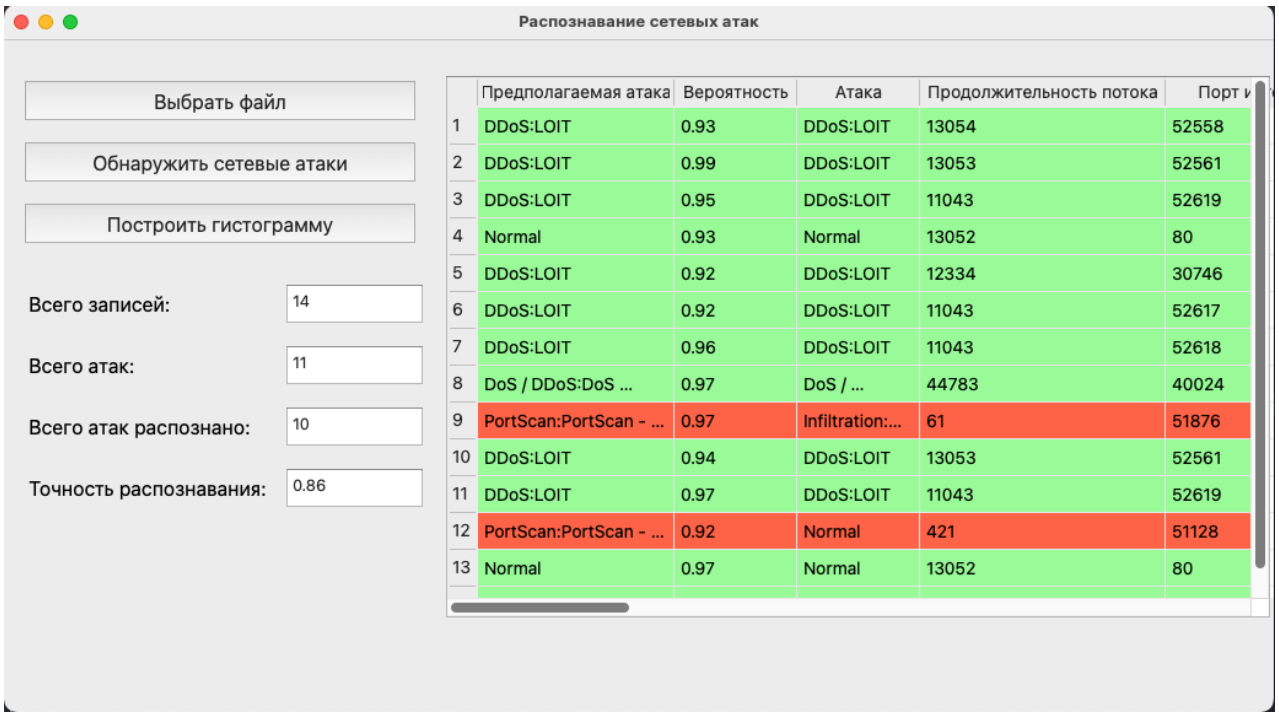

Рисунок 3.4 – Результат работы

При нажатии на кнопку «Построить гистограмму» на экран выведется гистограмма, отображающая количество атак принадлежащих к различным классам сетевых атак.

