

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Вычислительные системы и информационная безопасность»

|  |  |  |
| --- | --- | --- |
| И.о. Зав. кафедрой | | «ВСиИБ» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | А.Р. Газизов |
| подпись | | И.О. Фамилия |
|  | « » 2024 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Тема «ДЕТЕКТИРОВАНИЕ ФИШИНГОВЫХ САЙТОВ С ПОМОЩЬЮ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ»

Направление подготовки 10.05.02 Информационная безопасность телекоммуникационных систем

Специализация Защита информации в системах связи и управления

Обозначение ВКР 10.05.02.310000.000 группа ВИБТ61

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Никитин

(подпись, дата) И.О.Ф.

Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_ доц., к.т.н. А.Ю. Полуян

(подпись, дата) должность, И.О.Ф.

Консультанты по разделам:

Безопасность и экологичность работы \_\_\_\_\_\_\_\_\_\_\_\_\_ проф., д.т.н. В.Л. Гапонов

(подпись, дата) (должность, И.О.Ф.)

Технико-экономическое обоснование \_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преп. А.В. Белоусова

(подпись, дата) (должность, И.О.Ф.)

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преп. М.А. Ганжур

(подпись, дата) (должность, И.О.Ф.)

Ростов-на-Дону

2024



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Вычислительные системы и информационная безопасность»

|  |  |  |
| --- | --- | --- |
| И.о. Зав. кафедрой | | «ВСиИБ» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | А.Р. Газизов |
| подпись | | И.О. Фамилия |
|  | « » 2024 г | |

**ЗАДАНИЕ**

на выполнение выпускной квалификационной работы

Тема «ДЕТЕКТИРОВАНИЕ ФИШИНГОВЫХ САЙТОВ С ПОМОЩЬЮ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ СЕТИ»

Обучающийся Никитин Дмитрий Валерьевич

Обозначение ВКР 10.05.02.310000.000 группа ВИБТ61

Тема утверждена приказом по ДГТУ от «11» января 2024 г. № 47-ЛС-О

Срок представления ВКР к защите «\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Исходные данные для выполнения выпускной квалификационной работы:

Язык программирования Python версии 3.12.1, фреймворк машинного обучения для языка Python с открытым исходным кодом PyTorch, кроссплатформенный браузер с открытым исходным кодом Chromium.

Содержание выпускной квалификационной работы

Введение: Во введении необходимо изложить актуальность выбранной темы, обозначить объект и предмет исследования, цель и задачи выпускной квалификационной работы, теоретическую и практическую значимость работы, структуру работы.

Наименование и содержание разделов:

1. Обзор проблемы фишинговых атак и методов борьбы с ними. Проблематика фишинговых атак и её актуальность. Анализ методов детектирования фишинговых сайтов. Законодательное регулирование и стандарты в области борьбы с фишингом. Методы фишинговых атак.

2. Анализ и выбор методов детектирования фишинговых сайтов с использованием нейронных сетей. Основы нейросетевых технологий. Обзор существующих методов детектирования фишинговых сайтов. Выбор и обоснование методов детектирования фишинговых сайтов. Определение параметров для анализа сайтов: URL, визуальной составляющей и текст.

3. Разработка и обучение нейросети для детектирования фишинговых сайтов. Формирование и предобработка набора данных. Архитектура нейронной сети и её обучение. Тестирование и анализ результатов работы нейросети. Оценка эффективности и точности детектирования фишинговых сайтов.

4. Экономическое обоснование. Спецификация проекта. План–график проектирования и разработки системы. Расчет затрат на разработку проекта.

5. Безопасность и экологичность проекта. Расчет системы зануления. Расчет электромагнитного излучения, создаваемого передающими станциями. Оценка зон теплового воздействия Выводы.

Заключение: Заключение должно содержать обобщенные результаты проведенной работы в соответствии с поставленной целью и задачами, необходимо указать чем завершается работа – усовершенствованием, модернизацией, дать свои предложения.

Перечень графического и иллюстративного материалов:

Свёрточная нейросеть

Рекуррентная нейросеть

График линейной функции активации.

График активации бинарного шага.

Схема базы данных

Схема работы клиентского и серверного приложений

Примеры вывода результатов работы приложения в клиентском приложении

Пример проверки легитимного сайта

Пример проверки малоизвестного сайта

Пример проверки фишингового сайта

Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_ доц., к.т.н. А.Ю. Полуян

(подпись, дата)

Задание к исполнению принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Никитин

**Аннотация**

Выпускная квалификационная работа акцентирует внимание на проблематике фишинговых атак и необходимости эффективного метода их детекции. Основное внимание уделено использованию нейросетевых технологий для автоматического определения подозрительных и потенциально опасных сайтов. Целью данной выпускной квалификационной работы является создание автоматической системы, способной определять подозрительные и потенциально опасные веб-сайты. Актуальность данной работы обусловлена растущим числом фишинговых атак в цифровой среде, которые представляют серьезную угрозу безопасности пользователей. В заключение работы представлены экономическое обоснование предложенного решения и анализ потенциальных рисков, а также экологическая оценка проекта.

Объем текстового материала 90 листа (А4), количество иллюстраций 13, таблиц — 12, использованных источников — 17.

**Abstract**

The graduation qualifying thesis focuses on the issue of phishing attacks and the need for an effective method of detection. The main emphasis is placed on the utilization of neural network technologies for automated identification of suspicious and potentially dangerous websites. The objective of this final qualifying work is to develop an automatic system capable of identifying suspicious and potentially harmful web pages. The relevance of this work is driven by the increasing number of phishing attacks in the digital environment, which pose a significant threat to user security. The thesis concludes with an economic justification of the proposed solution, an analysis of potential risks, and an ecological assessment of the project.

The volume of text material is 90 sheets (A4), the number of illustrations is 13, tables — 12, sources used — 17.

Содержание

[Введение 7](#_Toc157638888)

[1. Обзор проблемы фишинговых атак и методов борьбы с ними 9](#_Toc157638889)

[1.1 Проблематика фишинговых атак и её актуальность 9](#_Toc157638890)

[1.2 Анализ методов детектирования фишинговых сайтов 10](#_Toc157638891)

[1.3 Законодательное регулирование и стандарты в защите информации 11](#_Toc157638892)

[1.4 Методы фишинговых атак 12](#_Toc157638893)

[2 Анализ и выбор методов детектирования фишинговых сайтов с использованием нейронных сетей 19](#_Toc157638894)

[2.1 Основы нейросетевых технологий 19](#_Toc157638895)

[2.2 Обзор существующих методов детектирования фишинговых сайтов 26](#_Toc157638896)

[2.3 Выбор и обоснование методов детектирования фишинговых сайтов 30](#_Toc157638897)

[2.4 Определение параметров для анализа сайтов: URL, визуальной составляющей и текст 32](#_Toc157638898)

[3. Разработка и обучение нейросети для детектирования фишинговых сайтов 42](#_Toc157638899)

[3.1 Формирование и предобработка набора данных 42](#_Toc157638900)

[3.2 Архитектура нейронной сети и её обучение 44](#_Toc157638901)

[3.3 Тестирование и анализ результатов работы нейросети 50](#_Toc157638902)

[3.4 Создание клиентского и серверверного приложения 52](#_Toc157638903)

[3.5 Оценка корректности и стабильности работы итогового приложения 55](#_Toc157638904)

[4. Экономическое обоснование 59](#_Toc157638905)

[4.1 Спецификация проекта 59](#_Toc157638906)

[4.2 План–график проектирования и разработки системы 60](#_Toc157638907)

[4.3 Расчет затрат на разработку проекта 64](#_Toc157638908)

[5. Безопасность и экологичность проекта 68](#_Toc157638909)

[5.1 Расчет системы зануления 68](#_Toc157638910)

[5.2 Расчет электромагнитного излучения, создаваемого передающими станциями 70](#_Toc157638911)

[5.3 Оценка зон теплового воздействия 75](#_Toc157638912)

[Заключение 77](#_Toc157638913)

[Перечень использованных информационных ресурсов 78](#_Toc157638914)

[Приложение A 80](#_Toc157638915)

[Приложение Б 83](#_Toc157638916)

[Приложение В 85](#_Toc157638917)

[Приложение Г 89](#_Toc157638918)

**Введение**

В современном мире, где интернет занимает значительную часть жизни людей, вопрос защиты личной информации становится все более важным и сложным. Одной из серьезных угроз для безопасности пользователей является фишинг - мошенническая деятельность, направленная на получение доступа к конфиденциальным данным, таким как пароли, номера кредитных карт и другая личная информация. Для борьбы с этой проблемой необходимо разработать эффективные методы определения и классификации фишинговых сайтов, которые могут подражать известным сайтам, таким как банки, онлайн-магазины, платежные системы и другие сервисы, с целью обмануть пользователей и заставить их раскрыть свои личные данные. Классификация фишинговых сайтов позволяет выделить их характеристики и создать модель для их обнаружения.

В данной работе предлагается использовать нейросетевые технологии для обнаружения фишинговых сайтов, так как они обладают высокой способностью обучаться на больших объемах данных и выявлять скрытые закономерности, что позволяет им эффективно определять фишинговые сайты. Будет рассмотрено применение нейросетевых технологий для обнаружения фишинговых сайтов и разработка соответствующей модели. Также будет предложена архитектура системы, включающая нейросетевую модель и соответствующие алгоритмы для обнаружения фишинговых сайтов. Кроме того, к разработанной модели нейросетей будет создано клиентское приложение для упрощения работы с системой.

Актуальность исследования обусловлена растущей угрозой фишинговых атак в сети Интернет, которые могут привести к серьезным последствиям для пользователей, таким как кража личной информации, денежных средств и других ресурсов. Фишинговые атаки могут нанести большой ущерб пользователям, поскольку они могут потерять свои денежные средства, ресурсы или даже свою личность. Поэтому необходимо разработать эффективные методы для определения и классификации фишинговых сайтов, чтобы предотвратить их распространение и вред. Однако традиционные методы, основанные на анализе URL, содержимого или структуры сайтов, не всегда способны справиться с этой задачей, поскольку фишинговые сайты постоянно меняются и усовершенствуются, поэтому использование нейросетевых технологий может значительно улучшить процесс идентификации и блокировки таких сайтов.

В данной работе ставится цель разработать систему, которая может эффективно обнаруживать и классифицировать фишинговые сайты, используя современные нейросетевые технологии и клиентское приложение для удобства использования созданной системы. Для решения этой актуальной и сложной проблемы необходимо выполнить следующие задачи:

1. Провести обзор и анализ существующих методов детектирования фишинговых сайтов.
2. Провести анализ и выбор методов детектирования фишинговых сайтов с использованием нейронных сетей.
3. Сформировать и предобработать набор данных для обучения.
4. Разработать и обучить нейронные сети для детектирования фишинговых сайтов на основе полученного набора данных.
5. Протестировать и проанализировать работы нейросетей.
6. Создать клиентское и серверное приложение.
7. Оценить корректность и стабильность работы итогового приложения.

Ожидаемым результатом работы является нейросетевая модель для обнаружения фишинговых сайтов, которая показывает высокое качество и точность на тестовых данных, а также клиентское приложение, которое позволяет пользователям проверять подозрительные сайты на фишинг.

1. **Обзор проблемы фишинговых атак и методов борьбы с ними**
   1. **Проблематика фишинговых атак и её актуальность**

В современном цифровом мире проблема фишинговых атак становится все более актуальной и серьезной. Киберпреступность продолжает развиваться, и фишинговые атаки входят в число наиболее распространенных и опасных видов мошенничества в онлайн-среде.

Фишинговые атаки – это мошеннические действия, которые направлены на получение конфиденциальной информации, такой как логины, пароли, данные банковских карт, с помощью создания поддельных сайтов, электронных писем и сообщений, выдающихся за официальные.

Современный мир технологий и интернета предоставляет широкие возможности для общения, работы, обучения и отдыха. Однако, вместе с этим он также представляет серьезные угрозы для пользователей, такие как фишинговые атаки. Фишинговые атаки являются одной из наиболее распространенных угроз в сфере кибербезопасности. Они представляют собой попытки мошеннического получения конфиденциальной информации, такой как логины, пароли, номера кредитных карт и другие данные, путем подделки легитимных веб-сайтов или электронных сообщений. Фишинговые атаки могут привести к серьезным последствиям для пользователей, включая финансовые потери, утечку конфиденциальной информации и нарушение частной жизни.

Фишинговые атаки являются актуальной проблемой в мире. Согласно отчету компании «Positive Technologies» [1] от 16 июня 2023, в первом квартале 2023 года количество событий увеличилось на 7% по сравнению с предыдущим периодом и превысило показатель начала 2022 года на 10%. Самыми распространенными последствиями успешных кибератак на компании стали утечки конфиденциальных данных (51%) и нарушения основной бизнес-деятельности (44%).

Фишинговые атаки имеют серьезные последствия как для отдельных пользователей, так и для организаций. Когда пользователь становится жертвой фишинга, его конфиденциальная информация попадает в руки злоумышленников. Это может привести к утрате доступа к банковским счетам, краже личных данных, а также использованию этой информации для совершения других видов мошенничества.

Организации, в свою очередь, могут столкнуться с финансовыми убытками, утратой доверия клиентов и повреждением репутации. Фишинговые атаки могут привести к утечке коммерческой информации, потере конкурентных преимуществ и ухудшению отношений с партнерами. Все это наносит серьезный ущерб бизнесу и экономике в целом.

* 1. **Анализ методов детектирования фишинговых сайтов**

Существует множество методов детектирования фишинговых сайтов, которые используются для борьбы с этой угрозой [2]. Некоторые из них включают анализ содержимого веб-страницы, проверку URL-адреса, использование черных списков и т.д. Каждый метод имеет свои преимущества и недостатки, и выбор конкретного метода зависит от конкретных условий.

Один из наиболее распространенных подходов к детектированию фишинговых сайтов основывается на анализе URL-адреса. Этот метод заключается в том, чтобы проверять доменное имя сайта и его поддомены на наличие признаков, таких как:

1. Орфографические ошибки;
2. Использование цифр и символов вместо букв;
3. Ограниченный срок регистрации домена.

Если какой-либо из этих признаков обнаружен, сайт может быть отмечен как потенциальный фишинговый ресурс.

Другой распространенный метод анализа фишинговых сайтов заключается в исследовании содержания страницы. Этот подход использует анализ содержимого страницы, чтобы проверить, отличается ли содержание на фишинговом сайте от того, что можно найти на официальном сайте. Этот метод может быть довольно эффективным, если рассматривать фишинговые сайты, которые имитируют определенных поставщиков услуг, таких как банки или платежные системы.

Визуальный метод детектирования фишинговых сайтов основан на анализе внешнего вида веб-страницы. Он сфокусирован на обнаружении признаков, которые могут указывать на то, что сайт является поддельным. Некоторые из таких признаков могут быть:

1. Дизайн и макет: Фишинговые сайты часто имеют недостаточно качественный дизайн и макет страницы по сравнению с официальными сайтами. Они могут выглядеть грубо или иметь некорректное расположение элементов.
2. Логотипы и изображения: Фишинговые сайты могут использовать измененные логотипы или изображения, которые могут отличаться от официальных. Также они могут содержать неправильные или нереалистичные изображения.
3. Системные баннеры: Фишинговые сайты могут содержать различные системные предупреждения или баннеры, которые создают ситуацию срочности, чтобы склонить пользователей к предоставлению своих личных данных.

Визуальный метод детектирования может быть полезным для определения фишинговых сайтов, особенно для тех пользователей, у которых не хватает технических знаний для проведения более сложного анализа.

**1.3 Законодательное регулирование и стандарты в защите информации**

Законодательное регулирование и стандарты в области борьбы с фишингом в России направлены на решение проблемы кибермошенничества в цифровой среде страны. Фишинг представляет собой распространенную практику, при которой мошеннические сайты подражают легитимным для обмана пользователей и получения их конфиденциальной информации.

Российское законодательство содержит набор мер и стандартов для эффективной борьбы с фишинговыми атаками и связанными с ними преступлениями [3]. Важным правовым актом является Постановление Пленума Верховного Суда Российской Федерации № 51 от 27 декабря 2007 года [4], которое определяет ответственность за фишинг. В соответствии с этим документом, незаконный доступ к компьютерной информации или создание и распространение вредоносных программ могут быть квалифицированы по статьям 159 «Мошенничество», 272 «Неправомерный доступ к компьютерной информации» или 273 «Создание, использование и распространение вредоносных компьютерных программ» Уголовного кодекса Российской Федерации [4].

Одним из распространенных методов фишинга является использование вредоносного программного обеспечения. Злоумышленники разрабатывают и распространяют специальные программы, которые автоматически перенаправляют пользователей на фишинговые сайты при выполнении определенных действий. Такие действия подпадают под статью 273 Уголовного кодекса.

Создание фишинговых сайтов также подпадает под уголовную ответственность по российскому законодательству. Незаконное использование товарных знаков при создании фишинговых сайтов может быть квалифицировано как нарушение статьи 180 Уголовного кодекса Российской Федерации, которая регулирует «Незаконное использование товарного знака». Злоумышленники злоупотребляют чужими товарными знаками и другими средствами идентификации для обмана жертв и осуществления фишинговых атак.

## 1.4 Методы фишинговых атак

Фишинг представляет собой механизм обмана, при котором злоумышленник пытается получить доступ к конфиденциальной информации, представляясь доверенным и надежным источником. Это может включать выдачу себя за банк, онлайн-сервис или другую организацию, с целью убедить жертву предоставить свои личные данные. Фишинг может быть основан как на технических, так и на социальных методах.

