Санкт–Петербургский государственный университет
Кафедра технологии программирования

МАРТЫНЮК Роман Анатольевич

Выпускная квалификационная работа
Реализация браузерного плагина для поиска фишинговых сайтов по косвенным признакам

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2016 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:
доцент кафедры технологий программирования,
к.т.н. Блеканов Иван Станиславович
Консультант:
ведущий разработчик в компании Neuromation
Найден Алексей Владимирович
Рецензент:
доцент кафедры компьютерного моделирования и многопроцессорных систем,
к.т.н Гришкин Валерий Михайлович

Санкт-Петербург
2020 г.

Содержание

Введение 3
Обзор существующих инструментов 4
Постановка задачи 5
1.1 Основные критерии 6
1.2 Описание алгоритма поиска опечаток 10
Глава 2. Нейросетевой подход 11
2.1. Данные 11
2.2 Обучение нейронной сети на языке Python 13
2.3 Обучение нейронной сети на языке JavaScript 17
Глава 3. Работа расширения EPSDfree 18
3.1 Описание работы расширения 18
3.2 Результат работы расширения в браузерах 19
Заключение 23
Список литературы 24
Приложения 25

Введение

Сейчас в Интернете остро стоит проблема кражи личных данных посредством фишинговых сайтов. На данный момент существуют инструменты, предупреждающие пользователя от том, что ссылка, по которой он пытается перейти, введет на поддельный сайт. Но они не гарантируют полную защиту от фишинга из-за принципа их работы, который мы рассмотрим далее в данной работе.

Забегая вперед, надо отметить, что сейчас есть бесплатные сервисы, которые могут проверять сайты на подлинность, но их проблема в том, что это отнимает время для проверки. Еще существует ограничение на количество запросов для бесплатной версии, а снимается оно (или увеличивается лимит) только при платной подписке. Обычный пользователь не будет проверять каждый сайт, на который он заходит, да и не каждый готов заплатить за подписку. Поэтому хотелось бы иметь простой способ определять фишинговый сайт в режиме реального времени.

Обзор существующих инструментов

В статье [1] описаны методы антифишинговой защиты самых популярных браузеров. Все имеют очень похожий принцип обнаружения фишиговых сайтов. Как пример, рассмотрим защиту от фишинга и вредоносного ПО в браузере Google Chrome:

«Функция безопасного просмотра Google Chrome, отвечающая за обнаружение фишинга и вредоносного ПО, включена по умолчанию. При попытке посещения сайта, подозреваемого в фишинге или распространении вредоносного ПО, браузер показывает предупреждение.

Принцип ее работы состоит в следующем. В браузер загружается список с информацией о сайтах, которые могут содержать вредоносное ПО или подозреваются в фишинге. Этот список не содержит полные адреса URL каждого подозрительного сайта. Вместо этого каждый URL хэшируется (изменяется таким образом, что его нельзя прочесть) и разделяется на фрагменты. Только часть каждого хэшируемого URL включается в список в браузере. При работе в Интернете браузер создает хэшированные версии посещаемых URL и проверяет их в соответствии со списком. Если адрес посещаемого сайта соответствует хэшированному фрагменту URL в списке, браузер свяжется с серверами Google и запросит полный список (а не только фрагменты) хэшированных URL подозрительных страниц. Затем компьютер определит, является ли сайт подозрительным, и выведет соответствующее предупреждение.» [1]

Как ещё пример, рассмотрим одно из самых популярных браузерных расширений uBlock origin, которое предназначено для блокировки рекламы, но также имеет инструменты для блокировки фишинговых сайтов. Как описано в документации [2], данное расширение использует собственные списки фильтров для блокировки рекламы и черные списки для блокировки вредоносных сайтов. Это говорит о том, что uBlock origin использует тот же принцип, что и антифишинговая защита браузеров, то есть черный список сайтов.

Постановка задачи

Из рассмотренного выше, можно заметить, что принцип, по которому работают современные инструменты определения статуса сайта, имеют изъян. Сайт не будет определен как фишинговый, если его нет в черном списке. По статистике за месяц на один такой сайт заходит от 15 до 18 тысяч пользователей. Опытный пользователь может определить, что сайт поддельный «на глаз», а обычный, не имеющий знаний в этой области, – может лишиться своих данных.

Поэтому ставится задача создания такого браузерного расширения, которое может определять фишинговый сайт в режиме реального времени по косвенным признакам, не опираясь на черный список. Данная задача включает в себя:

1. Определить основные критерии, которые будут моделировать, как опытный пользователь определяет фишинговый сайт или нет.
2. Создать механизм взаимодействия критериев друг с другом для определения статуса сайта.
3. Реализовать вышеперечисленные пункты в браузерном расширении.

Глава 1. Работа с критериями

Первый шаг всей работы – это определение критериев с помощью анализа схожих признаков у фишинговых сайтов.

1.1 Основные критерии

Часть критериев были взяты из [3, 4] и проверены на практике, а остальные были выделены в ходе изучения различных фишинговых сайтов. В итоге был составлен список основных критериев:

1. соединение по защищенному протоколу https;
2. наличие домена сайта в HSTS Preload List;
3. наличие у доменного имени верхнего уровня DNSSEC-записей;
4. является ли тело сайта картинкой;
5. наличие тега <form>, который отправляется на сторонний сайт;
6. отношение ссылок, ведущих на этот же сайт, к ссылкам, ведущих на сторонний сайт;
7. отношение количества ссылок в тегах <link>, использующих CSS-файлы этого сайта, к количеству ссылок, использующих CSS-файлы с других сайтов;
8. количество слов на странице;
9. количество слов с опечатками;
10. отношение количества слов с опечатками к общему количеству слов на странице.