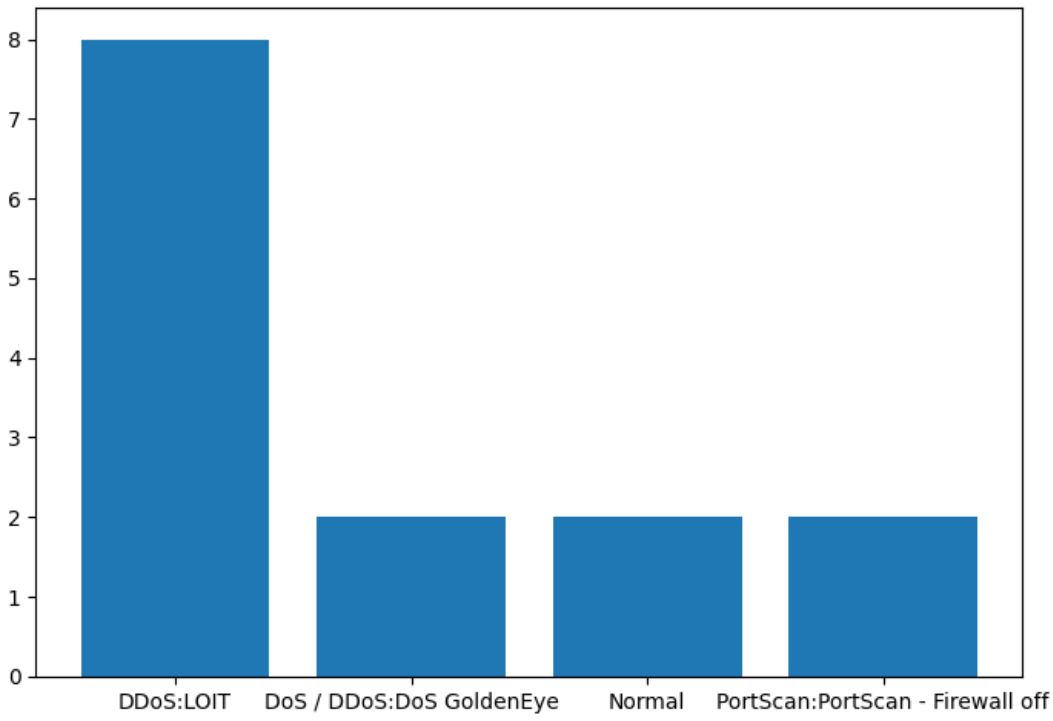

Рисунок 3.5 – Распределение сетевых атак

В листинге 3.3 приведены параметры для взаимодействия с модулем обучения модели.

Листинг 3.3 – Взаимодействие с модулем модели

```
python3 mlearn.py -h

options:
    -h, --help                show this help message and exit
    --dataroot DATAROOT       путь к датасету
    --batchSize BATCHSIZE     размер выборки данных на каждой итерации
    --nLayers NLAYERS         количество слоев
    --layerSize LAYERSIZE     размер слоя
    --niter NITER             количество эпох для обучения
    --net NET                 путь к сохраненной модели
    --manualSeed MANUALSEED   seed для перемешивания данных
    --stoppingWeightingMethod EAGERSTOPPINGWEIGHTINGMETHOD
                              функция весов
```

3.8 Тестирование программного обеспечения

Проведено тестирование программного обеспечения. Тесты приведены в таблице 3.1

Таблица 3.1 – Сравнение методов

Название функции	**Входные данные**	**Ожидаемый результат**
filter_input_data	CSV-файл с информацией о сетевых параметрах	Датасет типа pd.DataFrame [22] с отфильтрованными сетевыми параметрами
predict_all_normal	pd.DataFrame с данными о нормальном сетвом поведении	Массив данных, все элементы которого принадлежат классу Normal
predict_all_ddos_loit	pd.DataFrame с данными о сетевых атаках типа DDoS LOIT	Массив данных, все элементы которого принадлежат классу DDoS LOIT

predict_mix_attacks	pd.DataFrame с данными о всех типах сетевых атаках	Массив данных, элементы которого принадлежат соответствующим классам из входного датасета
predict_all_portscan	pd.DataFrame с данными о сетевых атаках типа PortScan	Массив данных, все элементы которого принадлежат классу PortScan
predict_all_ddos_hulk	pd.DataFrame с данными о сетевых атаках типа DDoS Hulk	Массив данных, все элементы которого принадлежат классу DDoS Hulk

3.9 Вывод

Были описаны средства реализации программного комплекса. Описаны входные и выходные данные. Описаны технологии и методы, использовавшиеся при реализации. Определены минимальные требования к вычислительной системе. Приведена инструкция для установки ПО. Приведено описание интерфейса пользователя. Проведено модульное тестирование.

4 Исследовательский раздел

В данном разделе проводится оценка точности классификации разработанного метода. Описывается его применимость для классификации различных типов сетевых атак. Приводятся преимущества и недостатки разработанного метода.

4.1 Выборка данных

Для обучения и тестирования модели на вход подаётся 2317923 записи о сетевых атаках. Для обучения модели была создана обучающая выборка, составляющая $\frac{2}{3}$ от общего числа записей. Оставшиеся $\frac{1}{3}$ необходимы для тестирования обученной модели и получения оценок качества классификции.

Однако количество записей об атаках каждого класса неоднородно, поэтому точность определения на некоторых классах слишком низкая.

4.2 Зависимость точности и потерь модели от количества эпох

На рисунках 4.1 и 4.2 представлены результаты обучения модели. Точность модели составила 92.1 процента, а потери составили 4 процента. Стоит отметить, что уже на десятой эпохе точность модели составляет более 88 процентов.