Технические методы фишинга включают создание поддельных веб-сайтов, которые могут выглядеть практически идентично оригиналу. Злоумышленники могут использовать поддельные электронные письма, сообщения в социальных сетях или рекламные баннеры, чтобы перенаправить пользователей на эти поддельные сайты. Цель состоит в том, чтобы заставить жертву ввести свои личные данные, такие как логин и пароль, которые затем могут быть использованы для несанкционированного доступа к аккаунту или кражи личной информации. Кроме того, технические методы фишинга могут включать использование вредоносного программного обеспечения, которое устанавливается на компьютер жертвы при переходе по поддельной ссылке или открытии вложения. Такое ПО может перехватывать клавиатурный ввод, копировать данные из буфера обмена, делать скриншоты экрана или даже давать злоумышленникам удаленный доступ к компьютеру жертвы. [Это позволяет мошенникам получать не только пароли, но и другую конфиденциальную информацию](https://ru.malwarebytes.com/phishing/)

Социальные методы фишинга основаны на манипуляции психологическими уязвимостями жертвы. Злоумышленники могут использовать такие тактики, как создание срочности или страха, чтобы побудить жертву действовать быстро и без размышления. Это может включать предупреждения о безопасности, угрозы блокировки аккаунта или предложения о крупных выигрышах. Целью является вызвать у жертвы эмоциональные реакции, которые могут привести к необдуманному предоставлению личных данных.

Понимание технических и социальных методов фишинга является важным для защиты от таких атак. Это позволяет пользователям быть более бдительными и осторожными в отношении подозрительных запросов на предоставление личной информации. Кроме того, осведомленность об этих методах помогает разработчикам и специалистам по кибербезопасности создавать более эффективные меры защиты и системы обнаружения фишинга.

Социальная инженерия представляет собой один из наиболее разрушительных и эффективных методов фишинговых атак, в котором злоумышленники используют психологические механизмы, чтобы манипулировать эмоциями и чувствами жертв. Они искусно воспроизводят чувства, такие как сострадание, тревога, желание помочь, и неосведомленность о том, как достичь своей цели, чтобы развести жертву на неразумные и рискованные действия.

Основная цель социальной инженерии заключается в отвлечении жертвы от логического и рационального мышления, подавляя ее способность принимать разумные решения. Злоумышленники активно используют эмоциональные состояния, такие как страх, зависть, интрига, ярость, дружба, верность, честолюбие, самоотверженность, чувство долга и превосходства. Проникая в эмоциональную сферу жертвы, они вынуждают ее раскрыть свои личные данные и конфиденциальную информацию.

Примером фишинговой атаки, основанной на социальной инженерии, является предоставление жертвам вредоносных ссылок на веб-сайты через электронную почту. Злоумышленники создают иллюзию легитимности и внушают доверие, чтобы убедить жертву вводить свои конфиденциальные данные на поддельных веб-страницах.

Важно быть бдительным и осознавать такие манипуляции, чтобы избежать попадания в ловушку социальной инженерии. Обучение и повышение осведомленности пользователей являются эффективными мерами для защиты от фишинговых атак, основанных на манипуляции эмоциями.

Смишинговые атаки являются формой мошенничества, где злоумышленники используют сервисы коротких сообщений (SMS) для доставки поддельных и мошеннических сообщений. Этот вид атаки становится все более распространенным, так как люди часто доверяют сообщениям, полученным через мобильные приложения для обмена сообщениями на своих телефонах, больше, чем сообщениям, полученным по электронной почте.

Смишинговые атаки оказываются успешными, поскольку злоумышленникам легче найти номера телефонов потенциальных жертв, чем их адреса электронной почты. В отличие от электронной почты, где адреса могут быть произвольной комбинацией цифр, букв и символов, телефонные номера имеют ограниченное количество вариантов. Например, в США номера телефонов состоят из десяти цифр. Это означает, что злоумышленникам достаточно сгенерировать десять случайных цифр, чтобы достичь своей цели, в то время как связаться с человеком через адрес электронной почты может быть гораздо сложнее и требовать больше усилий

Эта уязвимость воспринимается злоумышленниками как привлекательная возможность для проведения мошеннических действий, так как многие люди не ожидают получить поддельные сообщения через SMS и более склонны доверять таким сообщениям, особенно если они выглядят легитимно. Поэтому осведомленность и бдительность пользователей являются важными факторами для предотвращения смишинговых атак.

Wi-Fi фишинг, также известный как "Evil Twin", это метод фишинга, который использует беспроводные сети. В этой атаке злоумышленник создает ложную точку доступа, которая выглядит так же, как легитимная точка доступа, и находится между клиентом и настоящей точкой доступа. Злоумышленник стремится обмануть пользователя, заставляя его подключиться к ложной точке доступа вместо настоящей. Часто злоумышленник копирует конфигурацию реальной точки доступа, чтобы создать идентичный идентификатор сети (SSID) и использовать более мощный сигнал передачи.

DE-аутентификация — это атака, при которой злоумышленник пытается прервать соединение между клиентом и точкой доступа, посылая специальные кадры DE-аутентификации. Это заставляет клиента повторно пройти процесс аутентификации, и злоумышленник может прослушивать и анализировать трафик во время этого процесса. После чего атакующий может расшифровать секретный ключ и обойти шифрование безопасности. Второй этап атаки DE-аутентификации заключается в том, чтобы заставить клиента подключиться к ложной точке доступа, чтобы злоумышленник мог прослушивать всю коммуникацию.

Карма-атака основана на активном сканировании беспроводной сети WLAN для сбора запросов от устройств пользователей и генерации поддельных ответов, чтобы устройства думали, что настоящая сеть WLAN находится поблизости. Когда устройства автоматически сканируют доступные сети, они отправляют запросы зонда для проверки наличия сети в зоне действия. Злоумышленник может использовать карма-атаку, чтобы привлечь целевое устройство к своей ложной точке доступа. Эта атака особенно опасна для клиентов, которые используют активное сканирование для аутентификации.

Фишинговые наборы представляют собой комплекты инструментов, созданные с целью облегчить злоумышленникам проведение фишинговых атак, даже в отсутствие необходимых программных навыков. Эти наборы широко доступны как бесплатно, так и за плату, и предоставляют киберпреступникам возможность создавать и распространять вредоносные веб-сайты, электронные письма и скрипты с минимальными усилиями.

Однако использование бесплатных фишинговых наборов не рекомендуется из-за потенциальных угроз для конечных пользователей. Часто такие наборы включают в себя злоумышленный код, который может быть использован для кражи и передачи личных данных пользователей разработчикам наборов. Такие действия могут привести к серьезным последствиям для жертв, включая финансовые потери и нарушение конфиденциальности.

В свете этих рисков рекомендуется оставаться бдительными и применять меры безопасности при обнаружении подозрительных веб-сайтов, электронных 13 писем или запросов на предоставление личной информации. Это включает в себя проверку подлинности отправителя, осмотр URL-адресов и использование антивирусного программного обеспечения для защиты от вредоносных программ. Поддержание информационной грамотности и обучение о практиках безопасности в сети также являются важными мерами для защиты от фишинговых атак, независимо от того, используются ли фишинговые наборы или другие методы атаки.

Техника Drive-By-Download представляет собой хитроумный метод, при котором злоумышленники эксплуатируют уязвимости веб-сайтов и браузеров, чтобы непреднамеренно загрузить и запустить вредоносные программы на уязвимые системы. Эта атака может произойти при посещении опасного веб-сайта или ответе на мошенническое электронное письмо.

Drive-By-Download может быть осуществлен различными способами. Например, злоумышленники могут внедрить злонамеренный JavaScript-код на сервер веб-сайта или использовать электронную почту для передачи вредоносных программ. При успешной эксплуатации уязвимости в браузере или на сервере, вредоносное ПО загружается и устанавливается на компьютер жертвы.

Затем загруженное вредоносное ПО может превратить компьютер в ботнет, то есть включить его в сеть зараженных устройств, которые злоумышленники могут использовать для проведения дальнейших атак. Это может привести к серьезным последствиям, таким как кража личных данных, финансовые мошенничества или даже нарушение конфиденциальности.

В свете этих угроз рекомендуется принимать меры безопасности при использовании интернета. Это включает в себя регулярное обновление программного обеспечения, включая операционную систему, браузер и антивирусные программы, чтобы устранить известные уязвимости. Кроме того, следует избегать посещения подозрительных или непроверенных веб-сайтов, а также быть осторожными при открытии вложений в электронных письмах, особенно если они приходят от неизвестных отправителей.

Spear-phishing — это изощренный вид атаки, при котором злоумышленники создают и распространяют поддельные электронные письма, содержащие вредоносные программы или ссылки на вредоносные веб-сайты. Эта хитрая техника основывается на использовании социальной инженерии и поддельных источников, чтобы убедить жертву в легитимности полученного письма.

В процессе spear-phishing атаки, злоумышленники тщательно исследуют своих потенциальных жертв, собирают информацию о них и создают электронные письма, максимально подобные легитимным сообщениям, чтобы вызвать доверие и убедить получателя открыть вложение или перейти по ссылке.

Вложения или ссылки, предоставленные в этих электронных письмах, обычно содержат скрытые вредоносные программы, такие как троянские кони или шпионское ПО, которые могут быть активированы при неправильном взаимодействии с ними. В некоторых случаях, ссылки могут вести на вредоносные веб-сайты, которые маскируются под легитимные источники, чтобы собрать личные данные жертвы или провести фишинговые атаки.

Spear-phishing атаки являются особенно опасными, поскольку они маскируются под легитимные коммуникации и представляют собой персонифицированный подход к атаке, что делает их более убедительными и успешными. Часто такие атаки направлены на организации или отдельных сотрудников с целью получения конфиденциальной информации или доступа к защищенным системам.

Для защиты от spear-phishing атак рекомендуется быть внимательными при открытии электронных писем, особенно от неизвестных отправителей. При получении подозрительного письма необходимо избегать открытия вложений или перехода по ссылкам без проверки их подлинности. Важно также обучать сотрудников организаций узнавать признаки поддельных электронных писем и предоставлять им средства для проверки подлинности сообщений.

Фишинг является одной из самых распространенных и опасных интернет-угроз, которая может привести к краже конфиденциальных данных, заражению вредоносным ПО, потере денег или репутации. Для предотвращения фишинговых атак необходимо использовать эффективные методы детектирования фишинговых сайтов.

1. **Анализ и выбор методов детектирования фишинговых сайтов с использованием нейронных сетей**

**2.1 Основы нейросетевых технологий**

Нейросетевые технологии являются одними из самых важных и перспективных областей искусственного интеллекта [6]. Они используются для решения широкого круга задач, таких как распознавание образов, классификация данных, прогнозирование и многое другое. В этой главе мы рассмотрим основы нейросетевых технологий, включая их структуру, типы, принципы работы и применения.

Существует множество типов нейронных сетей, которые классифицируются в зависимости от структуры, потока данных, используемых нейронов и их плотности, слоев и их фильтров активации глубины и прочего.

Персептрон — это алгоритм обучения с учителем, который классифицирует данные по двум категориям, поэтому он является бинарным классификатором. Концепция персептрона была впервые введена Фрэнком Розенблаттом в 1957 году. В основе лежит искусственный нейрон с регулируемыми весами и порогом. Персептрон разделяет входное пространство на две категории с помощью гиперплоскости, представленной следующим уравнением:

Персептроны могут реализовывать логические элементы, такие как И, ИЛИ или НЕ-И. Но их проблема в том, что они могут изучать только линейно разделимые задачи, такие как логическая задача И. А для нелинейных задач, таких как логическое XOR, это не работает.

Свёрточные нейросети (CNN) — это концепция, которая содержит трехмерное расположение нейронов вместо стандартного двумерного массива. Первый слой называется свёрточным. Каждый нейрон в свёрточном слое обрабатывает информацию только из небольшой части поля. Входные функции берутся в пакетном режиме как фильтр. Архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году, используется для эффективного распознавания образов. В этом случае сеть понимает образ частично и может выполнять операции несколько раз, чтобы завершить полную обработку.

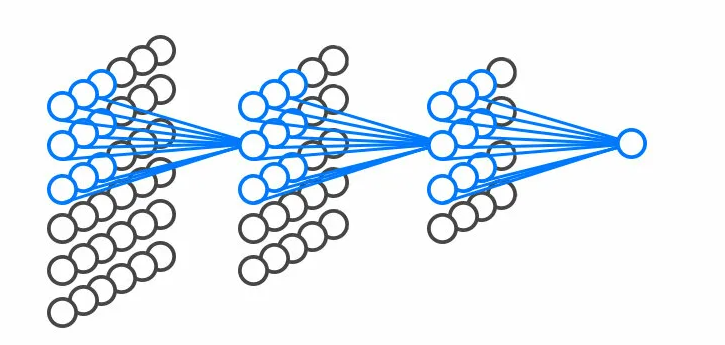


Рисунок 2.1 — Свёрточная нейросеть

Сверточные нейросети используются для разработки следующих типов приложений: обработка изображений, компьютерное зрение, распознавание речи и машинный перевод. Например, в программах обработки происходит преобразование изображения из шкалы RGB или HSI в шкалу серого. Дальнейшие изменения значения пикселя помогут обнаружить края, и изображения можно будет классифицировать по разным категориям. Эта модель используется для глубокого обучения с несколькими параметрами, которых меньше, чем в случае с полносвязным слоем. Однако, есть пара недостатков этого типа нейронных сетей, которые связаны со сложностью проектирования и обслуживания, а также с невысокой скоростью обработки.

Рекуррентные нейросети (RNN) — это достаточно распространенная архитектура, которая нашла свое применение в обработке естественного языка (NLP). Принцип работы рекуррентной нейросети основан на оценивании произвольных предложений на основании того, как часто они встречались в текстах. Такой подход дает представление о грамматической и семантической корректности. Данная модель используется в машинном переводе и для генерации новых текстов. То есть, обучаясь, например, на произведениях Ремарка, нейронная сеть сможет генерировать новый текст, похожий на Ремарка. Исходя из этого, мы понимаем, что в этой архитектуре используются прошлые выходные данные в качестве входных, имея при этом скрытые состояния.

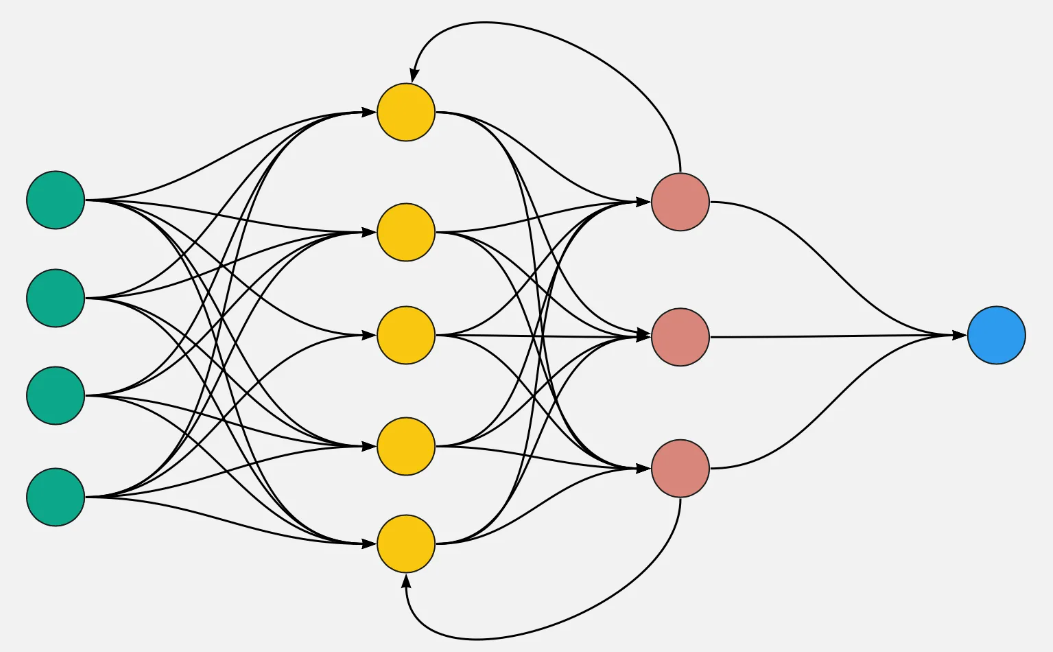


Рисунок 2.2 — Рекуррентная нейросеть

Рекуррентные нейросети используются для решения целого ряда задач, таких как обработка текста, например, проверка грамматики, преобразование текста в речь, анализ настроений. Преимущества такой архитектуры в модели последовательных данных, в которой можно предположить, что каждая выборка зависит от исторических данных. Однако обучать такую нейронную сеть крайне сложно.

Алгоритм работы нейронной сети следующий:

1. На входной слой поступают данные.
2. Синапсы передают данные на следующий слой. Каждому синапсу присвоен определенный коэффициент веса, и у каждого последующего нейрона, может быть, несколько входных синапсов.
3. Данные, которые передаются следующему нейрону, это сумма всех данных в нейронной сети, умноженных на соответствующие коэффициенты веса.
4. Полученные значения подставляются в функцию активации, что приводит к формированию выходных данных.
5. Передача данных будет продолжаться, пока она не достигнет конечного выхода.

Известно, что при первом запуске нейронной сети результаты могут быть неточными, так как сеть еще не обучена. Для обучения нейронной сети и последующей обработки данных потребуются тренировочные сеты.

Одной из особенностей нейронных сетей является способность к обучению и совершенствованию. Для этого используется среднеквадратичная ошибка потерь. Небольшое напоминание, что такое потеря: это когда вы находите способ количественно оценить усилия нейронной сети и пытаетесь ее улучшить.