Пять из этих критериев являются категориальными признаками, а остальные пять – численными.

DNSSEC	Домен	Протокол	Тело img	Опечатки/Слова	Форма	Поз.ссылки/Нег.ссылки	Ссылки стилистики	Опечатки	Слова
1.0	0.0	-1.0	1.0	0.000000	-1.0	0.000000	0.000000	0.0	10.0

Рис.1 Пример результата критериев

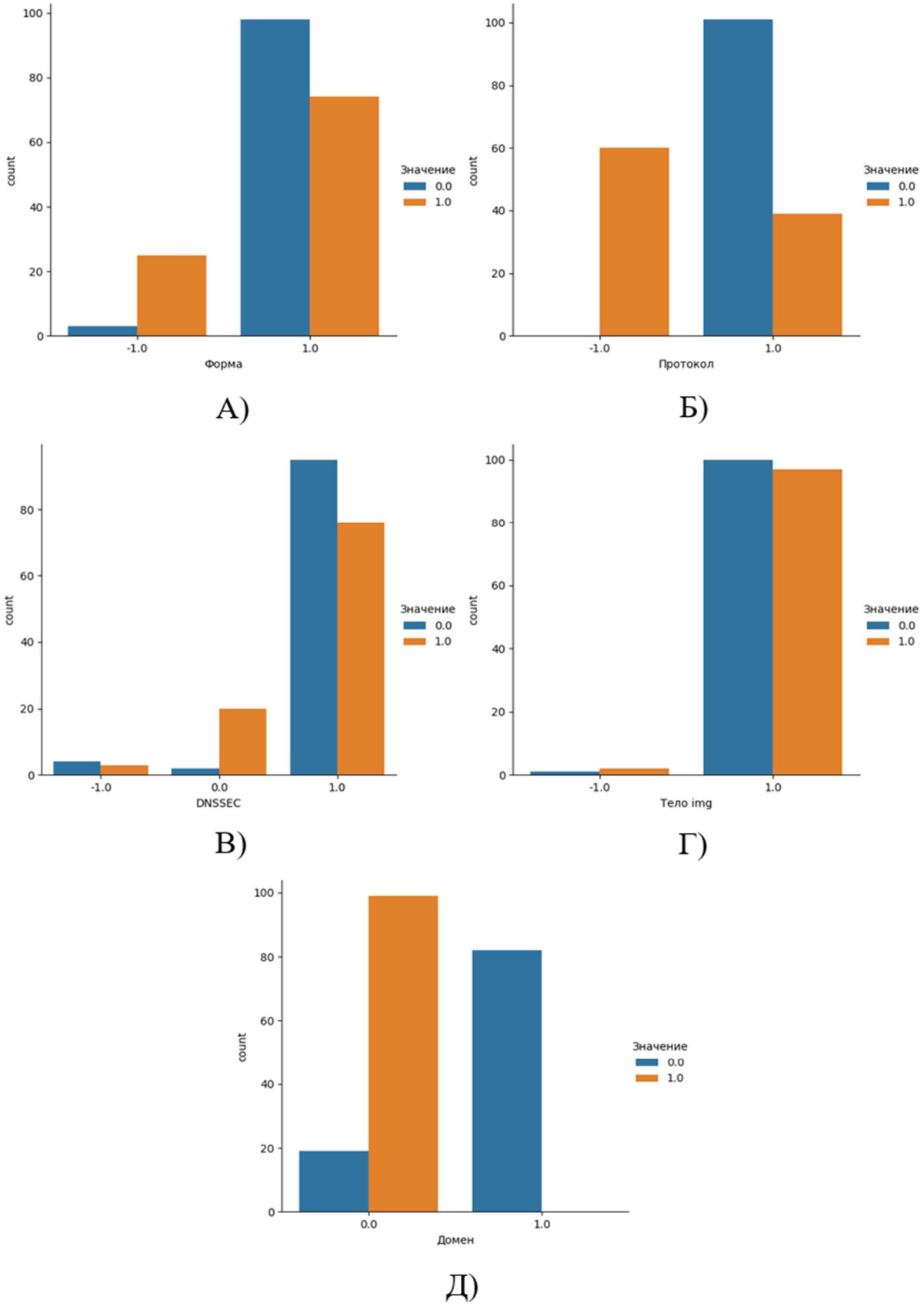


Рис. 2 Категориальные признаки

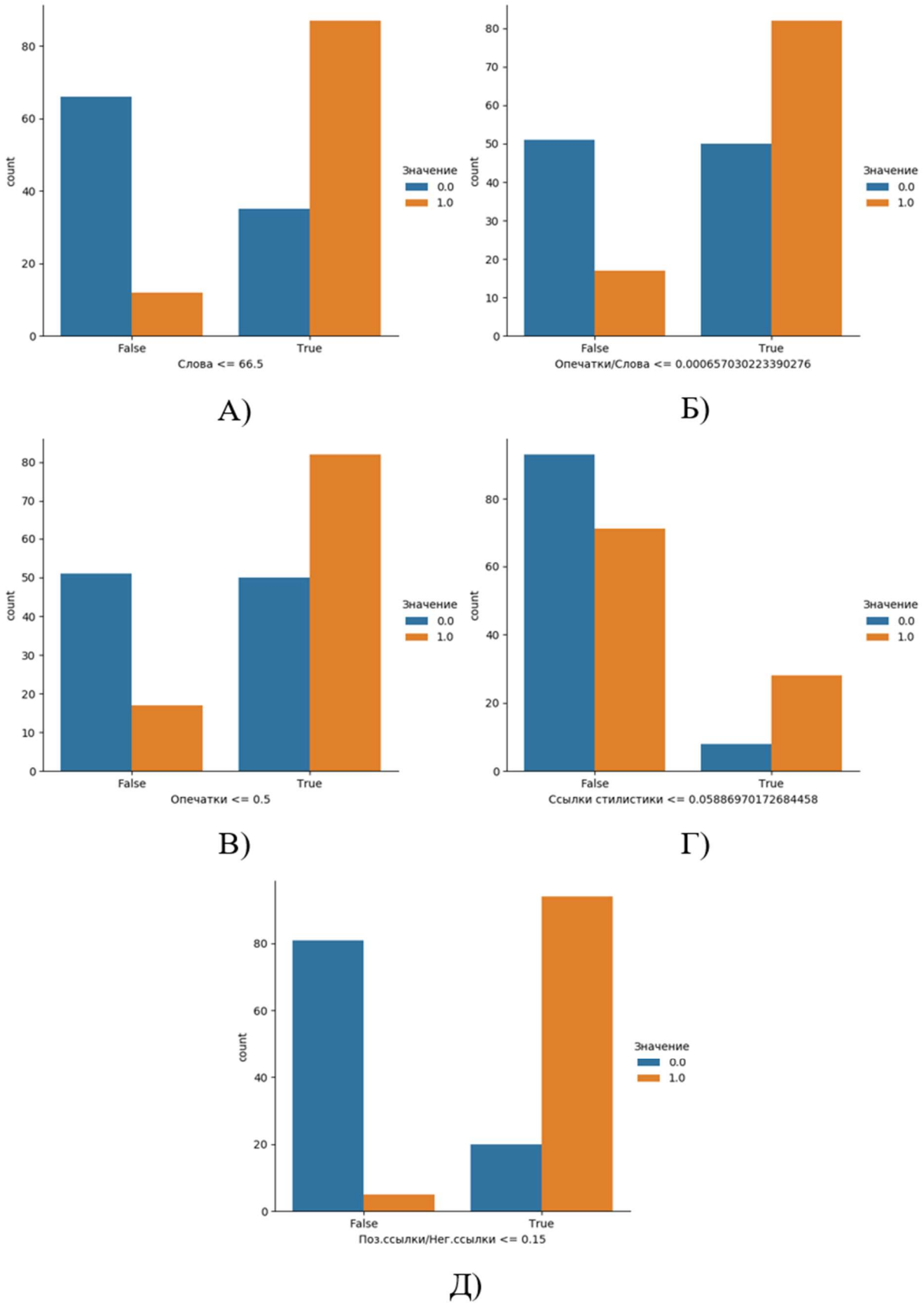


Рис. 3 Численные признаки

На Рис.2 представлена статистика категориальных признаков, а именно критерии 1, 2, 3, 4, 5 (Б, Д, В, Г, А соответственно). Оранжевым цветом показаны фишинговые сайты, а синим – нормальные. Можно увидеть количество сайтов, удовлетворяющих тому или иному значению признака. Подробнее об этих критериях:

А. 1 – форма сохраняется на данном сайте, -1 – форма отправляется на другой сайт;

Б. 1 – защищенный протокол https, -1 – незащищенный протокол http;

В. 1 – наличие у домена верхнего уровня DNSSEC-записей, -1 – отсутствие этих записей, 0 – нет информации;

Г. 1 – тело сайта не является картинкой, -1 – тело сайта картинка

Д. 1 – домен находится в HSTS Preload List, 0 – его там нет.

Аналогично предыдущему рисунку на Рис.3 показана статистика, но уже численных признаков 6, 7, 8, 9, 10 (Д, Г, А, В, Б соответственно). Для отображения этих графиков применялся другой подход. За основу был взят алгоритм построения деревьев решений C4.5. По каждому из этих признаков берется разделяющий критерий, благодаря которому множество делится на два подмножества, дающих максимальный прирост информации.

1.2 Описание алгоритма поиска опечаток

Для поиска опечаток применялась Hunspell — свободная библиотека для проверки орфографии. На вход она принимает два файла с расширениями .dic и .aff с наборами орфографических и морфологических правил. Для внедрения этой программы в данную работу, она была адаптирована для использования в расширениях.

На данный момент, расширение анализирует только русские и английские слова на сайтах. При этом перед обработкой текста отбрасываются все слова, начинающиеся с большой буквы, для того, чтобы предотвратить проверку правописания имен собственных, так как словари Hunspell не предусматривают правила для имен собственных. Если вдруг имя собственное было написано с маленькой буквы, то это слово будет рассматривается как опечатка. Также отбрасываются все числа, знаки препинания, ссылки и специальные символы.