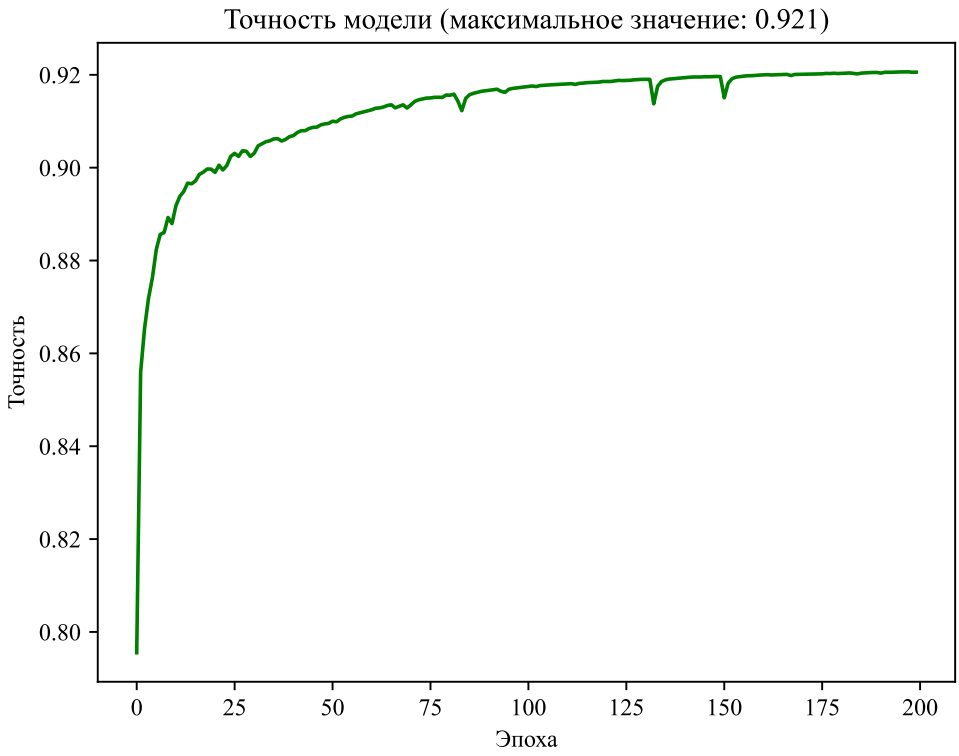

Рисунок 4.1 – Точность модели

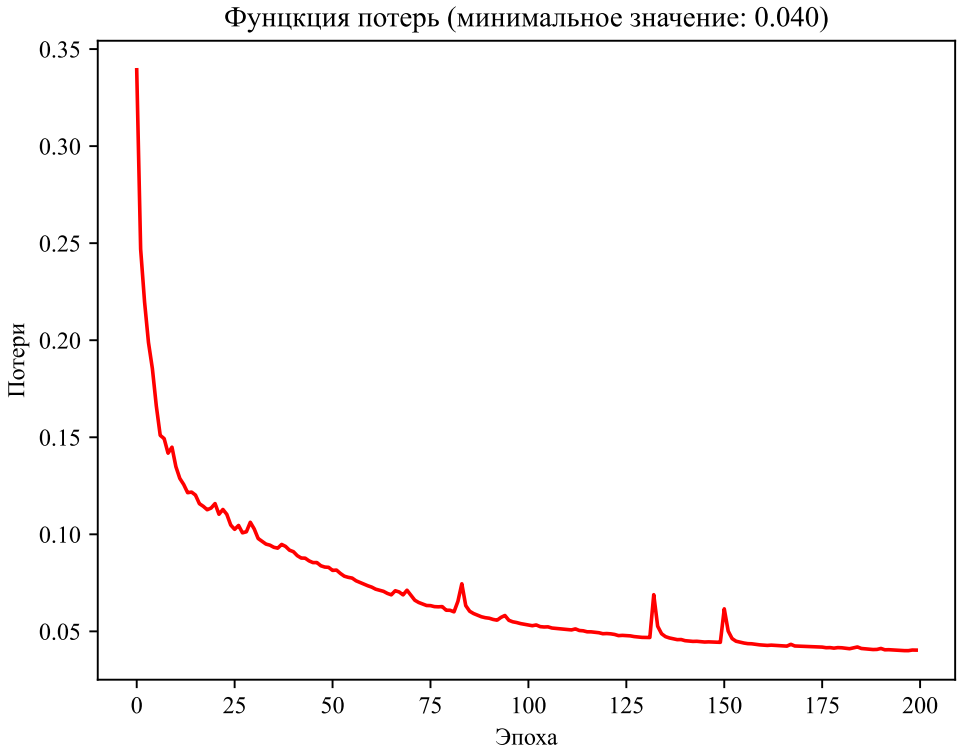

Рисунок 4.2 – Потери модели

4.3 Зависимость точности модели от класса сетевой атаки и количества слоев

На рисунках 4.3 и 4.4 продемонстрирована зависимось точности модели на каждом слое от класса сетевой атаки.