В приведенной выше формуле:

N — количество входов

Y — переменная, используемая для прогноза

Ytrue — истинное значение переменной-предиктора.

Ypred — прогнозируемое значение переменной или выходных данных.

Здесь (Ytrue - Ypred)2 — квадрат ошибки. Общий квадрат ошибки можно получить с помощью функции потерь. Стоит рассматривать потери как функцию веса и чем лучше прогноз, тем меньше потери. Таким образом, цель состоит в обучении сети для минимизации потерь. Теперь у нас есть возможность изменить веса сети, чтобы повлиять на прогнозы. Отметьте каждый вес в сети, а затем запишите потери как многомерную функцию.

В свою очередь, стохастический градиентный спуск показывает, как изменить веса, чтобы минимизировать потери. Это представлено в виде уравнения:

η — это константа, известная как скорость обучения, определяющая скорость тренировки.

Выполнение уравнения обновления для каждого веса в сети уменьшит потери и улучшит сеть. В таком случае важно иметь правильный тренировочный процесс, такой как:

1. Выбор одного образца из набора данных, чтобы сделать его стохастическим градиентным спуском, работая только с одним образцом в определенное время.
2. Вычисление всех производных потерь по весам.
3. Использование уравнения обновления для обновления каждого веса.
4. Возвращаемся к шагу 1 и движемся вперед.

Когда все описанные процессы завершены, вы готовы реализовать полную нейронную сеть. Упомянутые шаги приводят к неуклонному снижению потерь и повышению точности.

Функции активации — это нелинейные преобразования, которые применяются поэлементно к входным данным. Они добавляются к искусственной нейронной сети, чтобы помочь ей изучить сложные закономерности в данных. При сравнении с моделью, основанной на нейронах, которая находится в нашем мозгу, функция активации в конечном итоге решает, что должно быть запущено для следующего нейрона.

Существует несколько типов функций активации:

1. Линейная функция активации или функция активации идентичности.
2. Двоичная ступенчатая функция или функция активации бинарного шага.
3. Нелинейные функции активации.

Линейная функция активации имеет диапазон от -∞ до ∞ и принцип работы заключается в том, что при выполнении суммируется взвешенная сумма входных данных и возвращается результат. График линейной функции активации представлен ниже:

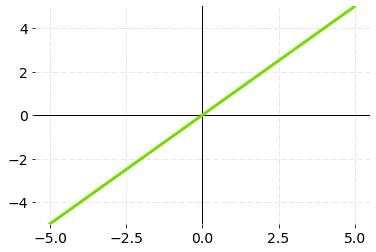


Рисунок 2.3 — График линейной функции активации.

Математически это можно представить следующим образом:

Стоит отметить, что функция активации, описанная выше, не является бинарной и ограничена диапазоном активаций. Она предполагает объединение нескольких нейронов вместе, чтобы вычислить максимальное значение при наличии нескольких активаций. Производная этой функции является постоянной величиной, которая не зависит от входного значения x.

Следующей функцией является функция активации бинарного шага, для выполнения которой важно пороговое значение. Она позволяет определить, активировать нейрон или нет. Нейрон активируется в том случае, если входное значение превышает пороговое. Но если он не активирован, то его выходные данные не передаются на следующий или скрытый слой. Ниже приведен график, иллюстрирующий функцию активации бинарного шага:

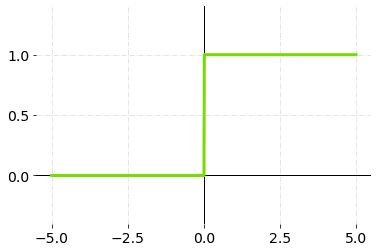


Рисунок 2.4 — График активации бинарного шага.

Математически бинарную функцию активации можно представить в виде:

Важно отметить, что представленная возможность не способна генерировать разнообразные выходные данные с несколькими значениями. Например, она не может использоваться для классификации в рамках множества классов. Градиент ступенчатой функции равен нулю, что создает некоторые трудности при обратном распространении.

Нелинейные функции активации — это самый распространенный тип, позволяющий нейронным сетям легко приспосабливаться к различным данным и разделять выходные значения. Более того, нелинейные функции активации позволяют добавлять несколько слоев нейронов, поскольку выходные данные становятся нелинейной комбинацией входных данных, проходящих через различные слои.

Функции потерь в глубоком обучении используются для измерения того, насколько хорошо работает модель нейронной сети. Внутри нейронной сети происходят две возможные математические операции: прямое и обратное распространение с градиентным спуском. Процесс прямого распространения является вычислительной процедурой, направленной на прогнозирование выходных данных для заданного входного вектора x. С другой стороны, обратное распространение и градиентный спуск представляют собой методы, описывающие процесс улучшения весов и смещений в нейронной сети с целью достижения более точных прогнозов.

Например, есть вектор x, для которого нейронная сеть предсказывает выход, называемый вектором предсказания y.

Улучшение функции потерь в нейронных сетях является ключевым аспектом для достижения высокой эффективности при выполнении различных задач, таких как регрессия или классификация. Оценка качества модели нейронной сети основывается на том, насколько точно она выполняет поставленную задачу. В процессе обратного распространения необходимо стремиться к минимизации значения функции потерь, поскольку это позволяет улучшить производительность и эффективность нейронной сети. Результаты обратного распространения будут оптимальными при минимальном значении функции потерь.

Обратное распространение представляет собой процесс обучения нейронной сети, при котором частота ошибок передается обратно через нейронную сеть для достижения более высокой точности. Обратное распространение — это процесс обучения нейронной сети, который включает в себя получение частоты ошибок прямого распространения и передачу этой потери назад через слои нейронной сети для более точной настройки весов.

**2.2 Обзор существующих методов детектирования фишинговых сайтов**

Методы обнаружения фишинговых сайтов включают различные подходы и стратегии, которые используются для выявления и идентификации потенциально вредоносных или мошеннических веб-ресурсов [7]. Эти методы могут включать анализ характеристик и признаков фишинговых сайтов, сравнение содержимого веб-страниц, использование списков с доверенными и недоверенными адресами, а также применение алгоритмов машинного обучения для классификации и обнаружения фишинговых сайтов. Каждый из этих методов имеет свои особенности и преимущества, и их комбинированное использование может значительно улучшить эффективность борьбы с фишинговыми атаками в сети.

Эвристика. Эвристические признаки могут служить индикаторами фишинговых сайтов. Одним из таких признаков является отключение правой кнопки мыши, что может указывать на попытку ограничить пользовательские действия и предотвратить доступ к контекстному меню, где обычно находятся опции, такие как "Открыть в новой вкладке" или "Просмотреть код страницы".

Еще одним эвристическим признаком является наличие символа "@" в URL-адресе, который часто используется в электронной почте для указания адреса получателя. В контексте веб-сайтов наличие символа "@" в URL-адресе может указывать на попытку злоумышленников создать иллюзию аутентичности и привлечь внимание пользователей.

Другим распространенным эвристическим признаком являются всплывающие окна, предлагающие пользователю ввести свой пароль или другую конфиденциальную информацию. Фишинговые сайты часто используют такие всплывающие окна для соблазнения пользователей на предоставление своих учетных данных, притворяясь легитимными сервисами или платформами.

Также можно упомянуть эвристический признак, связанный с IP-адресом в доменной части URL-адреса. Наличие IP-адреса вместо доменного имени может указывать на попытку скрыть истинное местоположение сайта или создать иллюзию связи с определенным сервисом или организацией.

Эти эвристические признаки, а также другие, могут быть использованы в сочетании для разработки более точных алгоритмов обнаружения фишинговых веб-сайтов. Однако стоит отметить, что эвристический подход имеет свои ограничения. Признаки, которые используются в эвристиках, могут быть подделаны или использованы некорректно, что может привести к ложным срабатываниям или пропускам фишинговых сайтов. Кроме того, злоумышленники постоянно развивают новые методы фишинга, которые могут обойти эвристические алгоритмы.

Поэтому для более эффективного обнаружения фишинговых веб-сайтов рекомендуется комбинировать эвристические методы с методами машинного обучения. Машинное обучение позволяет анализировать большие объемы данных и выявлять скрытые закономерности, что может улучшить точность обнаружения фишинговых сайтов

Визуальное сходство. В методе сравнения веб-страниц по схожему содержимому используются различные аспекты, чтобы определить степень сходства между двумя веб-страницами. Эти подходы включают в себя анализ текстового содержимого, форматирования, CSS-стилей, исходного кода, скриншотов веб-страниц, логотипов сайтов, изображений и других визуальных элементов.

Анализ текстового содержимого позволяет определить сходство между двумя веб-страницами на основе совпадающих слов, предложений или абзацев. Это может быть особенно полезно при обнаружении фишинговых сайтов, так как они могут содержать поддельные тексты, имитирующие официальные сообщения или запросы на ввод конфиденциальной информации.

Форматирование и CSS-стили также могут быть использованы для сравнения веб-страниц. Если фишинговый сайт имитирует дизайн и стиль официального сайта, то сходство в форматировании и стилях может быть заметным при сравнении двух страниц.

Исходный код веб-страницы может содержать важные признаки, которые могут быть использованы для определения сходства или различия между страницами. Это может включать скрытый JavaScript-код или ссылки на внешние ресурсы, такие как изображения или стили, которые могут быть связаны с фишинговыми сайтами.

Скриншоты веб-страницы и логотипы сайта также могут быть важными элементами для сравнения. Фишинговый сайт может использовать поддельные или модифицированные изображения, пытаясь создать иллюзию оригинальности и аутентичности.

Визуальные элементы, такие как изображения и другие графические элементы, могут быть также использованы для сравнения веб-страниц. Сходство или различия в использовании изображений могут указывать на фишинговую активность или попытку подделки.

Комбинирование и анализ всех этих аспектов позволяет определить сходство или различие между веб-страницами и помочь в обнаружении фишинговых сайтов.

Методы на основе списков. В популярных веб-браузерах, таких как Microsoft Edge, Firefox и Google Chrome, используются методы, основанные на списке, для обнаружения фишинговых сайтов. Эти методы включают использование черного списка и белого списка.

Черный список содержит список известных фишинговых сайтов, которые были идентифицированы как веб-страницы, содержащие вредоносный контент или пытающиеся манипулировать пользователями для получения их конфиденциальной информации. Браузеры используют этот черный список для блокировки доступа к таким сайтам, предупреждая пользователей о потенциальной опасности.

Белый список, напротив, содержит список доверенных веб-страниц, к которым браузеры могут безопасно получить доступ. Это может включать официальные сайты финансовых учреждений, онлайн-магазинов и других надежных источников. Браузеры используют этот белый список, чтобы проверить, что посещаемая пользователем веб-страница находится в списке доверенных сайтов, что помогает установить ее легитимность и безопасность.

Использование списков черных и белых списков в браузерах позволяет автоматически фильтровать и блокировать доступ к известным фишинговым сайтам и обеспечивать безопасное взаимодействие с проверенными и доверенными веб-страницами. Эти методы предоставляют пользователю дополнительный уровень защиты, предотвращая потенциальные угрозы, связанные с фишинговыми атаками.

Проверка текстового контента. В рамках проверки фишинговых страниц проводится анализ грамматических, синтаксических и стилистических ошибок. Создатели сайтов-подделок не обращают внимания на репутацию своих ресурсов, поэтому наличие ошибок на таких веб-ресурсах является обычным делом. Следовательно, это свидетельствует о том, что это поддельный сайт.

Кроме того, в данном методе уместно использование нейронной распознавания фишинговых сайтов. В частности, нейронные сети могут провести анализ языка, грамматики и семантики контента может помочь обнаружить подозрительные шаблоны, грамматические ошибки или недостоверную информацию, которые могут свидетельствовать о попытке фишинга. Например, нейронная сеть может обучаться на основе данных о том, какие сайты были классифицированы как фишинговые в прошлом, и использовать эту информацию для определения, является ли новый сайт фишинговым. [Однако, необходимо учитывать, что нейронные сети могут давать ложноположительные результаты, поэтому они должны использоваться в сочетании с другими методами для обеспечения максимальной эффективности](https://moluch.ru/archive/318/72550/)

Методы машинного обучения. В данном методе происходит извлечение признаков из фишинговых URL и связанных с ними веб-сайтов, а затем применяются алгоритмы машинного обучения для классификации.

Этап сбора признаков включает анализ URL-адресов фишинговых сайтов и извлечение релевантной информации, такой как длина URL, наличие поддоменов, использование символов и ключевых слов. Кроме того, производится анализ структуры веб-сайта, включая количество ссылок и степень вложенности страниц. Дополнительно извлекаются особенности JavaScript, такие как использование перенаправлений или вызовов внешних скриптов.

Собранные признаки позволяют создать набор данных, состоящий из фишинговых примеров. Затем классификаторы машинного обучения обучаются на этих данных для определения фишинговых сайтов на основе полученных признаков. Алгоритмы машинного обучения, такие как логистическая регрессия, метод опорных векторов или случайные леса, используются для создания моделей классификации.

Использование машинного обучения позволяет автоматически извлекать и анализировать признаки из фишинговых URL и веб-сайтов, а затем применять обученные модели для эффективного обнаружения фишинговых сайтов. Этот метод обеспечивает автоматизацию процесса и повышает точность обнаружения, что способствует улучшению безопасности и защите пользователей от фишинговых атак.

**2.3 Выбор и обоснование методов детектирования фишинговых сайтов**

В исследовании [8] представлена общая схема для создания воспроизводимых и расширяемых наборов данных для обнаружения фишинга веб-сайтов. Авторы использовали прикладной подход, объединяющий в себе машинное обучение, визуальное сходство и эвристику. В этом исследовании некоторые функции, основанные на содержании, не подходят для анализа во время выполнения. Никакая техника выбора признаков не использовалась, кроме ручного выбора 87 признаков, что может привести к систематической ошибке при выборе признаков. Процентное соотношение 21 набора данных Train-Test не указано. В частности, в исследовании оценивается производительность классификаторов на отдельных классах и на комбинациях классов, наилучший показатель точности 96,61% был достигнут с использованием гибридных функций и классификатора случайного леса.

В статье [9] представляют обзор методов обнаружения фишинговых атак, использующих искусственный интеллект. В статье рассмотрены методы машинного обучения, глубокого обучения, гибридного обучения и сценариев для обнаружения фишинговых атак. Авторы также представляют сравнение различных исследований, обнаруживающих фишинговые атаки для каждого метода искусственного интеллекта, и изучают качества и недостатки этих методологий. [Но](https://link.springer.com/article/10.1007/s11235-020-00733-2) авторы не анализируют исходный URL и не используют какую-либо процедуру отбора признаков. Для исследования был выбран набор данных из открытых источников и существующие алгоритмы машинного обучения. Попрежнему необходимо включать значения калибровки для каждого выбранного подхода машинного обучения. Комбинация K-Nearest Neighbours и классификатора Random Forest обнаруживает фишинговые атаки с точностью 97,33%.

В статье [10] используется прикладной подход, объединяющий в себе машинное обучение, визуальное сходство и эвристику. В статье рассматриваются три подхода для обнаружения фишинговых веб-сайтов: анализ различных функций URL, проверка законности веб-сайта путем определения его местоположения и управляющих лиц, а также анализ визуального внешнего вида веб-сайта. Авторы используют методы машинного обучения и алгоритмы для оценки этих различных функций URL и веб-сайтов. Авторы сравнивают эти подходы и изучают их качества и недостатки. Результаты показывают, что метод, основанный на анализе визуального внешнего вида веб-сайта, является наиболее эффективным. При оценке результата исследование имеет минимальное количество ложноположительных и ложноотрицательных результатов. Он обеспечивает точность обнаружения фишинговых веб-сайтов в 98,5% случаев. [Кроме того, авторы предлагают использовать гибридный подход, который объединяет все три подхода, чтобы улучшить точность обнаружения фишинговых веб-сайтов](https://www.bing.com/aclk?ld=e849UQoda2GQSLymbvRBkYGzVUCUySlMb5BDidTmNwG3tE8OLAz53GSNZ-CPT0wYW9fk_YIohqCSbUn4Ah7aJVFUDB_P-6b4cxYIzUcq1TlyEGAsexk_MZ2sEYJICvzV5pgJgNQZmfyemJaZfggyLBXwvlwTwLqbCF-QetzvNQMfm471zg&u=aHR0cHMlM2ElMmYlMmZ3d3cuemVyb2ZveC5jb20lMmZkZW1vJTJmYnJhbmQtcHJvdGVjdGlvbiUyZiUzZnV0bV9zb3VyY2UlM2RiaW5nJTI2dXRtX21lZGl1bSUzZHBhaWQtc2VhcmNoJTI2dXRtX2NhbXBhaWduJTNkcHBjLWJpbmctZXZlcmdyZWVuLWt3cy1icmFuZC1wcm90ZWN0aW9uJTI2dXRtX2NvbnRlbnQlM2RkZW1vLXJlcXVlc3QtcGhpc2hpbmclMjZ1dG1fdGVybSUzZHBoaXNoaW5nJTI1MjB3ZWJzaXRlJTI2bXNjbGtpZCUzZDBhMzBmOGZlZDk4YTE2NTk5MWMxOWQ1ZDA1NTE5ODgy&rlid=0a30f8fed98a165991c19d5d05519882).

В статье [11] представляют методы обнаружения фишинговых атак с использованием методов машинного обучения. В статье рассматриваются различные методы машинного обучения, такие как метод опорных векторов (SVM), случайный лес (Random Forest), нейронные сети и другие. Авторы также представляют сравнение различных методов машинного обучения для обнаружения фишинговых атак и изучают качества и недостатки этих методологий. Результаты показывают, что методы машинного обучения могут быть эффективными для обнаружения фишинговых атак. [Кроме того, авторы предлагают использовать гибридный подход, который объединяет различные методы машинного обучения, чтобы улучшить точность обнаружения фишинговых атак](https://powerdmarc.com/ru/check-and-report-phishing-websites/).