После того как были получены все слова, производится проверка каждого слова на правильность орфографии. При этом, если программа выдает слова для замены, значит оно расценивается как опечатка, если же замен не найдено, то считается, что это слово не содержится в словаре и не является опечаткой. Общее количество слов считается суммой правильных слов, слов, не содержащихся в словаре, и опечаток.

Глава 2. Нейросетевой подход

В качестве механизма взаимодействия критериев между собой было решено взять немногослойную нейронную сеть, которая после обучения настроит веса каждого критерия.

Вся работа с глубоким обучением проходила в три основных этапа:

- сбор данных для обучения;
- обучение и тестирование нейронной сети на языке Python;
- обучение нейронной сети на языке JavaScript на основе данных полученных с обучения на языке Python.

Далее рассмотрим все этапы подробнее.

2.1. Данные

На основе уже определенных критериев были собраны данные с 230 сайтов, среди которых ровно половина является фишинговыми, а остальные – нормальными. Все данные были визуализированы в 2D измерении с помощью алгоритма нелинейного снижения размерности и визуализации многомерных переменных t-SNE, взятого из библиотеки sklearn. Эту визуализацию можно видеть на Рис.4. На Рис.5 показан датасет сформированный из собранных данных. В колонке «Значение» единица означает, что сайт фишинговый, а ноль – обратное.

В итоге данные делятся на три основные выборки с определенными размерами:

1. тренировочная выборка размером 200 сайтов;
2. тестовая – 20 сайтов;
3. валидационная – 10 сайтов.

В каждой такой выборке половина сайтов является фишинговыми.

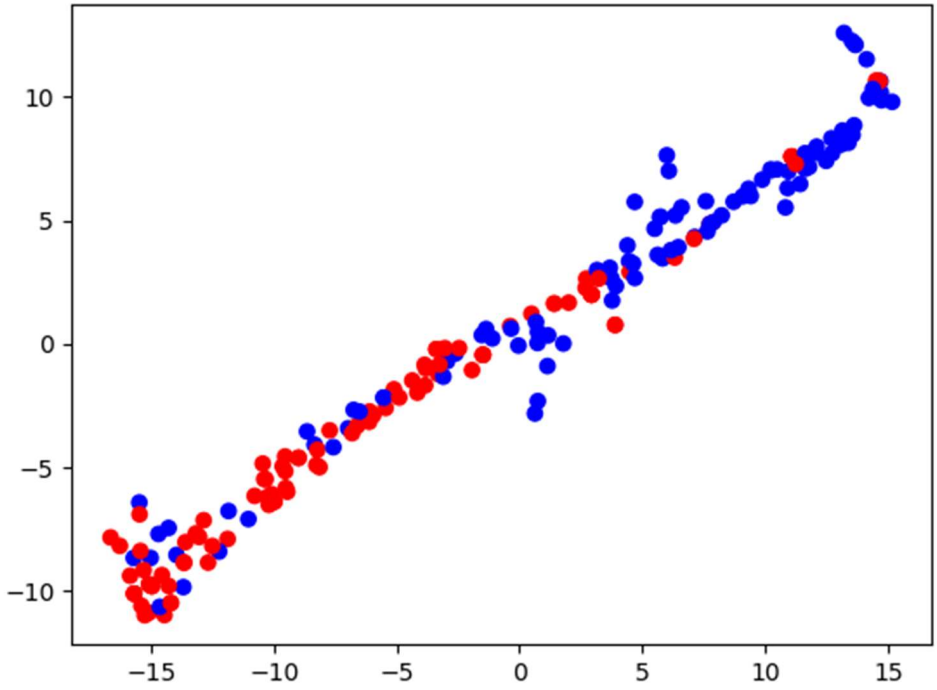


Рис.4 Визуализация данных: а) красные точки — фишиговые сайты, б) синие — нормальные сайты

	DNSSEC	Домен	Протокол	Тело img	Опечатки/Слова	Форма	Поз.ссылки/Нег.ссылки	Ссылки стилистики	Опечатки	Слова	Значение
https://azzouzi.org/	1.0	0.0	-1.0	1.0	0.000000	-1.0	0.000000	0.000000	0.0	10.0	1.0
https://www.phpprime.com/	1.0	0.0	-1.0	1.0	0.000000	1.0	0.000000	0.000000	0.0	10.0	1.0
https://getmdl.io/	1.0	1.0	1.0	1.0	0.010753	1.0	7.200000	4.000000	2.0	186.0	0.0
https://violetraven.co.uk/	1.0	1.0	1.0	1.0	0.000000	1.0	10.384615	57.500000	0.0	98.0	0.0
https://www.dallastxdivorce.com/	1.0	0.0	1.0	1.0	0.110465	-1.0	0.000000	12.000000	57.0	516.0	1.0
https://www.celestialdental.com/	1.0	0.0	1.0	1.0	-1.000000	1.0	-1.000000	1.000000	0.0	0.0	1.0
https://www.google.ro/	1.0	0.0	1.0	1.0	0.013115	1.0	24.833333	5.500000	4.0	305.0	0.0
https://www.healthcare.gov/	1.0	1.0	1.0	1.0	0.000000	1.0	46.000000	0.000000	0.0	15.0	0.0
https://delam.site/	1.0	0.0	-1.0	1.0	0.053571	-1.0	0.000000	1.272727	3.0	56.0	1.0
https://www.hlavi.hu/	1.0	0.0	1.0	1.0	0.000000	1.0	7.700000	1.000000	0.0	147.0	0.0

Рис.6 Датасет

2.2 Обучение нейронной сети на языке Python

Для подбора оптимальных параметров для дальнейшего обучения на языке JavaScript нейронная сеть сначала была написана на языке Python с использованием библиотеки Pytorch.