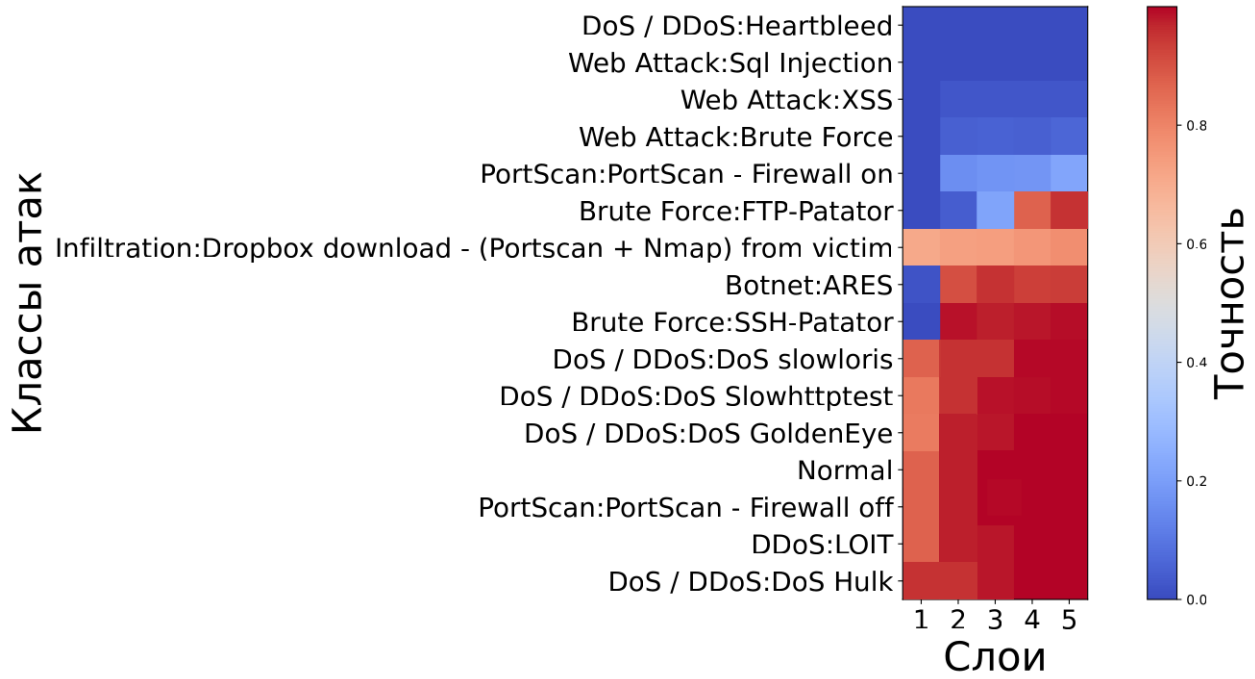

Рисунок 4.3 – 5 слоев × 128 нейронов

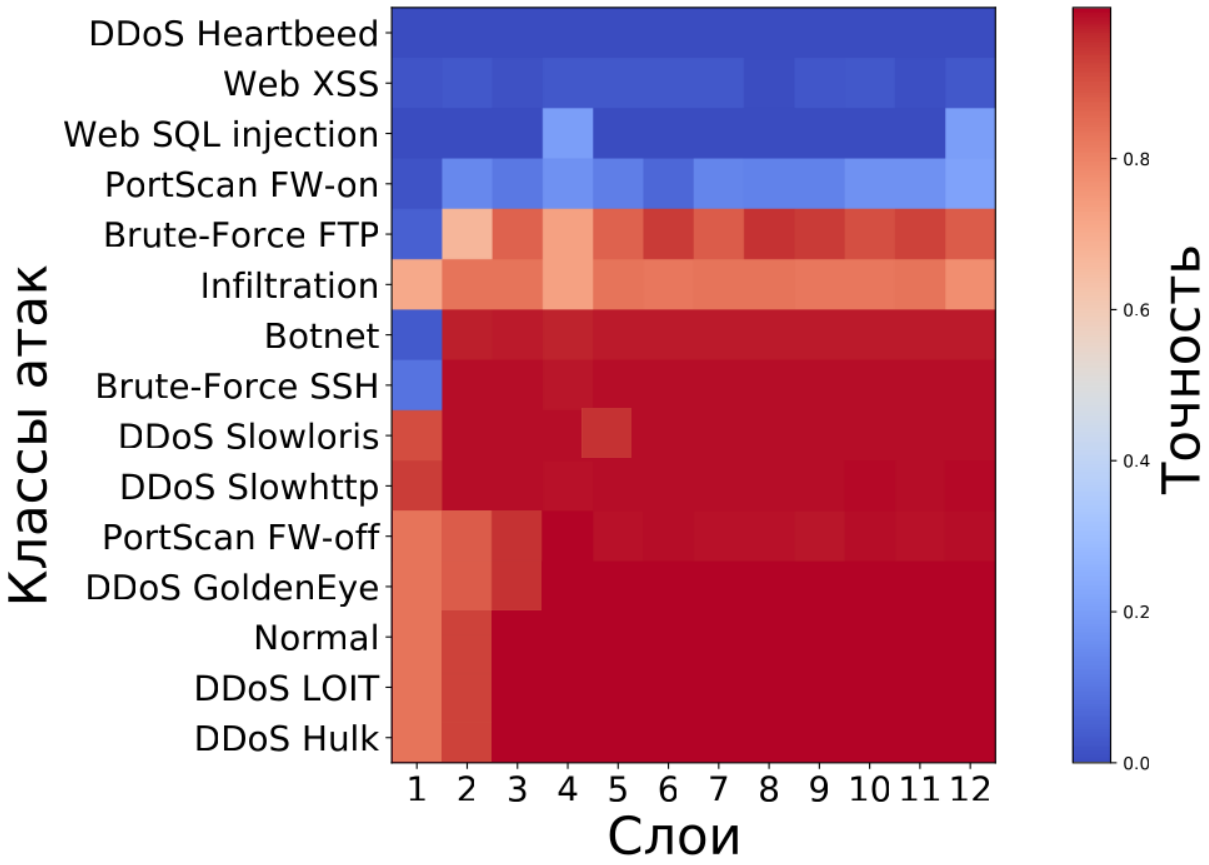

Рисунок 4.4 – 12 слоев × 64 нейрона

На основе полученных данных можно сделать следующие выводы:

— в среднем точность прогнозов возрастает по мере того, как увеличивается количество слоев. Это соответствует действительности, поскольку большее количество слоев обеспечивает большую абстракцию входного пространства и, следовательно, лучшее разделение экземпляров. Однако для получения максимальной точности для некоторых классов атак требуется всего несколько уровней. Это означает, что примененная модификация архитектуры многослойной нейронной сети позволяет значительно сокращать ресурсы при обучении, сохраняя максимальную точность;

— некоторые семейства атак (DDoS Heartbeed, Web-XSS, Web SQL injection, PortScan FW-on) демонстрируют крайне низкую точность. Это происходит по двум причинам:

1. Слишком мало выборок данных классов в обучающем множестве.
2. Потери, оптимизированные для обнаружения таких классов на определенном уровне перезаписываются потерями, оптимизированными для изучения большинства классов что приводит к стиранию информации об таких классах.

4.4 Исследование требуемого количества слоев для модели нейронной сети

Во время обучения модели пользователь может установить оптимальное значение достоверности. Установка слишком низкого значения может привести к тому, что нейронная сеть будет принимать решения на ранней стадии, тем самым используя меньше ресурсов, но при этом точность модели будет низкой. Аналогично, установка слишком высокого значения может привести к тому, что сеть всегда будет использовать все уровни, что приведет к пустой трате вычислительных ресурсов при обучении. Таким образом, желаемый уровень достоверности может быть выбран из результатов эксперимента показанных на рисунках 4.5 и 4.6.