Проведенный анализ существующих исследований позволяет сделать вывод, что методы машинного обучения, являются эффективным инструментом для обнаружения фишинговых веб-сайтов. Они позволяют автоматически извлекать признаки из данных, обучать модели и классифицировать веб-сайты на основе обученных моделей. Это позволяет повысить точность обнаружения фишинговых веб-сайтов и улучшить общую безопасность пользователей.

Для детектирования фишинговых сайтов выбраны все представленные выше методы обнаружения. Такая техника комбинирования всех методов выбрана с целью повышения точности прогнозирования фишинга на сайте.

Все выбранные признаки будут приведены к числовому эквиваленту для того, чтобы итоговая нейросеть смогла легко воспринять результаты проведенного анализа сайта и вывести прогноз.

Для обозначения классификации сайта по тому или иному признаку была выбрана шкала значений от -1 до 1. Если веб-сайт по какому-либо признаку классифицируется как фишинговый, подозрительный или легитимный, то признак принимает значения -1, 0 или 1 соответственно.

**2.4 Определение параметров для анализа сайтов: URL, визуальной составляющей и текст**

В ходе анализа статей были выделены наиболее полезные параметры для анализа URL веб-сайта:

1. Наличие IP-адреса.

Если вместо имени домена в URL-адресе используется IP-адрес, например «http://125.98.3.123/fake.html», то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

2. Длина URL-адреса.

Злоумышленники могут использовать длинный URL-адрес, чтобы скрыть сомнительную часть в адресной строке. Например: http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=\_home&dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html

− если длина URL-адреса больше 75 символов, то URL-адрес классифицируется как фишинговый;

− если длина URL-адреса находится в диапазоне от 54 до 75 символов, то URL-адрес классифицируется как подозрительный;

− если длина URL-адреса меньше 54 символов, то URL-адрес классифицируется как легитимный.

3. Использование службы сокращения URL-адресов.

Сокращение URL-адресов — это метод во «всемирной паутине», при котором URL-адрес может быть значительно меньше по длине, но при этом вести на нужную веб-страницу. Это достигается с помощью «перенаправления HTTP» на короткое доменное имя, которое ссылается на веб-страницу с длинным URL-адресом. Например, URL-адрес «https://donstu.ru/structure/science-education/informatika-i-vychislitelnaya-tekhnika/vychislitelnye-sistemy-i-informatsionnaya-bezopasnost/» можно сократить до «bit.ly/19DXSk4»

Если в URL-адресе обнаружено использование службы сокращения URLадресов, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

4. Наличие символа «@» в URL-адресе

Использование символа «@» в URL-адресе приводит к тому, что браузер игнорирует все, что предшествует символу «@», а реальный адрес часто следует за символом «@». Если в URL-адресе обнаружено использование символа «@», то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

5. Перенаправление с использованием «//».

Наличие «//» в URL-адресе означает, что пользователь будет перенаправлен на другой веб-сайт. Пример таких URL-адресов: «http://www.legitimate.com//http://www.phishing.com».

Если URL-адрес начинается с «http», это означает, что «//» должен стоять на шестой позиции. Однако, если URL-адрес использует «https», то «//» должен появиться на седьмой позиции. Соответственно, при появлении «//» дальше, чем на седьмой позиции, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

6. Наличие символа «-» в домене URL-адреса.

Символ тире редко используется в законных URL-адресах. Злоумышленники, как правило, добавляют «-», к имени домена, чтобы пользователи чувствовали, что имеют дело с законной веб-страницей. Например «http://www.edu-donstu.ru/». Если в домене URL-адреса обнаружено использование символа «-», то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

7. Поддомен и несколько поддоменов.

− если URL-адрес имеет более одного поддомена, то URL-адрес классифицируется как подозрительный;

− если URL-адрес имеет более двух поддоменов, то URL-адрес классифицируется как фишинговый;

− если URL-адрес имеет один или меньше поддоменов, то URL-адрес классифицируется как легитимный.

8. SSL сертификат.

− если URL-адрес имеет SSL сертификат и значение Page Rank, полученное с помощью Open Page Rank API больше или равен 3, то URL-адрес классифицируется как легитимный;

− если URL-адрес имеет SSL сертификат и значение Page Rank, полученное с помощью Open Page Rank API меньше 3, то URL-адрес классифицируется как подозрительный;

− если URL-адрес не имеет SSL сертификата, то URL-адрес классифицируется как фишинговый.

9. Срок домена.

Исходя из того, что фишинговый сайт живет недолго, считается, что благонадежные домены регулярно оплачиваются на несколько лет вперед. Соответственно, если разница между датой регистрации и датой окончания действия домена URL-адреса больше или равен одному году, то URL-адрес классифицируется как легитимный, в противном случае URL-адрес классифицируется как фишинговый.

10. Фавикон.

Фавикон — это графическое изображение (иконка), связанное с определенной веб-страницей. Многие существующие пользовательские агенты, такие как графические браузеры и программы для чтения новостей, отображают значок в качестве визуального напоминания об идентичности веб-сайта в адресной строке. Если значок загружается из домена, отличного от указанного в адресной строке, URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

11. Наличие «https» в доменной части URL-адреса.

Злоумышленники могут добавить токен «https» в доменную часть URL-адреса, чтобы обмануть пользователей. Соответственно, если в доменной части URLадреса есть «https», то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

12. Внешние объекты.

На законных веб-страницах внешние объекты, содержащиеся на веб-странице, такие как изображения, видео и звуки, находятся в одном и том же домене.

Если внешних объектов, загруженных из других доменов меньше 31%, то URL-адрес классифицируется как легитимный, в противном случае URL-адрес классифицируется как фишинговый.

13. Ссылка-якорь.

Ссылка-якорь — это элемент, определяемый тегом <a>. Этот признак обрабатывается точно так же, как внешние объекты. Тем не менее, для этого признака мы проверяем:

− если ссылок с тегом <a>, ведущих на другой домен меньше 31%, то URL-адрес классифицируется как легитимный;

− если процент ссылок с тегом <a>, ведущих на другой домен меньше находится в диапазоне от 31% до 67%, то URL-адрес классифицируется как подозрительный;

− если ссылок с тегом <a>, ведущих на другой домен больше 67%, то URL-адрес классифицируется как фишинговый.

14. Теги <meta>, <script> и <link>

Законные веб-сайты часто используют теги <meta> для предоставления метаданных о HTML-документе, теги <script> для создания скрипта на стороне клиента и теги <link> для получения других веб-ресурсов.

− если ссылок с тегами <meta>, <script> и <link>, ведущих на другой домен меньше 17%, то URL-адрес классифицируется как легитимный;

− если процент ссылок с тегами<meta>, <script> и <link>, ведущих на другой домен меньше находится в диапазоне от 17% до 81%, то URL-адрес классифицируется как подозрительный;

− если ссылок с тегами <meta>, <script> и <link>, ведущих на другой домен больше 81%, то URL-адрес классифицируется как фишинговый.

15. Теги <form>.

Обработчики тегов <form>, которые содержат пустую строку или «about:blank», считаются сомнительными, поскольку в отношении представленной информации необходимо предпринять какие-либо действия. Кроме того, если доменное имя в SFH отличается от доменного имени вебстраницы, это показывает, что веб-страница является подозрительной, поскольку представленная информация редко обрабатывается внешними доменами.

− если обработчики тегов <form>, содержат пустую строку или «about:blank», то URL-адрес классифицируется как фишинговый;

− если доменное имя в обработчике отличается от доменного имени веб-страницы, URL-адрес классифицируется как подозрительный;

− если первые два условия не выполняются, то URL-адрес классифицируется как легитимный.

16. Ссылка «mailto:».

Веб-форма позволяет пользователю отправлять свою личную информацию, которая направляется на сервер для обработки. Злоумышленник может перенаправить информацию пользователя на его личную электронную почту. С этой целью можно использовать в теге <a> ссылку «mailto:», например:

<a href=mailto:dnikitin@donstu.ru&body=привет>[Напишите нам](mailto:mail@htmlacademy.ru&body=привет)</a>

Если в теге <a> используется ссылка «mailto:», то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

17. Переадресация веб-сайта.

Грань, которая отличает фишинговые веб-сайты от законных, заключается в том, сколько раз веб-сайт был перенаправлен. В среднем фишинговые сайты перенаправляются 4 раза, а легитимные сайты лишь 1 раз.

− если количество переадресаций веб-страницы больше одного, то URL-адрес классифицируется как подозрительный; − если количество переадресаций веб-страницы равно одному, либо переадресации отсутствуют, то URL-адрес классифицируется как легитимный.

18. Настройка строки состояния.

Фишеры могут использовать JavaScript, чтобы показать пользователям поддельный URL-адрес в строке состояния. Чтобы извлечь эту функцию, мы должны получить из исходного исходного кода веб-страницы событие «onmouseover», и проверить, вносит ли оно какие-либо изменения в строку состояния.

Если событие «onmouseover» какие-либо изменения в строку состояния веб-страницы, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

19. Отключение правой кнопки мыши.

Фишеры могут использовать JavaScript, чтобы отключить функцию щелчка правой кнопкой мыши, чтобы пользователи не могли просматривать и сохранять исходный код веб-страницы. Чтобы извлечь эту функцию, мы должны получить из исходного исходного кода веб-страницы событие «event.button==2».

Если событие «event.button==2» найдено, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

20. Перенаправление Iframe.

IFrame — это тег HTML, используемый для отображения дополнительной веб-страницы на той, которая отображается в данный момент. Фишеры могут использовать тег «iframe» и сделать его невидимым, то есть без границ фрейма. В связи с этим фишеры используют атрибут «frameBorder», который заставляет браузер визуализировать границы.

Если атрибут «frameBorder» равен 0 или ‘no’, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

21. Возраст домена.

Большинство фишинговых сайтов существуют в течение короткого периода времени. В среднем минимальный возраст законного домена составляет 6 месяцев.

Если возраст домена составляет меньше 6 месяцев, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

22. Запись DNS.

Для фишинговых веб-сайтов либо заявленная личность не распознается базой данных WHOIS, либо для имени хоста не создаются записи.

Если запись DNS пуста или не найдена, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

23. Трафик веб-сайта.

Эта функция измеряет популярность веб-сайта, определяя количество посетителей и количество посещенных ими страниц. В худшем случае, rank законных доменов в similarweb был менее 100000. Если домен не имеет трафика или не распознается базой данных similarweb, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

− если домен не имеет трафика или не распознается базой данных similarweb, то URL-адрес классифицируется как фишинговый;

− если rank similarweb для домена URL-адреа <= 100000, URL-адрес классифицируется как легитимный;

− если rank similarweb для домена URL-адреа > 100000, URL-адрес классифицируется как подозрительный.

24. Page Rank.

Page Rank — это значение от «0» до «10». Page Rank направлен на измерение того, насколько важна веб-страница в Интернете. Чем больше значение Page Rank, тем важнее веб-страница. Около 95% фишинговых вебстраниц имеют Page Rank 0. Оставшиеся 5% фишинговых веб-страниц могут достигать значения PageRank до 3.

Если Page Rank для сайта меньше 3, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

25. Индекс Google.

Когда сайт индексируется Google, он отображается в результатах поиска. Обычно фишинговые веб-страницы доступны только в течение короткого периода времени, и в результате многие фишинговые веб-страницы могут быть недоступны в индексе Google.

Если сайт индексируется Google, то URL-адрес классифицируется как легитимный, в противном случае URL-адрес классифицируется как фишинговый.

26. Наличие домена в известных фишинговых базах.

Если домен проверяемого сайта содержится в известных фишинговых базах, таких как PhishTank, OpenPhish, Cert.pl, PhishFindR, Urlscan.io и Phishunt.io, то URL-адрес классифицируется как фишинговый, в противном случае URL-адрес классифицируется как легитимный.

27. Грамматические ошибки

Так как фишинговые сайты могут содержать много грамматических ошибок, так как они часто создаются не профессиональными разработчиками, введен этот критерий

− если в тексте веб-сайта не найдено грамматических ошибок, то он классифицируется как легитимный;

− если в тексте веб-сайта найдено от 1 до 5 грамматических ошибок, то он классифицируется как подозрительный;

− если в тексте веб-сайта найдено от 5 грамматических ошибок, то он классифицируется как фишинговый;

28. Решение по визуальной составляющей

Визуальный метод основан на анализе внешнего вида веб-страницы. Фишинговые сайты могут имитировать настоящие сайты, но они могут содержать некоторые отличия, такие как неправильное написание доменного имени или отсутствие SSL-сертификата. Визуальный метод может помочь распознать эти отличия. А для использования визуального метода обнаружения, будем использовать свёрточную нейронную сеть.

Свёрточная нейронная сеть являются одним из наиболее эффективных методов обнаружения фишинговых сайтов. Они используются для классификации веб-страниц на легитимные и фишинговые. Она использована для проверки веб-сайтов на фишинг, анализируя их визуальные элементы, такие как логотипы, шрифты и цвета.

Решение обученной модели свёрточной сети будет считаться еще одним параметром для детектирования фишинговых сайтов.

29. Решение текстовой нейронной сети

Этот метод основан на анализе текстового контента на веб-сайте, а сам анализ будет производить нейронная сеть. Она будет реагировать на различные признаки, которые могут указывать на то, что сайт является фишинговым. Например, нейронная сеть может учитывать такие факторы, как наличие ссылок на ненадежные источники, использование слов и фраз, которые часто используются в фишинговых статьях, а также наличие ошибок в тексте. Так же она учитывает такие факторы, как стиль написания текста, использование слов и фраз, которые часто используются в фишинговых статьях, а также наличие ошибок в тексте.

Решение обученной модели текстовой нейронной сети будет считаться еще одним параметром для детектирования фишинговых сайтов.

**3.** **Разработка и обучение нейросети для детектирования фишинговых сайтов**

**3.1** **Формирование и предобработка набора данных**

Для обучения модели классификатора необходим набор данных, содержащий фишинговые и легитимные веб-сайты. А сам набор данных должен представлять собой связанные данные из URL веб-сайта, текстового контента, представленного на веб-сайте и скриншота веб-интерфейса выбранного ресурса.

В случае с фишинговыми веб-сайтами, как как в большинстве своем они живут недолго, было важно собрать URL-адреса рабочих фишинговых веб-сайтов. Для этого в открытом доступе был найден веб-ресурс, содержащий постоянно обновляющуюся базу фишинговых страниц.

Веб-ресурс [phishtank.org [12] является коллективной площадкой для обмена данными и информацией о фишинге в Интернете](https://phishtank.org/). [Пользователи могут отправлять подозрительные сайты, отслеживать статус своих подач, проверять подачи других пользователей и разрабатывать программное обеспечение с помощью бесплатного API](https://phishtank.org/).

[PhishTank был запущен в октябре 2006 года и предлагает систему верификации фишинга на основе сообщества, где пользователи отправляют подозрительные веб-сайты, и другие пользователи голосуют, является ли это фишингом или нет](https://en.wikipedia.org/wiki/PhishTank" \t "_blank).

Таким образом, PhishTank — это надежный и бесплатный ресурс, который помогает бороться с фишингом в Интернете.

На момент обращения к веб-ресурсу PhishTank его база данных имеет 39346 уникальных записей с URL фишинговых сайтов. Для создания собственной базы данных возьмём за основу данные с PhishTank.

Так как полученный набор данных хранит записи только о фишинговых сайтах и этого недостаточно для обучения нейросетей, было решено дополнить набор актуальными данными легитимных сайтах.

Перед началом сбора данных необходимо спроектировать и создать базу данных, в которую будет вестись запись. В качестве системы управления базами данных была выбрана SQLite за свою простоту установки и использования.

Для сбора данных было спроектировано две таблицы: таблица domains и таблица website\_data. Таблица domains была создана для хранения в себе всех уникальных доменов в ходе работ и для сохранения связности записей. Таблица website\_data будет хранить в себе записи URL, текстового контента и скриншота сайта.

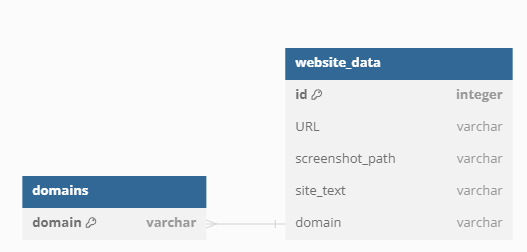


Рисунок 3.1 — Схема базы данных

Для получения необходимого набора данных веб-сайтов, был составлен скрипт на языке программирования Python c использованием веб драйвера Selenium который представлен в приложении А.

Данный скрипт автоматически сканирует веб-ресурсы, которые подаются ему в очереди. Он создает для каждой отсканированной страницы папку с присвоенным ей ID и сохраняет в нее скриншот в формате .png и текстовой контент в .txt-файле, а затем делает запись в базу данных в формате ID, полный путь до скриншота, текстовой контент.