Так как датасет достаточно маленький, была выбрана архитектура с количеством параметром меньше количества тренировочных данных. На Рис.6 изображена данная архитектура, включающая в себя входной слой размером 10 нейронов, выходной слой с размером 1 нейрон и 2 линейных скрытых слоя с размерами по 10 нейронов.

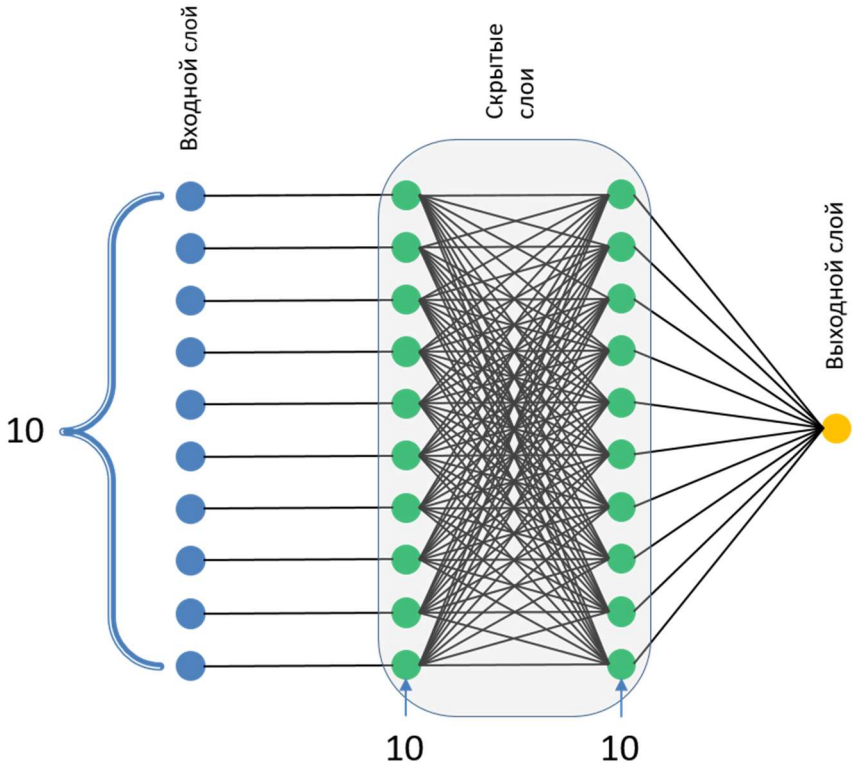


Рис.6 Архитектура нейронной сети

Основные параметры обучения:

1. функция потерь – бинарная кросс энтропия:

$$l(x,y) = L = \{l_1, \dots, l_n\}^T, l_n = -w_n[y_n \cdot \log_2(x_n) + (1 - y_n) \cdot \log_2(1 - x_n)]$$

$$l(x,y) = \begin{cases} mean(L), if\ reduction = 'mean'; \\ sum(L), if\ reudction = 'sum'. \end{cases}$$

(По умолчанию $reduction = 'mean'$)

2. алгоритм оптимизации Adam:
    - скорость обучения — 1e-3;
3. batch size — 20 сайтов;
4. максимальное количество эпох — 5000;
5. минимальная ошибка — 5e-3.

Всего обучалось 5 моделей, для каждой датасет делился на три основные выборки, рандомно перемешиваясь, но сохраняя принцип, что в каждой выборке половина сайтов являются фишинговыми. Это сделано для того, чтобы понять на каком наборе данных лучше всего тренировать нейронную сеть на языке JavaScript.

Таблица 1. Результат обучения нейронных сетей

Имя модели	Эпоха	Ошибка	Точность на валидационной выборке	Точность на тестовой выборке
Model №1	100	0.1663	0.9	0.95
	1000	0.02222	0.9	1.0
	2000	0.00619	1.0	1.0
	2149	0.00499	1.0	1.0
Model №2	100	0.22227	0.8	0.9
	1000	0.02195	0.9	0.85
	2000	0.01317	1.0	0.85
	2455	0.00499	1.0	0.85
Model №3	100	0.15588	0.5	1.0
	1000	0.05179	0.8	1.0
	2000	0.02189	0.9	0.9
	2803	0.00499	0.9	0.9
Model №4	100	0.32357	1.0	0.95
	1000	0.09417	1.0	0.95
	2000	0.02986	1.0	1.0
	3000	0.0101	1.0	0.9
	4000	0.00577	1.0	0.9
	4220	0.00499	1.0	0.9
Model №5	100	0.24863	1.0	0.95
	1000	0.03014	0.8	0.95
	1673	0.005	1.0	0.95