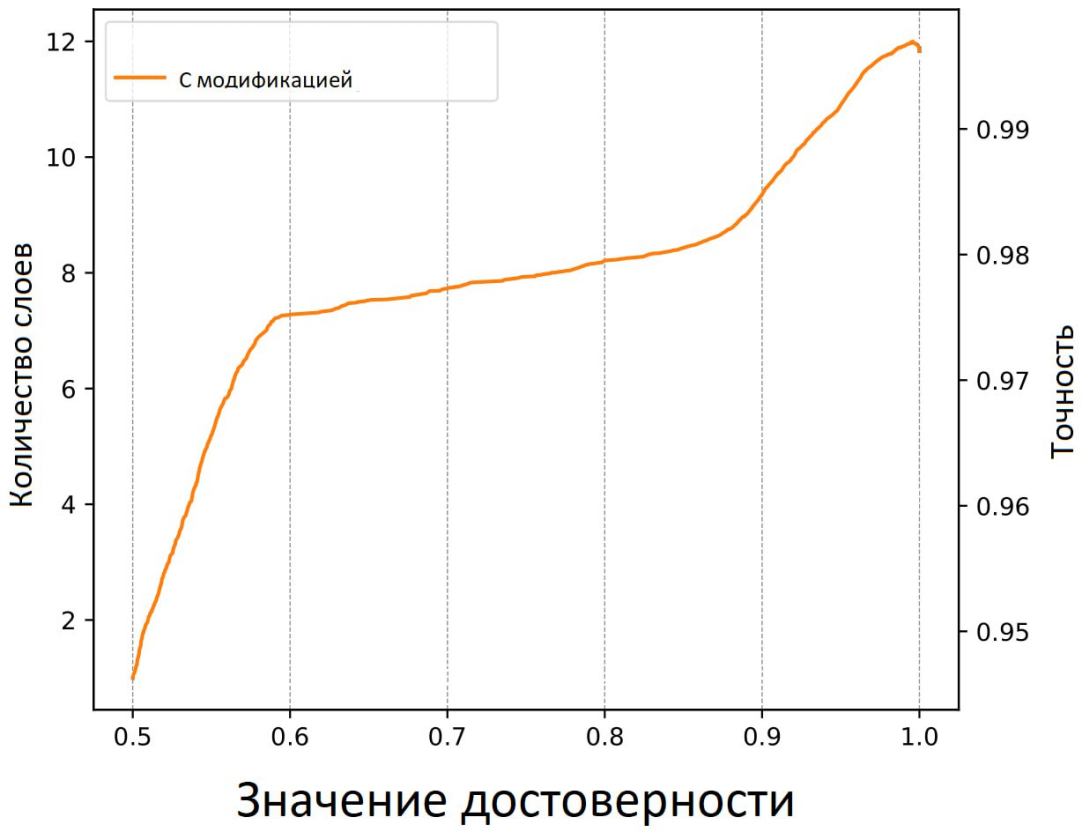

Рисунок 4.5 – 12 слоев × 64 нейрона

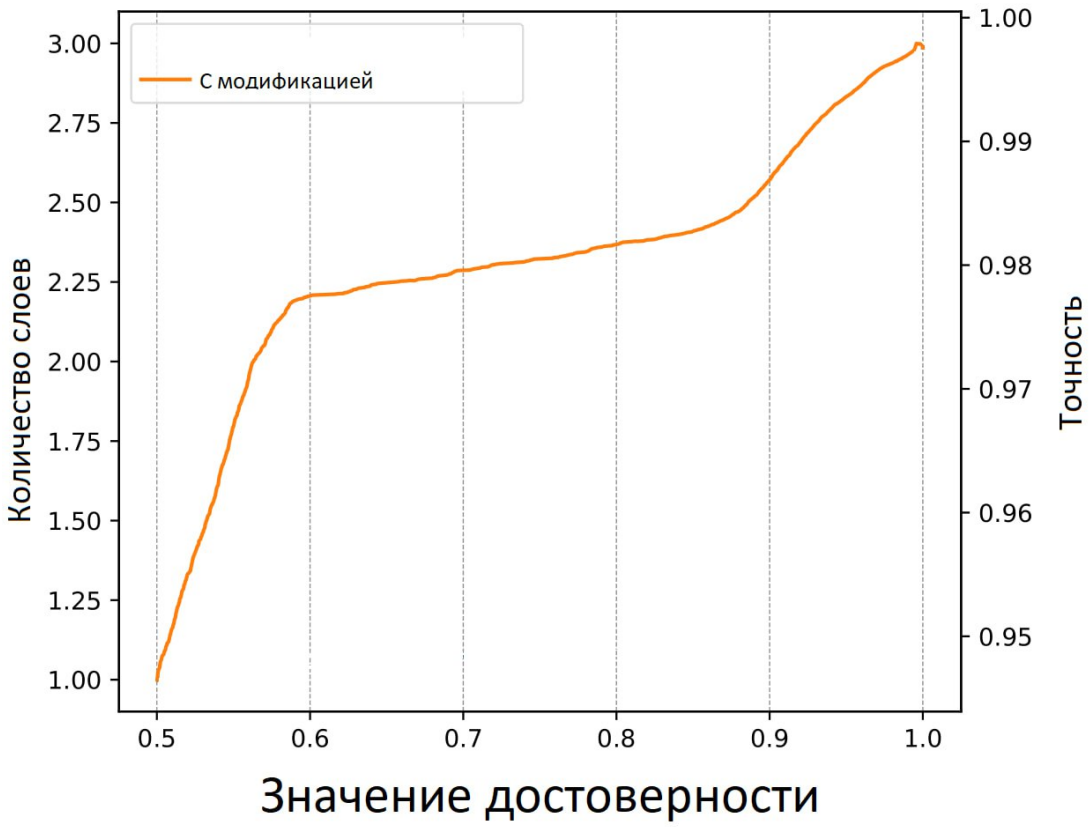

Рисунок 4.6 – 3 слоя × 128 нейронов

4.5 Оценка разработанного программного обеспечения

У разработанного программного обеспечения были выявлены следующие достоинства и недостатки.

Достоинства:

— высокая точность;

— возможность сокращения времени обучения модели за счет уменьшения количества слоев.