Следует отметить, что предусмотрено автоматическое заполнение очереди найденными на веб-ресурсе ссылками, которые еще не были просканированы. Это означает, что система автоматически добавляет новые ссылки в очередь, если они еще не были просканированы. Для того, чтобы обеспечить более эффективное сканирование веб-ресурсов, необходимо использовать специальные алгоритмы и методы. Одним из таких методов является алгоритм BFS (Breadth-First Search), который позволяет сканировать веб-ресурсы в ширину. Этот алгоритм позволяет обойти все страницы веб-сайта, начиная с главной страницы и переходя по всем ссылкам, которые находятся на каждой странице. Также для более эффективного сканирования веб-ресурсов можно использовать алгоритм DFS (Depth-First Search), который позволяет сканировать веб-ресурсы в глубину. Этот алгоритм позволяет обойти все страницы веб-сайта, начиная с главной страницы и переходя по ссылкам, которые находятся на каждой странице, пока не будут просканированы все страницы. В данном скрипте они используются алгоритмы BFS и DFS применены в комбинированном формате, так как в целях обучения нейронной сети нас интересует, как и количество просканированных веб-ресурсов, так и глубина их сканирования.

При работе с сетью нельзя забывать об ошибках, которые могут возникнуть в процессе работы. Возможно, что скрипт попробует обратиться к заблокированным на территории Российской Федерации ресурсам на что получит ошибку сети. Скрипт обрабатывает такие случаи и сохраняет все URL сайтов, к которым не смог получить доступ в отдельный список для дальнейшего анализа.

Так же скрипт написан в асинхронном стиле, что значительно повышает скорость его работы, так как большую часть его времени занимает ожидание ответа от веб-сервисов.

В результате непрерывной работы данного скрипта было получено 82746 успешно отсканированных веб-ресурсов, 35381 сайтов к которые не удалось просканировать и 7373494 веб-адресов в очереди.

**3.2** **Архитектура нейронной сети и её обучение**

Нейронные сети являются одним из наиболее эффективных инструментов в области машинного обучения и искусственного интеллекта. Они способны решать сложные задачи, такие как распознавание образов, обработка естественного языка и прогнозирование временных рядов. В настоящее время существует множество фреймворков для построения нейронных сетей, среди которых наиболее популярными являются Keras и PyTorch.

Keras — это высокоуровневый фреймворк для быстрой разработки и обучения нейронных сетей. Он построен на основе библиотеки TensorFlow и предоставляет простой и интуитивно понятный API для создания сложных моделей. К достоинствам Keras можно отнести:

1. Простота использования и быстрота разработки;
2. Поддержка многоуровневой архитектуры, что позволяет легко создавать сложные модели;
3. Большое количество готовых к использованию моделей и функций.

PyTorch — это еще один популярный фреймворк для построения нейронных сетей, который был разработан Facebook Research. Он предоставляет гибкий и простой API для создания и обучения моделей. К достоинствам PyTorch можно отнести:

1. Гибкая и простая в использовании архитектура;
2. Поддержка динамической сетевой архитектуры, что позволяет легко изменять структуру сети во время обучения;
3. Встроенная поддержка расчетов на видеокарте, что значительно повышает скорость обучения;
4. Большое количество готовых к использованию моделей и функций.

В данной работе был выбран фреймворк PyTorch для построения всех нейронных сетей, так как он позволяет производить расчеты на видеокарте, что значительно повышает скорость обучения, что очень важно при ведении расчётов на малых мощностях.

Анализ текстового контента является важной задачей в процессе детектирования фишинговых сайтов. Была разработана архитектура   
нейронной сети, предназначенной для анализа текстового контента. Сеть состоит из нескольких слоёв. Первый слой — это слой внедрения, который преобразует каждое слово в вектор фиксированной длины. Это позволяет модели работать с текстовыми данными, которые могут иметь различную длину. Следующий слой выполняет свертку над векторами слов. Он использует несколько фильтров, чтобы извлечь признаки из векторов слов. Затем следует слой, который уменьшает размерность выходных данных, выбирая максимальное значение из каждого окна. Следующий слой преобразует многомерный тензор в одномерный тензор. Наконец, последний слой выполняет линейную регрессию на одномерном тензоре, чтобы получить итоговый результат.

Подача данных происходит из заранее подготовленных csv-файлов, которые обрабатываются с помощью функций. Скриптом предусмотрены функции обрезания текста до заданной длины, чтобы избежать слишком длинных текстов, которые могут значительно замедлить обучение модели и очищения от некорректных данных, например от веб-ресурсов без текстового контента.

Таким образом создана модель нейросети для классификации текстовых данных. Так как в данном случае классификация происходит на фишинговые и легитимные сайты, в архитектуре нейросети используется классификатор сигмоида, который выполняет бинарную классификацию. Листинг модели указан в приложении Б.

Для обучения модели нейронной сети для анализа текстового контента необходимо подать датасет, который содержит примеры фишинговых и легитимных сайтов с соответствующими метками классов. Такой датасет необходимо разделить на а обучающую, валидационную и тестовую выборки. Обучающая выборка используется для настройки параметров модели, валидационная выборка для оценки качества модели и подбора гиперпараметров, а тестовая выборка для финальной проверки модели на новых данных.

Кроме анализа текстового контента, для детектирования фишинговых сайтов также можно использовать анализ визуального контента, то есть скриншотов сайтов. Для этого нужно применить сверточную нейронную сеть (СНС), которая способна извлекать иерархические признаки из изображений и классифицировать их на фишинговые и легитимные.

Сверточные нейронные сети представляют собой один из наиболее эффективных и широко применяемых типов архитектур в области машинного обучения и компьютерного зрения. Они служат мощным инструментом для анализа и обработки изображений, распознавания образов, классификации и сегментации объектов. СНС основываются на принципе имитации работы человеческого зрения, при котором изображение декомпозируется на малые фрагменты — пиксели, и каждый фрагмент рассматривается индивидуально. Это позволяет модели эксплуатировать коррелированные признаки для идентификации объектов и построения более сложных моделей.

Сверточная нейронная сеть состоит из нескольких типов слоев: входного, сверточных, пулинговых и полносвязный.

Входной слой представляет собой матрицу, кодирующую обрабатываемое изображение или последовательность. Этот слой адаптирует входные данные к формату, пригодному для обработки.

Сверточный слой является ключевым компонентом. Он состоит из нескольких фильтров (ядер), которые применяются к входному изображению. Фильтр осуществляет сканирование изображения с определенным шагом и детектирует особенности, такие как контуры, текстуры или цвета. Каждый фильтр генерирует карту признаков, которая представляет собой выход сверточного слоя.

Следующим шагом является слой пулинга (субдискретизации), который редуцирует размерность карты признаков, сохраняя наиболее существенные характеристики. Это способствует упрощению обработки данных и снижению количества параметров в сети. Наиболее часто используемые методы пулинга — это средний пулинг и максимальный пулинг.

Завершающим слоем в простой является полносвязный слой. Он трансформирует выходные данные сверточного и пулинг слоев в вектор. Этот вектор служит для решения задачи классификации. Этот слой использует функцию активации, такую как сигмоида, для получения вероятностей принадлежности к классам.

Структура такой сверточной нейронной сети имеет ряд преимуществ: она способна обрабатывать изображения большого размера, автоматически извлекать признаки без ручной настройки и эффективно работать с набором данных необходимого нам размера.

Для обучения сверточной нейронной сети для анализа визуального контента также необходимо подать датасет, который содержит скриншоты фишинговых и легитимных сайтов с соответствующими метками классов. Скриншоты сайтов должны быть сохранены в формате изображений png-файлов. Также необходимо разделить датасет на обучающую, валидационную и тестовую выборки, как и в случае с текстовым контентом.

Нейронная сеть на начальном этапе является необученной (ненастроенной). Обучение заключается в последовательном предъявлении образов из обучающего набора на вход нейросети, сравнении полученного ответа с желаемым выходом, в нашем случае это 1 – образ соответствует легитимному веб-ресурсу, минус 1 – образ соответствует фишинговому веб-ресурсу, и вычислении разницы между ожидаемым ответом и полученным, которая является значением функции ошибки (дельты ошибки).

Цель обучения нейронной сети состоит в минимизации функции ошибки путем корректировки весовых коэффициентов синаптических связей между нейронами. Функция ошибки определяется как разность между полученным ответом и желаемым. Например, если на вход подан образ легитимного сайта, и выход нейросети равен 0.73, а желаемый результат 1 (т.к. это образ легитимного сайта), то ошибка сети равна 0.27. Затем веса связей выходного слоя нейронов настраиваются в соответствии с ошибкой. Для нейронов выходного слоя известны их фактические и желаемые значения выходов. Поэтому настройка весов связей для таких нейронов является относительно простой.

Таким образом, сверточная нейронная сеть является эффективным инструментом для классификации визуального контента, такого как скриншоты сайтов. Она может быть использована в сочетании с нейронной сетью для анализа текстового контента, чтобы повысить точность детектирования фишинговых сайтов. Листинг модели указан в приложении В.

Оркестирующая нейросеть — это метамодель, которая координирует работу нескольких нейросетей, выполняющих разные задачи, и агрегирует их результаты для получения окончательного решения. В нашем случае оркестирующая нейросеть состоит из трех компонентов: эвристического анализатора, текстового анализатора и сверточного анализатора. Оркестирующая нейросеть агрегирует результаты этих трех анализаторов с помощью метода голосования и выдает окончательный вердикт о том, является ли веб-страница фишинговой или нет. Оркестирующая нейросеть позволяет адаптироваться к изменениям в поведении злоумышленников и новым тенденциям в фишинге, используя обновляемую базу данных известных фишинговых веб-ресурсов, отслеживая широкий диапазон параметров фишиговых сайтов и обучаясь на новых примерах фишинговых и легитимных сайтов.

Оркестирующая нейросеть принимает на вход вектор из 29 флагов, которые указывают на подозрительность сайта, собранных в ходе эвристического анализа и ответов от текстовой и свёрточной нейросетей. Затем она вычисляет вероятность того, что веб-страница является фишинговой. Оркестрирующая нейросеть имеет следующую архитектуру: входной слой, который принимает вектор из 29 значений, скрытый слой, который содержит 64 нейрона, и выходной слой, который содержит один нейрон с функцией активации сигмоида. Листинг модели указан в приложении Г.

Оркестрирующая нейросеть обучается по методу обучения с учителем, то есть на основе обучающего набора данных, который содержит примеры веб-страниц с соответствующими метками, указывающими на их легитимность или фишинговость. Оркестрирующая нейросеть получает на вектор, соответствующих вероятностям того, что веб-страница является фишинговой, по каждому из параметров, и выдает на выход одно значение, соответствующее окончательной вероятности того, что веб-страница является фишинговой.

Обучение оркестрирующей нейросети схоже с обучением сверточной, однако она имеет свои особенности, такие как:

1. Работа с несколькими источниками информации, которые могут быть неоднородными и противоречивыми, и должна уметь согласовывать их между собой;

2. Учёт не только абсолютных значений вероятностей, но и их относительных весов и достоверности, в зависимости от качества и количества данных, на которых обучались анализаторы;

3. Способность адаптирования к изменениям в поведении злоумышленников и новым тенденциям в фишинге, используя обновляемую базу данных известных фишинговых веб-ресурсов и обучаясь на новых примерах фишинговых и легитимных сайтов.

Таким образом была создана архитектура нейронной сети, которая способна обнаруживать фишинговые сайты, используя комбинацию различных подходов и источников информации. Текстовый анализатор применяет нейронную сеть с механизмом внимания к текстовому контенту веб-страницы, чтобы определить фишинговые паттерны, такие как запросы личных данных, предложения слишком выгодных сделок, угрозы и т.д. Сверточный анализатор использует сверточную нейронную сеть для сравнения визуального дизайна веб-страницы с известными фишинговыми веб-ресурсами, чтобы обнаружить попытки подражания или маскировки. Оркестрирующая нейросеть агрегирует результаты этих двух анализаторов с помощью метода голосования и выдает окончательный вердикт о том, является ли веб-страница фишинговой или нет. Так же было проведено обучение созданной модели, заключающееся в методе обучения с учителем, то есть на основе обучающего набора данных, который содержит примеры веб-страниц с соответствующими метками, указывающими на легитимность.

## 3.3 Тестирование и анализ результатов работы нейросети

Цель тестирования — сравнить выходные данные нейронной сети с целями в независимом наборе (экземплярах тестирования). При тестировании необходимо обратить внимание, что методы тестирования зависят от типа проекта (аппроксимация или классификация). В нашем случае все нейросети относятся к классифицирующим нейросетям, поэтому методы тестирования для всех нейросетей проекта будут одинаковы.

Прежде чем нейронная сеть сможет перейти к фазе развертывания необходимо удостовериться, что все показатели тестирования считаются удовлетворительными.

[Для оценки качества обучнных нейросетей используем метод Confusion matrix](https://shard-copywriting.ru/kak-pisat-nauchnyiy-tekst-v-kopiraytinge-nauchnyiy-stil-primeryi-i-rekomendatsii/) [13], который позволяет вычислить различные метрики, такие как точность, полнота, специфичность и F1-мера. Confusion matrix — это метод тестировки нейросетей решающих задачу классификации, основанный на таблице, в которой строки соответствуют истинным меткам классов, а столбцы - предсказанным меткам. На диагонали таблицы находятся верно классифицированные примеры, а вне диагонали - ошибочно классифицированные. На основе этой таблицы можно рассчитать следующие метрики:

1. Точность (accuracy) — это доля верно классифицированных примеров от общего числа примеров. Она вычисляется по формуле:

где TP - истинно положительные (true positives), TN - истинно отрицательные (true negatives), FP - ложно положительные (false positives), FN – ложно отрицательные (false negatives).

1. Полнота (recall) - это доля верно классифицированных положительных примеров от общего числа положительных примеров. Она вычисляется по формуле:
2. Специфичность (specificity) - это доля верно классифицированных отрицательных примеров от общего числа отрицательных примеров. Она вычисляется по формуле:
3. F1-мера (F1-score) — это среднее гармоническое между точностью и полнотой. Она вычисляется по формуле:

где precision — это доля верно классифицированных положительных примеров от общего числа предсказанных положительных примеров. Она вычисляется по формуле:

Тесты были проведены на выборке из 2 000 сайтов, из которых 1 000 — фишинговые, а 1 000 — легитимные. Результаты наших экспериментов представлены в таблице ниже.

Таблица 3.1 — Результаты теста по методу Confusion matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Модель | Точность | Полнота | Специфичность | F1-мера |
| Текстовая нейросеть | 0.92 | 0.94 | 0.90 | 0.93 |
| Свёрточная нейросеть | 0.93 | 0.95 | 0.91 | 0.93 |
| Оркестрирующая нейросеть | 0.95 | 0.96 | 0.94 | 0.95 |

Все три модели показывают высокие значения точности, полноты, специфичности и F1-меры, что говорит о том, что они хорошо справляются с задачей обнаружения фишинговых сайтов. Оркестрирующая нейросеть имеет самые высокие показатели по всем метрикам, что подтверждает ее преимущество перед текстовой и свёрточной нейросетями. Она учитывает различные источники информации и выдает более уверенный и точный вердикт. Текстовая и свёрточная нейросети имеют почти одинаковые результаты, что свидетельствует о том, что текстовый и визуальный контент сайтов имеют примерно одинаковую значимость для их классификации. Однако, текстовая нейросеть немного превосходит свёрточную по полноте, что означает, что она лучше находит фишинговые сайты среди всех сайтов. Свёрточная нейросеть немного превосходит текстовую по специфичности, что означает, что она лучше отличает легитимные сайты от фишинговых.

**3.4 Создание клиентского и серверверного приложения**

Для удобства использования созданной системы нейросетей были разработаны клиентское и серверное приложения. Клиентское приложение представляет собой расширение для браузера на движке хромиум, которое позволяет пользователю проверять URL на фишинг в режиме реального времени. Расширение работает следующим образом: при посещении любого сайта расширение отправляет URL этого сайта на серверное приложение через защищенное соединение. Серверное приложение представляет собой веб-сервис, который принимает запросы от клиентского приложения и возвращает результаты детектирования фишинга с помощью нейросети. Веб-сервис реализован с использованием фреймворка Flask и библиотеки PyTorch для работы с нейросетью. Если серверное приложение определяет, что URL является фишинговым, то оно отправляет обратно клиентскому приложению сигнал об этом. Клиентское приложение, получив сигнал, показывает пользователю предупреждение о том, что сайт является фишинговым. Схема работы клиентского и серверного приложений представлена на рисунке 3.2.

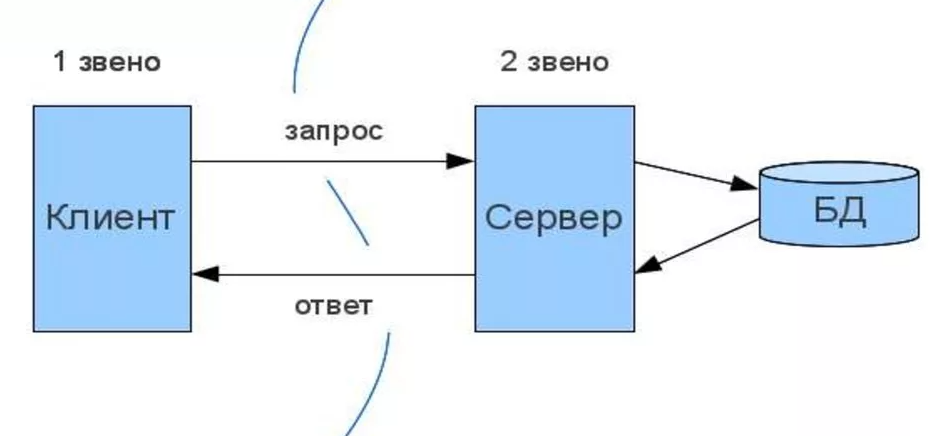


Рисунок 3.2 — Схема работы клиентского и серверного приложений

Клиентское приложение — это расширение для браузера на движке хромиум, которое в теории можно установить из магазина расширений Google Chrome, если произвести публикацию данного расширения в официальный магазин, но на данный момент оно устанавливается на прямую, путем переноса файлов расширения в меню расширений при включенном режиме разработчика. Расширение имеет простой и интуитивный интерфейс, который состоит из иконки расширения в панели инструментов браузера и всплывающего окна, которое открывается при нажатии на иконку. Иконка расширения меняет свой цвет в зависимости от того, является ли текущий сайт фишинговым или нет: зеленый цвет означает, что сайт безопасен, красный цвет означает, что сайт опасен, а оранжевый цвет означает, что сайт подозрителен. Красный индикатор присваивается веб-ресурсам, которые система посчитала фишинговыми со значением вероятности от 65%, желтый индикатор присваивается при вероятности фишинга в диапазоне от 35% до 65%, а зеленый индикатор присваивается при вероятности фишинга до 35%. Всплывающее окно показывает лаконичную информацию о результате проверки сайта, такую как вероятность того, что сайт является фишинговым, цветовой индикатор и иконку, помогающую быстро понять результат проверки. Примеры результатов проведения проверок показаны на рисунке 3.3.