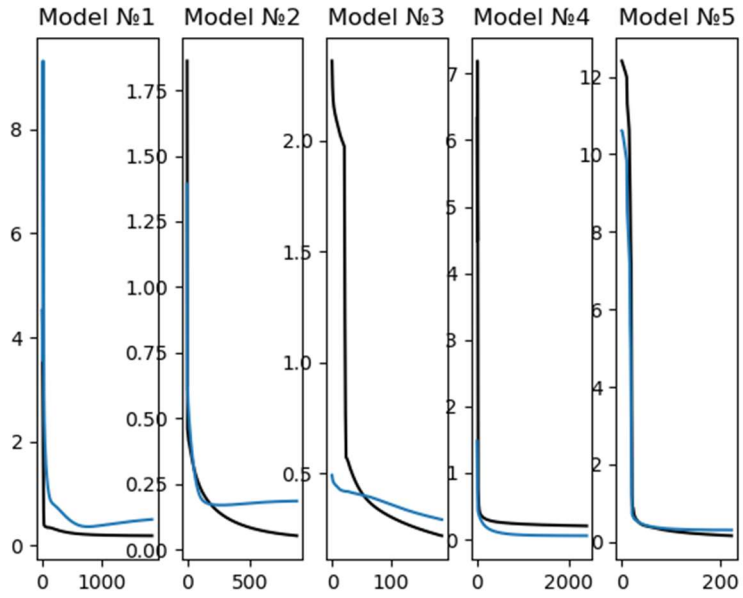


Рис.7 Графики функции ошибки на тренировочной и валидационной выборках

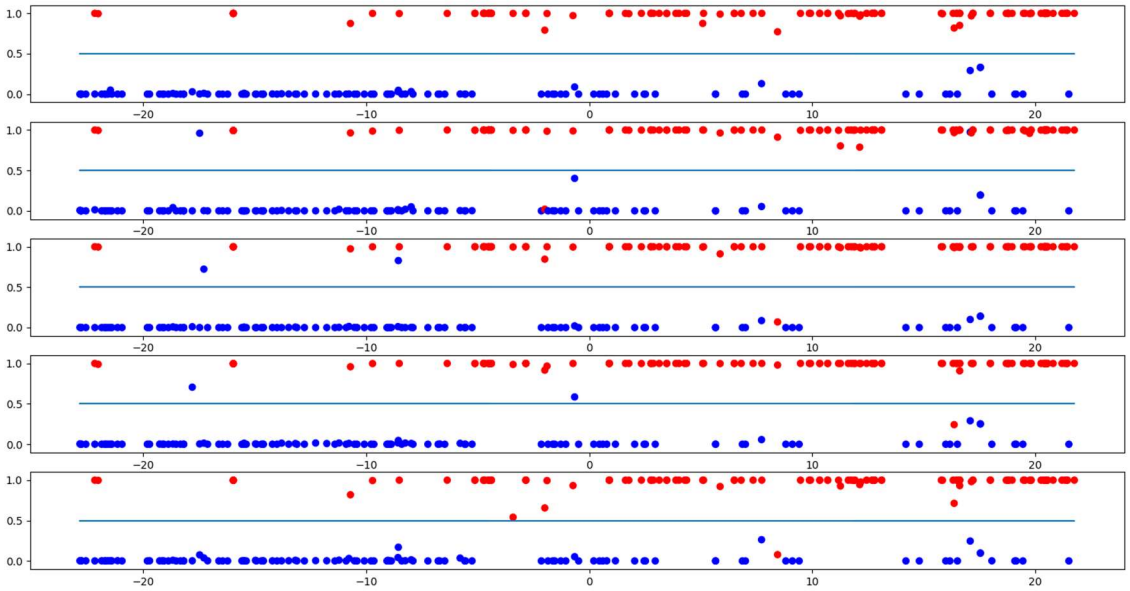


Рис.8 Визуализация работы моделей на всем множестве

В Таблице 1 и Рис.7 представлены результаты обучения 5 нейронных сетей. Из Рис.7 можно заметить, что на второй модели мы встретили переобучение. На всех остальных отклонение незначительное, но на первой модели лучше взять количество эпох, соответствующее значению на «локте» (или перегибе) функции потерь на валидационной выборке. Рис.8 показывает, что первая модель лучше всего разделяет всё множество и имеет наибольшую точность предсказания, а

значит в дальнейшем для обучения нейронной сети на языке JavaScript мы будем использовать наборы данных, которые использовались для обучения первой модели.