Недостатки:

— низкая точность предсказания следующих атак: DDoS Heartbeed, Web XSS, Web SQL injection, PortScan FW-on.

Вывод

Проанализирована зависимость точнсоти обучения модели от количества эпох. Определено требуемое количествах слоев для обнаружения атак

принадлежащих к различным классам. Приведены достоинства и недостатки разработанного программного обеспечения.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы был разработан метод обнаружения сетевых атак с использованием многослойной нейронной сети.

В результате проделанной работы были выполнены все поставленные задачи:

— описаны термины предметной области и обозначена проблема;

— описаны технологии, с помощью которых можно реализовать метод обнаружения сетевых атак;

— разработан метод обнаружения сетевых атак;

— разработан программный комплекс, реализующий интерфейс для взаимодействия с разработанным методом;

— разработанный метод исследован на предмет применимости.

Таким образом, поставленная цель работы — разработать метод обнаружения сетевых атак с использованием многослойной нейронной сети, была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Исупов А. О.* Актуальные киберугрозы: IV квартал 2022 года [Электронный ресурс] // Россия, Positive Technologies Researches. — 2022. — URL: `https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2022-q1/`.
2. *Лукацкий А. В.* Обнаружение сетевых атак // Россия, СПб, БХВ-Петербург. — 2016.
3. *Шаньгин В. Ф.* Информационная безопасность компьютерных систем и сетей // Россия, Москва, ИНФРА-М. — 2018.
4. *Seredinski F.* Anomaly detection in TCP/IP networks using immune systems paradigm // Poland, Warsaw, Computer Communications. — 2010.
5. *Hofmeyr S. A.* Architecture for an Artificial Immune System // Journal of Evolutionary Computation. — 2010.
6. *Chen W. H.* Application of SVM and ANN for intrusion detection // Computers and Operations Research. — 2011.
7. *Vaitsekhovich L.* Intrusion Detection in TCP/IP Networks Using Immune Systems Paradigm and Neural Network Detectors // XI International PhD Workshop OWD. — 2019.
8. *Battulga D.* Classification of Artificial Intelligence IDS for Smurf Attack // International Journal of Artificial Intelligence and Applications. — 2012.
9. Что такое нейронные сети? - Российская Федерация | IBM [Электронный ресурс]. — Режим доступа: `https://www.ibm.com/ru-ru/cloud/learn/neural-networks` (дата обращения 04.02.2022).
10. О линейной регрессии - Российская Федерация | IBM [Электронный ресурс]. — Режим доступа: `https://www.ibm.com/ru-ru/analytics/learn/linear-regression` (дата обращения 04.02.2022).
11. Метод опорных векторов - Университет ИТМО [Электронный ресурс]. — Режим доступа: `https://neerc.ifmo.ru/wiki/index.php?title=\T2A\CYRM\T2A\cyre\T2A\cyrt\T2A\cyro\T2A\cyrd_\T2A\cyro\T2A\cyrp\T2A\cyro\T2A\cyrr\T2A\cyrn\T2A\cyrery\T2A\cyrh_\T2A\cyrv\T2A\`

cyre\T2A\cyrk\T2A\cyrt\T2A\cyro\T2A\cyrr\T2A\cyro\T2A\cyrv_(SVM) (дата обращения 10.05.2022).

12. *Abraham A.* Distributed intrusion detection systems: a computational intelligence approach // Applications of Information Systems to Homeland Security and Defense. — 2015.

13. *Jijo B.*, *Mohsin Abdulazeez A.* Classification Based on Decision Tree Algorithm for Machine Learning // Journal of Applied Science and Technology Trends. — 2021. — Янв. — Т. 2. — С. 20—28.

14. *Komar M.* Development of Neural Network Immune Detectors for Computer Attacks Recognition and Classification // IEEE 7th Intern. Conf. on Intelligent Data Acquisition and Advanced Computing Systems. — 2013.

15. *Moustafa N.*, *Slay J.* UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set) // Military Communications and Information Systems Conference. — 2015. — Нояб. — С. 1—6.

16. Welcome to Python.org [Электронный ресурс]. — Режим доступа: https://www.python.org/ (дата обращения 18.05.2022).

17. PyTorch [Электронный ресурс]. — Режим доступа: https://pytorch.org/ (дата обращения 18.05.2022).

18. PyQt5 [Электронный ресурс]. — Режим доступа: https://pypi.org/project/PyQt5/ (дата обращения 18.05.2022).

19. Scikit-learn [Электронный ресурс]. — Режим доступа: https://scikit-learn.org/stable/ (дата обращения 18.05.2022).

20. NumPy [Электронный ресурс]. — Режим доступа: https://numpy.org (дата обращения 18.05.2022).

21. Matplotlib [Электронный ресурс]. — Режим доступа: https://matplotlib.org (дата обращения 18.05.2022).

22. pandas.DataFrame [Электронный ресурс]. — Режим доступа: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html (дата обращения 18.05.2022).