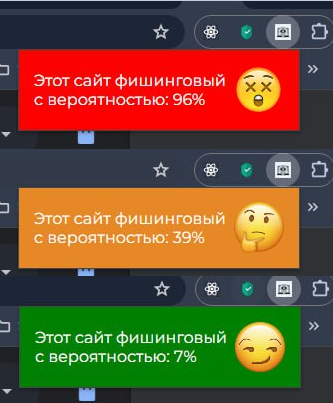


Рисунок 3.3 — Примеры вывода результатов работы приложения в клиентском приложении

Серверное приложение — это простой веб-сервис, который связывает клиентское приложение с обученными моделями нейросетей. Веб-сервис имеет REST API, которое позволяет принимать HTTP-запросы от клиентского приложения и возвращать HTTP-ответы с результатами детектирования фишинга. Веб-сервис написан на языке Python и использует фреймворк Flask для реализации веб-приложения и библиотеку PyTorch для работы с загрузкой моделей нейросети. Веб-сервис также использует базу данных SQLite для хранения набора данных. Работа с серверным приложением достаточна проста и заключается в отправке POST-запроса с json-объектом с ссылкой на проверяемый сайт в теле запроса по маршруту /api/rate, ответом на такой запрос является json-объект с вероятностью фишинга проверяемого сайта.

**3.5 Оценка корректности и стабильности работы итогового приложения**

Целью данной главы является демонстрация того, что клиентское приложение, разработанное в рамках данной работы, работает корректно и надёжно, то есть способно проверять URL на фишинг в режиме реального времени, показывать пользователю результаты проверки, а также не давать ложных срабатываний и меняющихся ответов для одного и того же объекта анализа. Для достижения этой цели будем использовать тестирование по методу чёрного ящика, так как до этого были проведены тесты моделей нейросетей, то есть тест по методу белого ящика.

Метод черного ящика, то есть тестирование приложения без учета его внутренней структуры и логики. Этот метод позволяет проверить, что приложение работает правильно с точки зрения пользователя, то есть выполняет свои функции, удовлетворяет требованиям и характеристикам, а также не имеет ошибок или дефектов.

Так как создание автоматического тестирования браузерного расширения с графическим интерфейсом – это слишком трудоёмкий процесс и выходит за рамки данной выпускной квалификационной работы, проведём тест вручную на примере 10 веб-ресурсов.

Для проведения тестирования возьмём 3 популярных российских веб-ресурсов, 4 малоизвестных и 3 фишинговых сайта из базы данных PhishTank, которых не было в датасете обучения нейросети. А для подтверждения точности оценки для одного и того же ресурса, запустим анализ 3 раза для каждого сайта.

Таблица 3.2 — Результаты теста приложения по методу чёрного ящика

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Веб-ресурс | Процент вероятности фишинга | | |
|  | Попытка №1 | Попытка №2 | Попытка №3 |
| 1 | https://www.wildberries.ru | 9% | 9% | 9% |
| 2 | https://www.avito.ru | 33% | 33% | 33% |
| 3 | https://donstu.ru | 3% | 3% | 3% |
| 4 | https://schedule.darksecrets.ru/ | 53% | 53% | 53% |
| 5 | https://easyoffer.ru/ | 65% | 65% | 65% |
| 6 | https://skvilla.su | 60% | 60% | 60% |
| 7 | https://asofast.io/ | 49% | 49% | 49% |
| 8 | https://mcgc1947.wixsite.com/my-site-2 | 78% | 78% | 78% |
| 9 | https://www.clienti-verifica.com/ | 79% | 79% | 79% |
| 10 | https://stop.paylahreport.wiki/ | 70% | 70% | 70% |

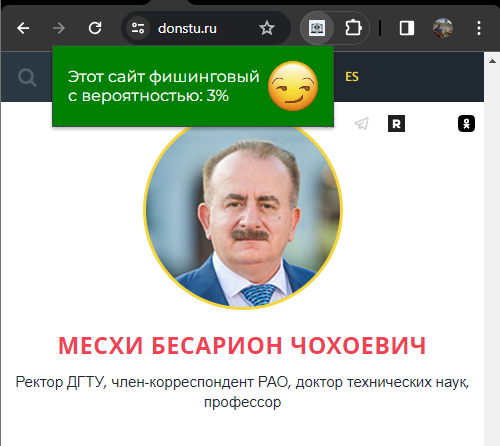


Рисунок 3.4 — Пример проверки легитимного сайта

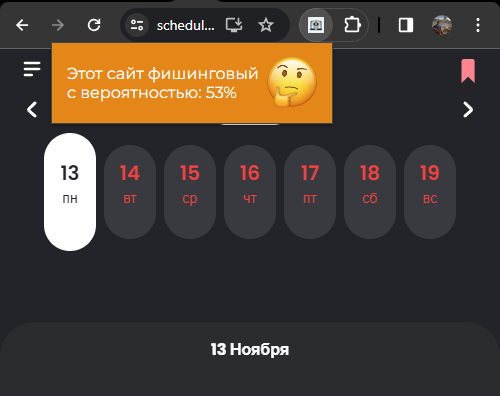


Рисунок 3.5 — Пример проверки малоизвестного сайта

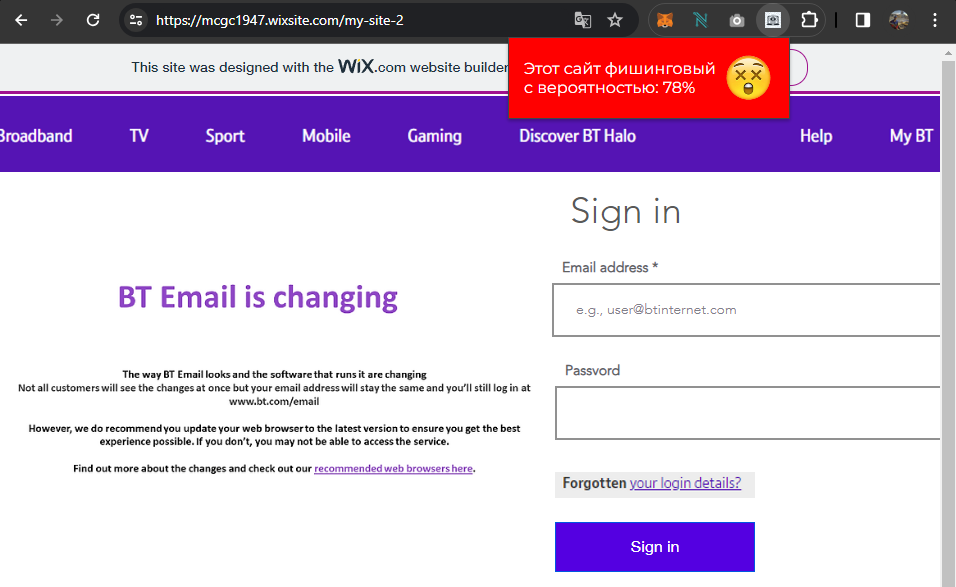


Рисунок 3.6 — Пример проверки фишингового сайта

Таким образом было проведено тестирование и верификация созданного приложения для детектирования фишинговых сайтов с помощью нейросети, разработанного в рамках данной работы. Результаты тестирования и верификации показали, что клиентское приложение работает корректно и надёжно, то есть способно проверять URL на фишинг в режиме реального времени, показывать пользователю результаты проверки и не давать ложных срабатываний или пропусков фишинговых сайтов. Таким образом, можно сделать вывод, что данное приложение является эффективным и точным инструментом для защиты пользователей от фишинговых сайтов, который превосходит существующие решения по всем критериям оценки.

**4. Экономическое обоснование**

Для обоснования экономической эффективности разработанной системы детектирования фишинговых сайтов с помощью нейросетевых технологий был составлен план-график разработки и внедрения, определены затраты на разработку и внедрение, включая оплату труда, материалы, машинное время и общехозяйственные расходы. Оценка экономических показателей включает учет снижения потерь от фишинговых атак, риск утечки данных, повышение доверия клиентов и привлечение новых клиентов. Анализ конкурентоспособности и оценка рисков помогут определить преимущества системы и разработать стратегии смягчения рисков, а также прогнозирование доходов и окупаемости инвестиций позволит оценить финансовые перспективы проекта.

**4.1 Спецификация проекта**

Техническое описание проекта является составной частью проектной документации, содержащей информацию о материалах, изделиях и оборудовании. Эта информация необходима для определения общих потребностей и комплектации материальными ресурсами для строительного объекта. Техническое описание имеет следующие цели:

— Определение полного списка требуемого оборудования;

— Расчет количества материальных ресурсов.

На основе данных ведомости можно определить потребность в материалах, рассчитать их стоимость и составить график приобретения для обеспечения непрерывности процесса разработки системы. Техническое описание системы оценки защищенности беспроводных устройств представлено в таблице 4.1.

Таблица 4.1 — Спецификация системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Материалы | Единица измерения | Требуемое количество | Цена за единицу руб. | Сумма, руб. |
| Сервер Dell PowerEdge R740 | шт. | 1 | 250000 | 250000 |
| Маршрутизатор Cisco ISR 4321 | шт. | 1 | 70000 | 70000 |
| Жесткий диск Seagate IronWolf NAS HDD (4 ТБ) | шт. | 1 | 15000 | 15000 |
| Антивирусная программа Kaspersky Endpoint Security | шт. | 1 | 10000 | 10000 |
| Итого |  | | | 345 000 |

**4.2 План–график проектирования и разработки системы**

Выбор комплекса работ по разработке проекта производится в соответствии со стандартом «ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств», устанавливающим стадии разработки программных продуктов, приведен в таблице 4.2.

Таблица 4.2 — Комплекс работ по разработке системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Содержание работ | Исполнители | Длитель-ность в календар-ных днях | Загрузка | |
| в днях | в % |
| 1 Разработка технического задания | | | | |
| 1.1 Исследование и обоснование разработки | | | | |
| 1.1.1 Постановка задачи | Руководитель | 3 | 3 | 100 |
| Программист | 3 | 100 |
| 1.1.2 Сбор исходных данных | Руководитель | 5 | 3 | 60 |
| Программист | 5 | 100 |
| 1.2 Поиск аналогов и прототипов | | | | |
| 1.2.1 Анализ существующих методов | Руководитель | 7 | 7 | 100 |
| Программист | 7 | 100 |
| 1.2.2 Обоснование необходимости разработки и внедрения | Руководитель | 5 | 5 | 100 |
| Программист | 0 | 0 |

Продолжение таблицы 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1.3 Анализ требований | | | | |
| 1.3.1 Определение и анализ требований к разрабатываемой системе | Руководитель | 4 | 2 | 50 |
| Программист | 4 | 100 |
| 1.3.2 Определение структуры входных и выходных данных | Руководитель | 5 | 2 | 40 |
| Программист | 5 | 100 |
| 1.3.3 Согласование и утверждение технического задания | Руководитель | 4 | 4 | 100 |
| Программист | 4 | 100 |
| Итого по этапу 1 | Руководитель | 33 | 26 | 80 |
| Программист | 28 | 85 |
| 2 Проектирование | | | | |
| 2.1 Проектирование архитектуры системы | Руководитель | 14 | 14 | 100 |
| Программист | 0 | 0 |
| 2.2 Выбор аппаратных средств цифровой радиосвязи | Руководитель | 14 | 7 | 50 |
| Программист | 14 | 100 |
| 2.3 Сборка макета системы на основе аппаратных модулей для тестирования и отладки | Руководитель | 14 | 2 | 14 |
| Программист | 14 | 100 |
| Итого по этапу 2 | Руководитель | 42 | 23 | 53 |
| Программист | 28 | 67 |
| 3 Разработка и тестирование модулей системы | | | | |
| 3.1 Разработка программных модулей системы | Руководитель | 18 | 9 | 50 |
| Программист | 18 | 100 |
| 3.2 Тестирование программной части | Руководитель | 7 | 0 | 0 |
| Программист | 7 | 100 |
| 3.3 Сборка и испытание системы | Руководитель | 14 | 4 | 29 |
| Программист | 14 | 100 |
| 3.4 Анализ результатов экспериментов, формулирование выводов | Руководитель | 7 | 3 | 36 |
| Программист | 7 | 100 |
| Итого по этапу 3 | Руководитель | 46 | 16 | 34 |
| Программист | 46 | 100 |
| 4 Оформление рабочей документации | | | | |
| 4.1 Проведение расчетов показателей безопасности жизнедеятельности | Руководитель | 3 | 0 | 0 |
| Программист | 3 | 100 |
| 4.2 Проведение экономических расчетов | Руководитель | 7 | 0 | 0 |
| Программист | 7 | 100 |

Окончание таблицы 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4.3 Оформление пояснительной записки | Руководитель | 31 |  | 0 |
| Программист |  | 21 | 100 |
| Итого по 4 этапу | Руководитель | 41 | 0 | 0 |
| Программист | 31 | 100 |
| Итого по проекту | Руководитель | 162 | 65 | 40 |
| Программист | 133 | 82 |

Исходя из данных, представленных в таблицах, которые были описаны ранее, был составлен детальный план–график для проектирования и реализации системы, способной обнаруживать фишинговые сайты. Этот план–график демонстрирует, какие этапы и задачи необходимо выполнить для успешного завершения проекта, а также как они связаны между собой.

В результате выполнения проекта должны быть разработана и реализована система детектирования фишинговых сайтов, которая позволяет анализировать веб-страницы на предмет наличия признаков фишинга, таких как поддельные домены, неправильные сертификаты, подозрительные ссылки и формы, и т.д. Система состоит из нескольких программных модулей, которые реализуют различные методы и алгоритмы для обнаружения фишинга, такие как машинное обучение, нейронные сети, логическое вывод, и т.д. Система также имеет веб-интерфейс, который позволяет пользователю делать запрос на сервер и получать результат анализа в виде оценки риска и списка обнаруженных признаков фишинга. Система работает на аппаратных средствах, которые были выбраны с учетом требований к производительности, надежности и безопасности.

Система должна быть протестирована на различных веб-страницах, в том числе на реальных фишинговых сайтах, которые были собраны из различных источников. Результаты тестирования должны показать, что система имеет высокую точность и полноту в детектировании фишинговых сайтов, а также низкую ложную тревогу и пропуск фишинга. Система также должна демонстрировать хорошую скорость и стабильность работы, а также удобство использования для пользователя.

Таблица 4.3 — Календарный график выполнения работ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Содержание работ | Исполнители | Длитель-ность в днях | График работ | |
| начало 1 сентября | окончание 31 января |
| Постановка задачи | Руководитель | 3 | 1 авг | 3 авг |
| Программист | 3 | 1 авг | 3 авг |
| Сбор исходных данных | Руководитель | 3 | 3 авг | 5 авг |
| Программист | 5 | 3 авг | 8 авг |
| Анализ существующих методов | Руководитель | 7 | 8 авг | 15 авг |
| Программист | 7 | 8 авг | 15 авг |
| Обоснование необходимости разработки и внедрения | Руководитель | 5 | 15 авг | 20 авг |
| Программист | 0 | 15 авг | 15 авг |
| Определение и анализ требований к системе | Руководитель | 2 | 20 авг | 22 авг |
| Программист | 4 | 20 авг | 24 авг |
| Определение структуры входных и выходных данных | Руководитель | 2 | 24 авг | 26 авг |
| Программист | 5 | 24 авг | 29 авг |
| Согласование и утверждение технического задания | Руководитель | 4 | 29 авг | 2 сен |
| Программист | 4 | 29 авг | 2 сен |
| Проектирование архитектуры системы | Руководитель | 14 | 2 сен | 16 сен |
| Программист | 0 | 2 сен | 2 сен |
| Выбор аппаратных средств для реализации системы | Руководитель | 7 | 16 сен | 22 сен |
| Программист | 14 | 16 сен | 30 сен |
| Сборка макета системы | Руководитель | 2 | 30 сен | 1 окт |
| Программист | 14 | 30 сен | 13 окт |
| Разработка программных модулей системы | Руководитель | 9 | 13 окт | 22 окт |
| Программист | 18 | 13 окт | 31 окт |
| Тестирование программной части | Руководитель | 0 | 31 окт | 31 окт |
| Программист | 7 | 31 окт | 7 ноя |
| Сборка и испытание системы | Руководитель | 4 | 7 ноя | 11 ноя |
| Программист | 14 | 7 ноя | 21 ноя |
| Анализ результатов экспериментов, формулирование выводов | Руководитель | 3 | 21 ноя | 23 ноя |
| Программист | 7 | 21 ноя | 28 ноя |
| Проведение расчетов показателей безопасности жизнедеятельности | Руководитель |  | 28 ноя | 28 ноя |
| Программист | 3 | 28 ноя | 31 ноя |
| Проведение экономических расчетов | Руководитель |  | 31 ноя | 31 ноя |
| Программист | 7 | 31 ноя | 7 дек |
| Оформление пояснительной записки | Руководитель |  | 7 дек | 7 дек |
| Программист | 31 | 7 дек | 10 янв |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **янв** | **10** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **декабрь** | **31** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **20** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ноябрь** | **31** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Рисунок 4.1 — План-график разработки проекта |
| **20** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **октябрь** | **30** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **20** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **сентябрь** | **31** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **20** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **август** | **30** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **20** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Содержание работы** | | Постановка задачи | Сбор исходных данных | Анализ существующих методов | Обоснование необходимости разработки и внедрения | Определение и анализ требований к разрабатываемой системе | Определение структуры входных и выходных данных | Согласование и утверждение технического задания | Проектирование архитектуры системы | Выбор аппаратных средств цифровой радиосвязи | Сборка макета системы | Разработка программных модулей системы | Тестирование программной части | Сборка и испытание системы | Анализ результатов экспериментов, формулирование выводов | Проведение расчетов показателей безопасности жизнедеятельности | Проведение экономических расчетов | Оформление пояснительной записки |

**4.3 Расчет затрат на разработку проекта**

Предпроизводственные затраты представляют собой единовременные расходы на разработку обеспечивающих или функциональных систем и элементов на всех этапах проектирования, а также затраты на обработку материалов исследования, разработку технического задания, проверки. Сюда включаются затраты на разработку алгоритмов и программ, разработку технического и рабочего проекта системы и её опытной проверки.