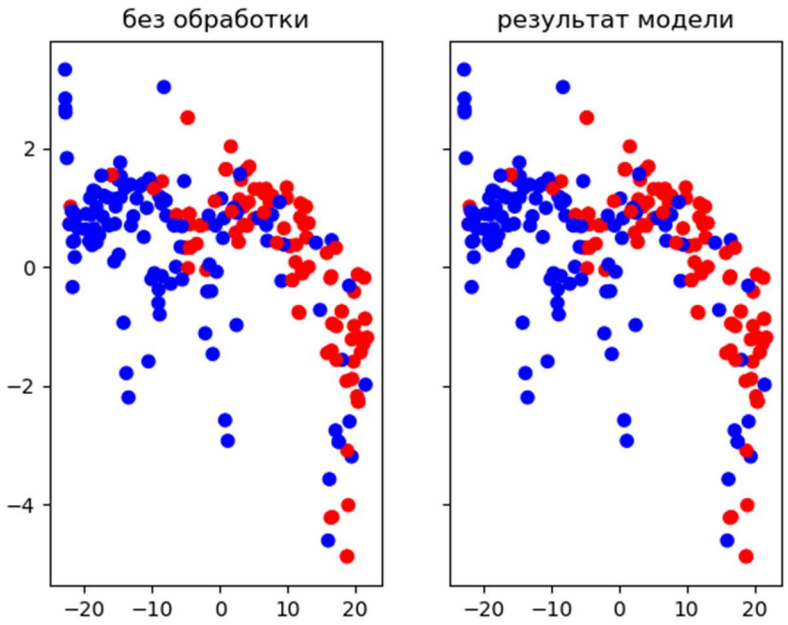


Рис.9 Визуализированный результат работы первой модели на всех данных

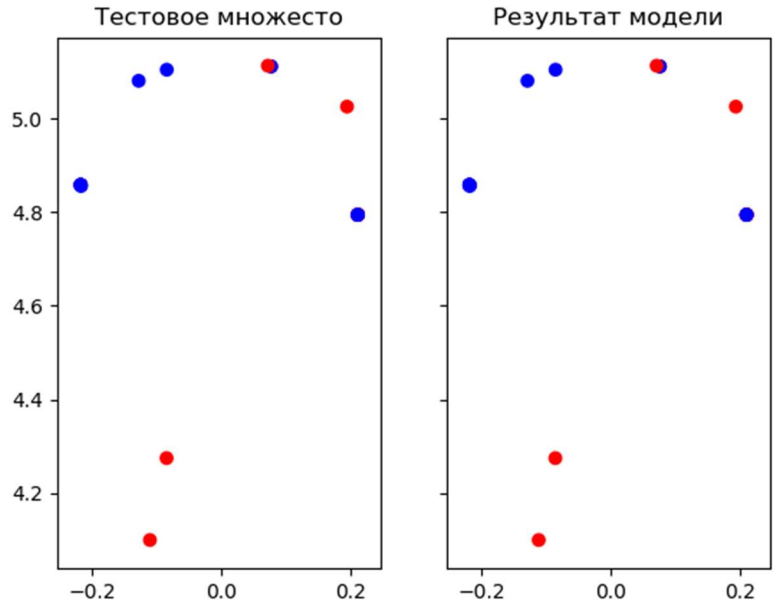


Рис.10 Визуализированный результат работы первой модели на произвольной тестовой выборке

2.3 Обучение нейронной сети на языке JavaScript

На JavaScript нейронная сеть была написана с помощью библиотеки Brain.js. Архитектура и данные были взяты такие же, как для обучения первой модели, написанной на языке Python. Для обучения использовались те же параметры обучения, за исключением функции потерь. В предыдущем параграфе было замечено, что у первой модели возможно небольшое переобучение, поэтому в качестве функции потерь была использована среднеквадратичная ошибка (MSE):

$$l(x, y) = L = \{l_1, \dots, l_n\}^T,\ l_n = (x_n - y_n)^2,$$
$$l(x, y) = mean(L)$$

Таблица 2. Результат обучения нейронной сети на языке JavaScript

Название модели	Эпоха	Ошибка
Model_JS	100	0.02504628191382472
	200	0.018852015170452655
	300	0.014522773979946852
	400	0.01462846398000555
	500	0.007162848323984566

В Таблице 2 показаны результаты обучения нейронной сети, переписанной на JavaScript. При этом точность на тестовой выборке для этой модели равна 1.

Библиотека Brain.js была адаптирована для работы в расширении. Из нее взяты только функции необходимые для предсказания результатов и загрузки весов уже обученной модели.

Глава 3. Работа расширения EPSDfree

3.1 Описание работы расширения

Рис.11 Логотип расширения

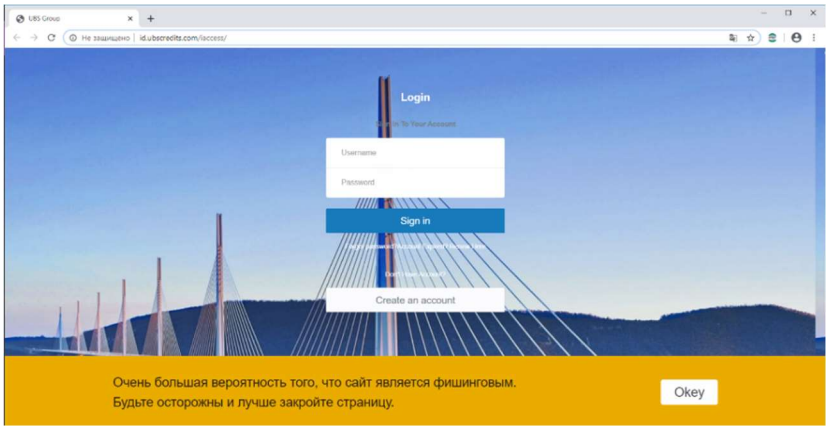
Данное расширение получило название EPSDfree, что расшифровывается как Extension for Phishing Sites Detection free. Его логотип можно увидеть на Рис.11.