23. *Котенко И. В.* Использование многоагентных технологий для комплексной защиты информации в компьютерных сетях // Россия, Томск, Известия ТРТУ. — 2001.

24. *Уланов А. В.* Проактивные механизмы защиты от сетевых червей: подход, реализация и результаты экспериментов // Россия, Омск, Информационные технологии ОГУ. — 2014.

25. *Govindarajan M.* Intrusion Detection Using an Ensemble of Classification Methods // Proc. of the World Congress on Engineering and Computer Science. — 2012.

26. *Powers S. T.* Hybrid Artificial Immune System and Self Organising Map for Network Intrusion Detection // Information Sciences. — 2018.

27. *Sharma S. D.* Improved Algorithm for Intrusion Detection Using Genetic Algorithm and SNORT // International Journal of Emerging Technology and Advanced Engineering. — 2014.

28. *Cannady J.* The Application of Artificial Neural Networks to Misuse Detection: Initial Results // Proceedings of the 1st International Workshop on Recent Advances in Intrusion Detection. — 2008.

29. *Mirkovic J.* A Practical IP Spoofing Defense through Route-Based Fltering // International Journal of Artificial Intelligence and Applications. — 2015.

30. Automatic Classifier Selection for Non-Experts / M. Reif [и др.] // Pattern Analysis and Applications. — 2012. — Февр. — Т. 17. — DOI: 10.1007/s10044-012-0280-z.

31. *Suwanda R.*, *Syahputra Z.*, *Zamzami E.* Analysis of Euclidean Distance and Manhattan Distance in the K-Means Algorithm for Variations Number of Centroid K // Journal of Physics: Conference Series. — 2020. — Июнь. — Т. 1566. — С. 012058. — DOI: 10.1088/1742-6596/1566/1/012058.

32. 1.4. Support Vector Machines | scikit-learn 1.1.0 documentation [Электронный ресурс]. — Режим доступа: https://scikit-learn.org/stable/modules/svm.html/ (дата обращения 18.05.2022).

ПРИЛОЖЕНИЕ А

Листинг 4.1 – Модель нейронной сети

```
    class EagerNet(torch.nn.Module):
    def __init__(self, n_input, n_output, n_layers, layer_size, device):
        super(EagerNet, self).__init__()
        self.n_output = n_output
        self.n_layers = n_layers
        self.beginning = torch.nn.Linear(n_input, layer_size+n_output).to(device)
        self.middle = torch.nn.Sequential(*[torch.nn.Linear(layer_size, layer_size+n_output).to(device) for _ in range(n_layers)])
        self.end = torch.nn.Linear(layer_size, n_output).to(device)

    def forward(self, x):
        all_outputs = []
        all_xs = []
        output_beginning = self.beginning(x)
        all_xs.append(x)

        x = torch.nn.functional.leaky_relu(output_beginning[:,:-self.n_output])

        all_outputs.append(output_beginning[:,-self.n_output:])

        for current_layer in self.middle:
            current_output = current_layer(x)
            all_xs.append(x)
            x = torch.nn.functional.leaky_relu(current_output[:,:-self.n_output])
            all_outputs.append(current_output[:,-self.n_output:])

        all_xs.append(x)
        output_end = self.end(x)
        all_outputs.append(output_end)

        return all_outputs, all_xs
```

Листинг 4.2 – Функция классификации

```
def multiclass_predict(net, dataset, device, n_outputs):
    batchSize = 1
    loader = torch.utils.data.DataLoader(dataset, batch_size=batchSize, shuffle=False)

    y_pred_list = [[] for _ in range(n_outputs)]
    y_list = []
    with torch.no_grad():
        net.eval()
        for data, labels in tqdm(loader):
            X_batch = data.to(device)
            outputs, _ = net(X_batch)

            for output_index, y_test_pred in enumerate(outputs):
```

```
14                  y_pred_softmax = torch.log_softmax(y_test_pred, dim
    = 1)
15                  _, y_pred_tags = torch.max(y_pred_softmax, dim = 1)
16                  y_pred_list[output_index].append(y_pred_tags.cpu().
    numpy())
17
18              _, labels = torch.max(labels, dim = 1)
19              y_list.append(labels.cpu().numpy())
20
21
22      y_list = [a.squeeze().tolist() for a in y_list]
23      for i, output in enumerate(y_pred_list):
24              y_pred_list[i] = [a.squeeze().tolist() for a in output]
25
26      return y_pred_list[-1]
```