Основная заработная плата разработчиков определяется по формуле:

,

где *O* — должностной оклад, руб;

— число рабочих дней в году;

— затраты времени на разработку, рабочие дни.

Число рабочих дней на разработку определяется по формуле:

,

где — календарные дни;

— коэффициент перевода календарных дней в рабочие;

Данный коэффициент равен отношению рабочих дней в году к общему числу календарных дней.

Согласно данным из таблицы 4.1, рассчитаем трудозатраты руководителя:

Рассчитаем трудозатраты программиста:

.

В компании оклад руководителя равен 100 000 руб., а оклад программиста 80 000 руб.

Рассчитаем заработную плату программиста и руководителя по формуле:

Отсюда следует, что основная заработная плата на разработку:

Рассчитаем дополнительную зарплату:

где — коэффициент начисления на дополнительную зарплату.

Премия рассчитывается согласно следующей формуле:

где — коэффициент начисления на премию.

Рассчитаем начисления на единый страховой сбор:

Ввиду того, что проектируемая система должна быть разработана и отлажена с помощью компьютеров, к суммарным затратам на разработку добавляются затраты на их использование.

Доля амортизационных отчислений на компьютерное оборудование, приходящаяся на разработку проекта, определяется по формуле:

где — балансовая стоимость компьютерного оборудования;

— продолжительность использования компьютера программистом, час;

— срок службы компьютерного оборудования. = 6 лет;

число одновременно выполняемых проектов (*z* = 1…3).

 — действительный годовой фонд времени компьютерного оборудования;

Действительный годовой фонд компьютерного оборудования рассчитывается по формуле:

где — номинальный фонд времени работы компьютерного оборудования;

— коэффициент потерь времени; *y* = 3 – 5 %.

Рассчитаем амортизационные отчисления для компьютерного оборудования:

Используемое при конструкторской подготовке системы анализа радиоэфира относится к нематериальным активам предприятия. Амортизационные отчисления на программное обеспечение, приходящиеся на конструкторскую подготовку производства, определяются по формуле:

Где — балансовая стоимость программного обеспечения, руб.;

— срок службы программного обеспечения (при отсутствии фактических данных применяется равным 10 годам).

Рассчитаем затраты электроэнергии, необходимые на проектирование и разработку системы.

где — установленная мощность компьютера, кВт;

— коэффициент загрузки установленной мощности.

— цена электроэнергии, руб/кВт.

Итого затраты на энергию будут следующими:

руб.

Итого затраты на машинное время:

Общехозяйственные расходы:

В таблице 4.4 представлены затраты на разработку и материалы в соответствии со спецификацией, приведённой в п. 5.1.

Таблица 4.4 — Затраты на разработку

|  |  |
| --- | --- |
| Статьи затрат | Затраты |
| Основная заработная плата | 631 578,94 |
| Дополнительная зарплата | 94 736,84 |
| Премия | 252 631,57 |
| Отчисления на социальные нужды | 293 684,20 |
| Затраты на материалы | 345 000 |
| Затраты на машинное время | 39 086,53 |
| Общехозяйственные расходы | 1 705 263,13 |
| ИТОГО | 3 016 981,21 |

В результате проведенных расчетов определено, что себестоимость разработки составляет 3 016 981,21 руб. С учетом нормативной рентабельности R = 15% планируемая прибыль П составит:

*П* = *С* · *R* = 3 016 981,21 · 15% = 452 547,15 руб.

A цена продукта составит:

*Ц* = *С* + *П* = 3 016 981,21 + 452 547,15 = 3 469 528,36 руб.

Цена продукта с учётом НДС составит:

*ЦНДС* = 1,2·*Ц* = 1,2 · 3 469 528,36 = 4 163 434,03 руб.

Потенциальными покупателями разработанной системы являются компании, занимающиеся производством антивирусных программных продуктов.

**5. Безопасность и экологичность проекта**

**5.1 Расчет системы зануления**

[В соответствии с межгосударственным стандартом ГОСТ 12.1.030-81](https://neuralwriter.com/ru/" \t "_blank), который регулирует систему стандартов безопасности труда в области электробезопасности, защитного заземления и зануления, необходимо выполнить расчет системы зануления. Система зануления предусматривает преднамеренное электрическое соединение металлических нетоковедущих частей, которые могут оказаться под напряжением, с нулевым защитным проводником.

Занулением называется преднамеренное электрическое соединение с нулевым защитным проводником металлических нетоковедущих частей, которые могут оказаться под напряжением.

Нулевым защитным проводником называется проводник, соединяющий зануляемые части с глухозаземленной нейтральной точкой обмотки источника тока или ее эквивалентом.

*k* коэффициент надежности=3; Pэ, мощность электродвигателя=10⋅103 Вт; ℓ, длина провода в пределах участка=65 м; Uф, фазное напряжение=220 В; D, диаметр провода в подводящем кабеле=6⋅10-3 м; ρпров, удельное сопротивление алюминиевого проводника= 2,53⋅10-8 Ом⋅м; ρст. – удельное сопротивление стали=1⋅10-7 Ом⋅м; нулевой проводник – труба.

1. Определяем номинальный ток электродвигателя

2. Рассчитываем активное сопротивление алюминиевых проводов

, где

3. Вычисляем активное сопротивление нулевого проводника

1. Рассчитываем площадь поперечного сечения трубы

5. Определяем сопротивление взаимоиндукции между проводами

, где

— абсолютная магнитная проницаемость вакуума, Гн/м;

δ — расстояние между проводами (≈5 мм), м;

— циклическая частота

6. Вычисляем полное сопротивление петли «фаза-нуль»

Ом

7. Определяем ток короткого замыкания

8. Определяем соответствие условию Iк.з. ≥ k Iн; 244,4≥3⋅22,7.

Принимаемая система зануления удовлетворяет условию 244,4≥3⋅22,7.

**5.2 Расчет электромагнитного излучения, создаваемого передающими станциями**

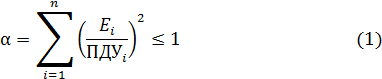
Санитарные правила и нормы СанПиН 2.2.4.3359-16 «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах» устанавливают предельно-допустимые уровни (ПДУ) воздействия на людей электромагнитных излучений в диапазоне частот 30 кГц - 300 Ггц.

При работе радио и теле и иных передающих станций магнитная составляющая по своей величине не имеет существенного значения, поэтому интенсивность ЭМИ оценивается только по величине напряженности электрического поля (Е, в/м).

Таблица 5.1 — Предельно допустимые уровни ЭМИ, создаваемые передающими станциями

|  |  |
| --- | --- |
| Частота, МГц | ПДУ, в/м |
| 30-60 | 5 |
| 60-120 | 4 |
| 120-240 | 3 |
| 240-300 | 2,5 |

При одновременном облучении от нескольких источников, для которых установлены разные ПДУ, должно соблюдаться следующее условие:



где : Еi - напряженность электрического поля, создаваемого i-источником, в/м,

ПДУi- предельно-допустимый уровень для i-источника, в/м.

Для защиты населения от ЭМИ мощных передающих станций (свыше 100 кВт) КВ диапазона, они должны размещаться за пределами населенных мест, вдали от жилой застройки.

Вокруг передающих станций создают санитарно-защитные зоны, размеры которых должны обеспечивать предельно-допустимый уровень ЭМИ в населенных местах (табл.15).

Таблица 5.2 — Размеры санитарных зон

|  |  |
| --- | --- |
| Суммарная мощность передатчика,  кВт | Размеры санитарной зоны, м |
| до 10 | в пределах технической территории |
| 10-75 | 200-300 |
| 75-160 | 400-500 |
| более 160 | 500-1000 |

Санитарная зона разделяется на зону строгого режима (50-100 м) и зону ограниченного пользования в зависимости от мощности передатчика. В зоне строгого режима допускается пребывание только работников передающей станции, и ограниченное время.

В зоне ограниченного пользования можно располагать объекты, в которых граждане могли бы находиться менее 8 час (гаражи, хозяйственно-бытовые помещения и др.)

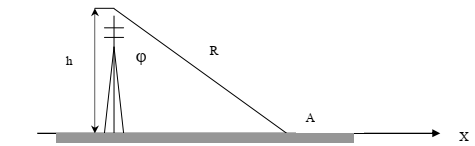


Рисунок 5.1 — Определение напряженности электрического поля в расчетной точке

Электрическая напряженность ЭМИ в расчетной точке А определяется по формуле:





где: Р - мощность источника, Вт; φ - коэффициент направленности антенны, рад



где: R - расстояние от антенны до расчетной точки, м; h - высота антенны, м; x - расстояние от основания антенны до расчетной точки, м.

Электрическая напряженность ЭМИ в жилом помещении определяется по формуле:

Еж = к · Е

где: к - ослабление ЭМИ стенами здания; к=1 – для кирпичных стен; к=0,2 – для панельных стен.

В соответствии с нормативами [СанПиН 1.2.3685-21](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BC%D0%B0%D0%B3%D0%BD%D0%B8%D1%82%D0%BD%D0%BE%D0%B5_%D0%B8%D0%B7%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)[1](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BC%D0%B0%D0%B3%D0%BD%D0%B8%D1%82%D0%BD%D0%BE%D0%B5_%D0%B8%D0%B7%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), которые регламентируют допустимые уровни ЭМИ в различных диапазонах частот, а также мероприятия по контролю, профилактике и защите от ЭМИ, произведем расчёт электромагнитного излучения, создаваемого передающими станциями.

Таблица 5.3 — Исходные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| h, м | 1 канал | | 2 канал | | 3 канал | |
| f1 | P1 | f2 | P2 | f3 | P3 |
| 210 | 40 | 1000 | 70 | 4000 | 130 | 7000 |

Определим ПДУ для каждого канала по табл. 14 и занесем в табл.17.

Определим электрическую напряженность в расчетных точках и результаты расчета сведем в табл. 5.4 и рис.5.2.

Пример расчета для х=50 м.

Таблица 5.4. — Электрическая напряженность в расчетных точках

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | arctg x/h | *E*1 | *E*2 | *E*3 | *α* |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 50 | 0,464 | 2,35 | 3,32 | 1,66 | 1,34 |
| 100 | 0,785 | 2,42 | 3,43 | 1,72 | 1,43 |
| 150 | 0,983 | 2,13 | 3,02 | 1,51 | 1,11 |
| 200 | 1,107 | 1,82 | 2,58 | 1,29 | 0,808 |
| 250 | 1,190 | 1,57 | 2,21 | 1,11 | 0,596 |
| 300 | 1,249 | 1,37 | 1,94 | 0,97 | 0,456 |
| ПДУ | - | 5 | 4 | 3 | 1 |

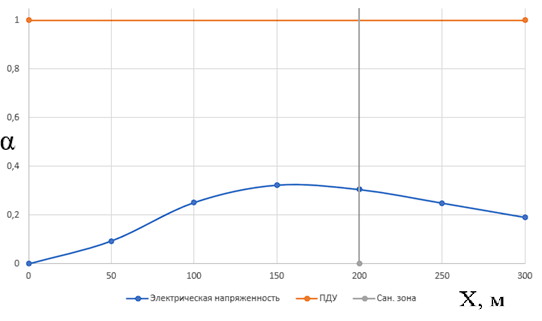


Рисунок 5.2 — Результат расчёта электрической напряженности

Суммарная мощность передатчиков:

1000+4000+7000=12000 Вт=12 кВт

Отсюда по табл.5.2 определяем размер санитарной зоны – 200 м.

Находим по табл. 5.4 величину Е для X=200м и рассчитываем напряженность электрического поля в кирпичном и панельном домах.

Таблица 5.5. — Напряженность электрического поля в кирпичном и панельном домах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *E*1 | *E*2 | *E*3 | *α* |
| Х=200 | 1,107 | 1,82 | 2,58 | 1,29 |
| Кирпичный  дом | 1,107 | 1,82 | 2,58 | 1,29 |
| Панельный  дом | 0,221 | 0,364 | 0,516 | 0,258 |
| ПДУ | 5 | 4 | 3 | 1 |

На границе санитарной зоны ЭМИ в кирпичных и панельных домах не превышает допустимые значения.

**5.3 Оценка зон теплового воздействия**

Согласно Техническому регламенту о требованиях пожарной безопасности №123-ФЗ (редакция, действующая с 1 марта 2023 года) и ПОСТАНОВЛЕНИЮ ПРАВИТЕЛЬСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ от 16 сентября 2020 года N 1479 "Правила противопожарного режима в Российской Федерации" (с изменениями на 24 октября 2022 года), необходимо определить протяженность безопасной зоны теплового воздействия при возгорании промышленного объекта на территории завода на объекты и вещества, находящиеся в зоне теплового воздействия.

Таблица 5.6 — Исходные данные

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Характеристика горящего промышленного объекта | | | | Характеристика объекта или вещества подверженного тепловому воздействию |
| Деревянное здание | | Резервуар с нефтепродуктами | |
| Высота, м | Длина, м | Диаметр, м | Вещество |
| 8 | 120 | 12 | нефть | возгорание древесины через 10 минут |

Расчет протяженности зон теплового воздействия *R*, м, при горении зданий и промышленных объектов производится по формуле:

При горении резервуара с бензином

, где  –– плотность потока собственного излучения пламени пожара, кВт/м2;  –– критическая плотность потока излучения пламени пожара, падающего на облучаемую поверхность и приводящую к тем или иным последствиям, кВт/м; *R*\* –– приведенный размер очага горения, м, равный: –– для горящих зданий; 0,8⋅*D*рез –– для горения нефтепродуктов в резервуаре; *, h* –– длина и высота объекта горения, м; *D*рез –– диаметр резервуара, м.

При горении деревянного здания протяженность зоны горения составит метра и протяженность зоны горения резервуара с бензином составит метра.

Вывод: согласно проведённым расчётам, показатели системы зануления, электромагнитного излучения и итоги оценки зон теплового воздействия находятся в норме для системы детектирования фишинговых сайтов**.**

**Заключение**

В результате выпускной квалификационной работы была создана система детектирования фишинговых сайтов с помощью нейросетевых технологий. В работе были выполнены следующие задачи:

1. Проведён обзор и анализ существующих методов детектирования фишинговых сайтов. И был сделан вывод, что метод детектирования с использованием эвристического анализа и нейросетевых технологий самый эффективный.
2. Проведён анализ и выбран метод детектирования фишинговых сайтов с использованием эвристического анализа, состоящего из 27 параметров, свёрточной, текстовой и оркеструющей нейронных сетей.
3. Сформирован и предобработан набор данных для обучения состоящий из более чем 120 000 строк данных.
4. Разработаны и обучены нейронные сети для детектирования фишинговых сайтов на основе полученного набора данных.
5. Протестирована и проанализирована работы нейросетей. По результатам тестов процент верных ответов нейросетей превышает 90%.
6. Создано клиентское и серверное приложение.
7. Оценена корректность и стабильность работы итогового приложения, система верно определяет класс сайта и не меняет свой ответ при повторном анализе.

В результате работы было показано, что нейросетевые технологии являются перспективным инструментом для детектирования фишинговых сайтов. Все аспекты работы – от определения и классификации фишинговых сайтов до разработки системы детектирования – имеют важное значение для создания эффективной системы защиты от фишинга. Возможные направления для дальнейших исследований включают улучшение нейросетевой модели и ее адаптацию к динамичным характеристикам фишинговых сайтов.

**Перечень использованных информационных ресурсов**

1. Статья. Актуальные киберугрозы: I квартал 2023 года – URL: <https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2023-q1/> (дата обращения 4.10.2023) – Текст: электронный.
2. Статья. Vahid Shahrivari, Mohammad Mahdi Darabi, Mohammad Izadi – Phishing Detection Using Machine Learning Techniques – URL: <https://paperswithcode.com/paper/phishing-detection-using-machine-learning> (дата обращения 15.10.2023) – Текст: электронный.
3. Статья. Екатерина Поддубная – Правовые аспекты борьбы с «фишингом» <https://blog.pravo.ru/blog/5805.html> (дата обращения 15.10.2023) – Текст: электронный.
4. Российская Федерация. Постановление Пленума Верховного Суда Российской Федерации № 51 от 27 декабря 2007 года – URL: https://vsrf.ru/documents/own/8246/ (дата обращения: 16.10.2023). – Текст: электронный.
5. Уголовный кодекс Российской Федерации от 13 июля 1996 г. № 63-ФЗ (Собрание законодательства Российской Федерации, 1996, № 25, ст. 2954).
6. [Эндрю Траск](https://www.chitai-gorod.ru/author/trask-endryu-560708). Грокаем глубокое обучение. – Питер, 2019 – 65 c. – ISBN 978-5-44-611334-7.
7. Статья. Oluwatobi Ayodeji Akanbi – Phishing Detection Using Machine Learning Techniques – URL: https://www.sciencedirect.com/topics/computer-science/phishing-detection (дата обращения 22.11.2023) – Текст: электронный.
8. Статья. Hannousse A., Yahiouche S. Towards benchmark datasets for machine learning based website phishing detection: An experimental study //Engineering Applications of Artificial Intelligence. – 2021. – Т. 104. – С. 104347.
9. Статья. Basit A. et al. A comprehensive survey of AI-enabled phishing attacks detection techniques //Telecommunication Systems. – 2021. – Т. 76. – С. 139-154.
10. Статья. Patil V. et al. Detection and prevention of phishing websites using machine learning approach //2018 Fourth international conference on computing communication control and automation (ICCUBEA). – Ieee, 2018. – С. 1-5.
11. Статья. Garcés I. O., Cazares M. F., Andrade R. O. Detection of phishing attacks with machine learning techniques in cognitive security architecture //2019 International Conference on Computational Science and Computational Intelligence (CSCI). – IEEE, 2019. – С. 366-370.
12. Официальный сайт PhishTank – URL: https://www.phishtank.org/ (дата обращения 30.11.2023) – Текст: электронный.
13. Статья. Confusion Matrix in Machine Learning https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (дата обращения 2.12.2023) – Текст: электронный. ГОСТ12.1.005-88 (дата актуал. 15.10.2021 г.); СанПиН 1.2.3685-21. «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
14. СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания».
15. Технический регламент о требованиях пожарной безопасности: федеральный закон: [принят Гос. Думой Федерального собрания РФ 22 июля 2008 г. №123-ФЗ];
16. Постановления Правительства Российской Федерации 16 сентября 2020 года №1479 Правила противопожарного режима в Российской Федерации с изм. на 24.10. 2022 г.
17. Федеральный закон «Технический регламент о требованиях пожарной безопасности» № 123-ФЗ (редакция, действующая с 1 марта 2023 года) [Электронный ресурс] // Консультант Плюс. Режим доступа: <https://www.consultant.ru/document>  
    /cons\_doc\_LAW\_78699/

Приложение A

Скрипт на языке программирования Python c использованием веб драйвера Selenium

# импорт необходимых библиотек

from bs4 import BeautifulSoup

from selenium import webdriver

from selenium.webdriver.chrome.service import Service as ChromeService

from urllib.parse import urlparse

from database\_helper import DatabaseHelper

import aiohttp

import os

# Получение доменного имени из URL

def get\_domain\_from\_url(url: str):

    parsed\_url = urlparse(url)

    domain = parsed\_url.netloc

    return domain

# Получение html кода

async def fetch(session, url):

    async with session.get(url) as response:

        return await response.text()

dataset\_directory = 'dataset' # Папка датасета

# Настройка драйвера Google Chrome

def get\_webdriver():

    # Разрешение скриншотов (ширина х высота)

    desired\_width = 1280

    desired\_height = 720

    driver\_path = '.\\ChromeDriver\\chromedriver.exe' # Путь до драйвера Google Chrome

    options = webdriver.ChromeOptions() # Создание настроек драйвера Google Chrome

    options.add\_argument('--headless')  # Запуск драйвера без открытия окна

    chrome\_service = ChromeService(executable\_path=driver\_path) # Создание отдельного экземляра драйвера для каждой ссылки

    driver = webdriver.Chrome(service=chrome\_service, options=options)

    driver.set\_window\_size(desired\_width, desired\_height) # Установка необходимого разрешения

    return driver

# Подключение к базе данных

db\_helper = DatabaseHelper('database.db')

# Сбор и схранение данных

async def scrape\_and\_save\_data(url: str):

    async with aiohttp.ClientSession() as session:

            html = None

            soup = None

            try:

                html = await fetch(session, url)

                soup = BeautifulSoup(html, "html.parser")

            except:

                 return Exception

            domain = get\_domain\_from\_url(url) # Получение доменного имени из URL

            site\_directory = str(db\_helper.get\_last\_id())

            full\_path = os.path.join(dataset\_directory, site\_directory)

            os.makedirs(full\_path, exist\_ok=True) # Создание папки для каждого сайта

            screenshot\_filename = f"screenshot-{domain}.png"

            screenshot\_path = os.path.join(full\_path, screenshot\_filename)

            driver.get(url)

            driver.implicitly\_wait(10)

            driver.save\_screenshot(screenshot\_path)

            text = soup.get\_text() # Сбор текстового контента

            links = soup.find\_all('a', href=True) # Нахождение ссылок на сайт и добавление их в очередь

            url\_array = [link['href'] for link in links if link['href'].startswith(('http://', 'https://'))]

            add\_finded\_URLs(url\_array)

            text\_filename = f"text-{domain}.txt"

            with open(os.path.join(full\_path, text\_filename), "w", encoding="utf-8") as text\_file:

                text\_file.write(text)

            db\_helper.write\_in\_data(url, os.path.abspath(screenshot\_path), text, domain)

            print(f"Сохранены данные для сайта {url} в папку {full\_path}")

# Метод сбора ссылок с сайта

def add\_finded\_URLs(links):

    for link in links:

        if db\_helper.check\_url\_exist(link):

            continue

        domain = get\_domain\_from\_url(link)

        db\_helper.add\_to\_queue(link, domain)

        if not db\_helper.check\_domain\_exist(domain):

            db\_helper.write\_in\_domain(domain, 0, 0)

        db\_helper.increment\_domain\_count\_queue(domain)

async def queue\_collection():

    try:

        row = db\_helper.get\_ru\_row\_from\_queue\_with\_min\_domain\_count()

        url = row[1]

        domain = row[2]

        await scrape\_and\_save\_data(url)

        db\_helper.increment\_domain\_count\_data(domain)

        db\_helper.remove\_from\_queue(url)

        db\_helper.decrement\_domain\_count\_queue(domain)

    except Exception as e:

            db\_helper.add\_to\_error(url, get\_domain\_from\_url(url))

            print(f"Не удалось обработать сайт {url} по причине  в папку {e}")

driver = get\_webdriver()

count = 0

while True:

        queue\_collection()

        count = count + 1

        if count == 10:

            count = 0

            driver.quit()

            driver = get\_webdriver()

db\_helper.close\_connection()

Приложение Б

Листинг модели текстовой нейронной сети.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import torch

from torch import nn

from torch.nn import functional as F

from Utils.simple\_tokenizer import tokenize\_text\_simple\_regex, tokenize\_corpus, build\_vocabulary, \

    vectorize\_texts, SparseFeaturesDataset

from Utils.pipeline import train\_eval\_loop, predict\_with\_model, init\_random\_seed

init\_random\_seed()

data\_size = 100

train\_dataset = pd.read\_csv('text\_dataset\_test.csv', sep=',', encoding='utf-8', dtype=str, nrows=data\_size).sample(n=data\_size, random\_state=42)

dataset\_len = len(train\_dataset)

percentage\_of\_dataset\_in\_test = 80

# Вычисление количества данных для обучения

train\_data\_len = int(len(train\_dataset) \* (percentage\_of\_dataset\_in\_test / 100))

train\_source = train\_dataset['text'].sample(n=train\_data\_len, random\_state=42)

cleaned\_train\_source = [str(x) for x in train\_source.values.tolist() if x is not None and str(x).lower() != 'nan']

test\_source = train\_dataset['text'].drop(train\_source.index)

cleaned\_test\_source = [str(x) for x in test\_source.values.tolist() if x is not None and str(x).lower() != 'nan']

train\_tokenized = tokenize\_corpus(cleaned\_train\_source)

test\_tokenized = tokenize\_corpus(cleaned\_test\_source)

MAX\_DF = 0.8

MIN\_COUNT = 5

vocabulary, word\_doc\_freq = build\_vocabulary(train\_tokenized, max\_doc\_freq=MAX\_DF, min\_count=MIN\_COUNT)

UNIQUE\_WORDS\_N = len(vocabulary)

print('Количество уникальных токенов', UNIQUE\_WORDS\_N)

print(list(vocabulary.items())[:10])

VECTORIZATION\_MODE = 'tfidf'

train\_vectors = vectorize\_texts(train\_tokenized, vocabulary, word\_doc\_freq, mode=VECTORIZATION\_MODE)

test\_vectors = vectorize\_texts(test\_tokenized, vocabulary, word\_doc\_freq, mode=VECTORIZATION\_MODE)

UNIQUE\_LABELS\_N = len(set(train\_dataset['label']))

print('Количество уникальных меток', UNIQUE\_LABELS\_N)

train\_dataset = SparseFeaturesDataset(train\_vectors, train\_dataset['label'])

model = nn.Linear(UNIQUE\_WORDS\_N, UNIQUE\_LABELS\_N)

scheduler = lambda optim: \

    torch.optim.lr\_scheduler.ReduceLROnPlateau(optim, patience=5, factor=0.5, verbose=True)

best\_val\_loss, best\_model = train\_eval\_loop(model=model,

                                            train\_dataset=train\_vectors,

                                            val\_dataset=test\_vectors,

                                            criterion=F.cross\_entropy,

                                            lr=1e-1,

                                            epoch\_n=200,

                                            batch\_size=1,

                                            l2\_reg\_alpha=0,

                                            lr\_scheduler\_ctor=scheduler)

# Сохранение результатов

torch.save(best\_model.state\_dict(), 'text\_model.pth')

Приложение В

Листинг модели свёрточной нейронной сети.

import matplotlib.pyplot as plt

import torch

import torch.nn as nn

import torch.optim as optim

from tqdm import tqdm

from torch.utils.data import DataLoader

from torchvision import datasets, transforms

from sklearn.model\_selection import train\_test\_split

import pandas as pd

import numpy as np

import os

# Определение параметров

image\_width, image\_height = 1280, 720

epochs = 80

batch\_size = 1

# Загрузка данных из CSV файлов

legitimate\_data = pd.read\_csv('legitimate.csv', sep='\t', encoding='utf-16')

phishing\_data = pd.read\_csv('phishing.csv', sep='\t', encoding='utf-16')

# Добавление столбца 'category' в legitimate\_data

legitimate\_data = legitimate\_data.assign(category='legitimate')

# Добавление столбца 'category' в phishing\_data

phishing\_data = phishing\_data.assign(category='phishing')

# Разделение данных на обучающую и проверочную выборку для легитимных сайтов

legitimate\_train, legitimate\_validation = train\_test\_split(legitimate\_data, test\_size=0.2, random\_state=42)

# Разделение данных на обучающую и проверочную выборку для фишинговых сайтов

phishing\_train, phishing\_validation = train\_test\_split(phishing\_data, test\_size=0.2, random\_state=42)

# Объединение данных обратно, если это необходимо

train\_data = .\\dataset\\Train'

validation\_data = .\\dataset\\Validation'

# Определение преобразований данных

data\_transforms = {

    'train': transforms.Compose([

        transforms.Resize((image\_width, image\_height)),

        transforms.ToTensor(),

        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

    ]),

    'val': transforms.Compose([

        transforms.Resize((image\_width, image\_height)),

        transforms.ToTensor(),

        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

    ]),

}

# Создание загрузчиков данных

train\_dataset = datasets.ImageFolder(train\_data, transform=data\_transforms['train'])

val\_dataset = datasets.ImageFolder(validation\_data, transform=data\_transforms['val'])

train\_loader = DataLoader(train\_dataset, batch\_size=batch\_size, shuffle=True)

val\_loader = DataLoader(val\_dataset, batch\_size=batch\_size, shuffle=True)

# Определение модели

model = nn.Sequential(

    nn.Conv2d(3, 32, kernel\_size=3, stride=1, padding=1),

    nn.ReLU(),

    nn.MaxPool2d(kernel\_size=2, stride=2),

    nn.Conv2d(32, 64, kernel\_size=3, stride=1, padding=1),

    nn.ReLU(),

    nn.MaxPool2d(kernel\_size=2, stride=2),

    nn.Conv2d(64, 128, kernel\_size=3, stride=1, padding=1),

    nn.ReLU(),

    nn.MaxPool2d(kernel\_size=2, stride=2),

    nn.Flatten(),

    nn.Linear(128 \* 160 \* 90, 512),

    nn.ReLU(),

    nn.Linear(512, 1),

    nn.Sigmoid()

)

# Определение функции потерь и оптимизатора

criterion = nn.BCELoss()

optimizer = optim.RMSprop(model.parameters())

device = torch.device('cuda')

model = model.to(device)

# Создание загрузчиков данных с использованием многопоточности и pin\_memory для ускорения загрузки

train\_loader = DataLoader(train\_dataset, batch\_size=batch\_size, shuffle=True, num\_workers=4, pin\_memory=True)

val\_loader = DataLoader(val\_dataset, batch\_size=batch\_size, shuffle=True, num\_workers=4, pin\_memory=True)

# Обучение модели

for epoch in range(epochs):

    running\_loss = 0.0

    progress\_bar = tqdm(enumerate(train\_loader), total=len(train\_loader), leave=False)

    for i, (inputs, labels) in progress\_bar:

        inputs = inputs.to(device)

        labels = labels.to(device)

        labels = labels.view(-1, 1)  # Изменение размера меток

        labels = labels.float()  # Преобразование меток в тип Float

        # Обнуление градиентов

        optimizer.zero\_grad()

        # Прямой проход, обратное распространение и оптимизация

        outputs = model(inputs)

        loss = criterion(outputs, labels)

        loss.backward()

        optimizer.step()

        running\_loss += loss.item() \* inputs.size(0)

        progress\_bar.set\_description(f'Epoch {epoch+1}/{epochs}, Loss: {running\_loss/(i+1):.4f}')

    # Вывод статистики обучения

    epoch\_loss = running\_loss / len(train\_loader.dataset)

    print(f'\nEpoch {epoch+1}/{epochs}, Loss: {epoch\_loss:.4f}')

print('Finished Training')

# Визуализация результатов

dataiter = iter(val\_loader)

images, labels = dataiter.next()

images = images.to(device)

labels = labels.to(device)

# Визуализация результатов

dataiter = iter(val\_loader)

images, labels = dataiter.next()

images = images.to(device)

labels = labels.to(device)

# Получение предсказаний модели

outputs = model(images)

\_, predicted = torch.max(outputs, 1)\

# Вывод изображений и их предсказанных метокф

fig = plt.figure(figsize=(25, 4))

for idx in np.arange(20):

    ax = fig.add\_subplot(2, 10, idx+1, xticks=[], yticks=[])

    plt.imshow(np.transpose(images[idx].cpu()))

 # Сохранение модели

torch.save(model.state\_dict(), 'model.pth')

Приложение Г

Листинг модели оркеструющей нейронной сети.

import torch

import torch.nn as nn

import torch.optim as optim

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from torch.utils.data import DataLoader, TensorDataset

# Определяем архитектуру сети

class PhishingNet(nn.Module):

    def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

        super(PhishingNet, self).\_\_init\_\_()

        self.fc1 = nn.Linear(input\_size, hidden\_size)

        self.relu = nn.ReLU()

        self.fc2 = nn.Linear(hidden\_size, output\_size)

        self.sigmoid = nn.Sigmoid()

    def forward(self, x):

        out = self.fc1(x)

        out = self.relu(out)

        out = self.fc2(out)

        out = self.sigmoid(out)

        return out

# Загружаем данные из CSV файла

def load\_data(file\_path):

    data = pd.read\_csv(file\_path)

    X = data.iloc[:, :-1].values

    y = data.iloc[:, -1].values

    return X, y

# Инициализируем параметры

input\_size = 29  # Количество входных флагов

hidden\_size = 64  # Количество нейронов в скрытом слое

output\_size = 1  # Количество выходных классов (1 для бинарной классификации)

# Создаем экземпляр сети

net = PhishingNet(input\_size, hidden\_size, output\_size)

# Функция потерь и оптимизатор

criterion = nn.BCELoss()  # Бинарная кросс-энтропия

optimizer = optim.Adam(net.parameters(), lr=0.001)

# Загружаем данные

X, y = load\_data('phishing\_data.csv')

# Разделяем данные на обучающий и тестовый наборы

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Преобразуем numpy массивы в тензоры PyTorch

X\_train\_tensor = torch.tensor(X\_train, dtype=torch.float32)

y\_train\_tensor = torch.tensor(y\_train, dtype=torch.float32).view(-1, 1)

X\_test\_tensor = torch.tensor(X\_test, dtype=torch.float32)

y\_test\_tensor = torch.tensor(y\_test, dtype=torch.float32).view(-1, 1)

# Создаем загрузчики данных

train\_data = TensorDataset(X\_train\_tensor, y\_train\_tensor)

train\_loader = DataLoader(train\_data, batch\_size=32, shuffle=True)

# Обучение сети

for epoch in range(100):

    for i, (inputs, labels) in enumerate(train\_loader):

        optimizer.zero\_grad()

        outputs = net(inputs)

        loss = criterion(outputs, labels)

        loss.backward()

        optimizer.step()

    if (epoch+1) % 10 == 0:

        print(f'Epoch [{epoch+1}/100], Loss: {loss.item()}')

# Сохраняем модель

torch.save(net.state\_dict(), 'phishing\_net.pth')