Рис.12 Схема работы расширения

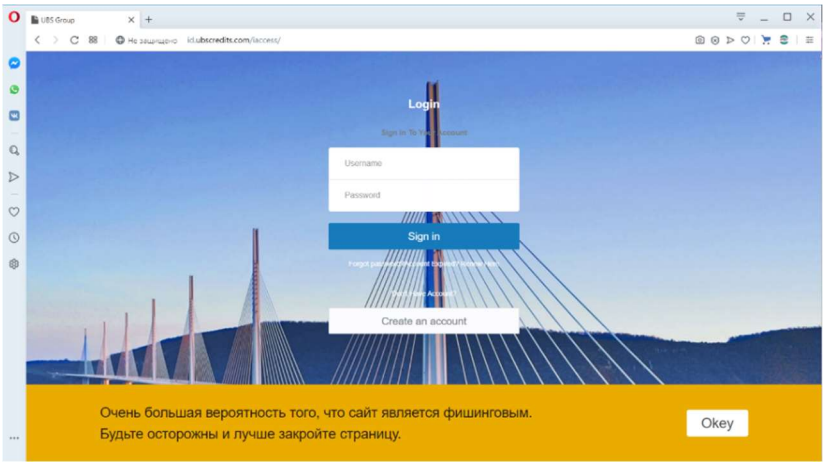
На Рис.12 представлена схема работы данного расширения. Как только загружается сайт, DOM его страницы пропускается через расширение. Из него выделяются все нужные для критериев компоненты, например, слова, ссылки и так

далее. Как только получена вся необходимая информация, она прогоняется через критерии. После этого имеются результаты 10 критериев, которые нужно как-то связать между собой. Для этого используется нейронная сеть. После обработки данных нейронной сетью, получается результат в виде вероятности принадлежности сайта к фишинговому. Если эта вероятность меньше 0.5, то расширение ничего не делает, позволяя пользователю беспрепятственно работать с сайтом. Если же результат больше 0.5, то расширение создает и встраивает в DOM страницы всплывающее окно желтого цвета внизу видимого окна с предупреждением.

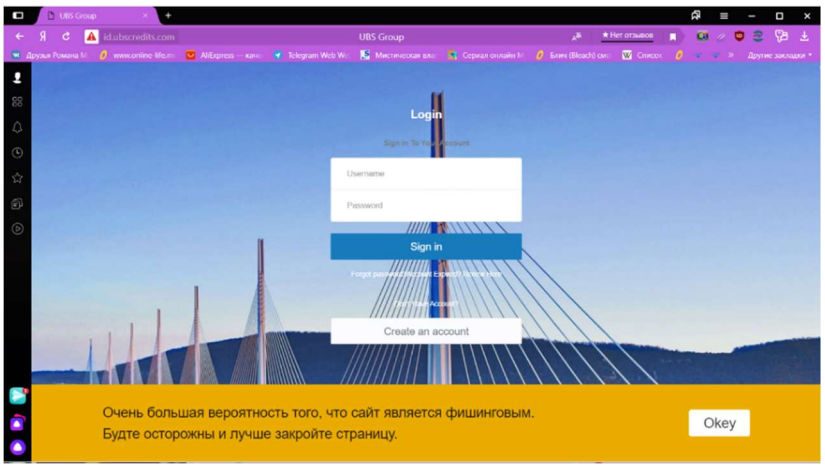
3.2 Результат работы расширения в браузерах



а)



б)



в)

Рис 13. Пример работы расширения на фишинговом сайте в браузерах а) Google Chrome, б) Opera и в) Яндекс

Файл manifest.json написан по стандартам для расширений Google Chrome, которые так же подходят для браузеров Opera и Яндекс. В самом коде есть фрагмент, представленный на Рис.14, который позволяет работать основным функциям в данных браузерах.

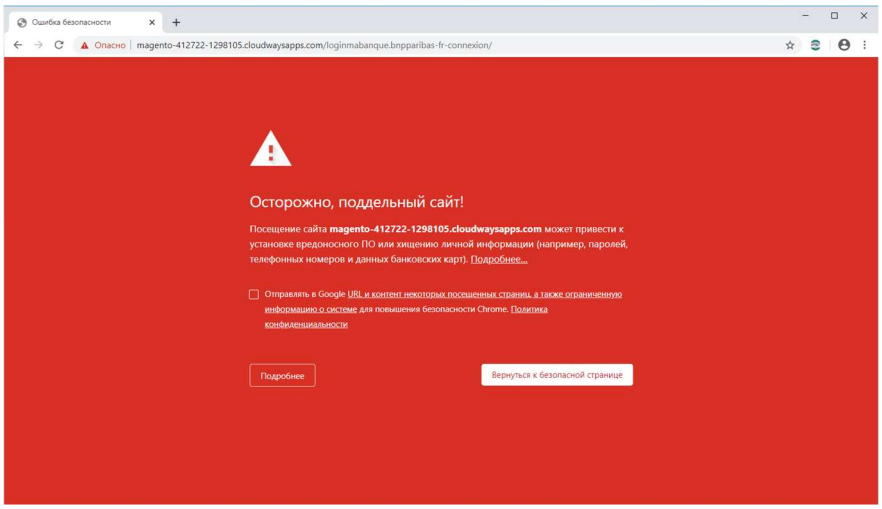
```
window.browser = (function () {
    return window.msBrowser ||
        window.browser ||
        window.chrome;
})();
```

Рис.14 Фрагмент кода из файла appraiser.js

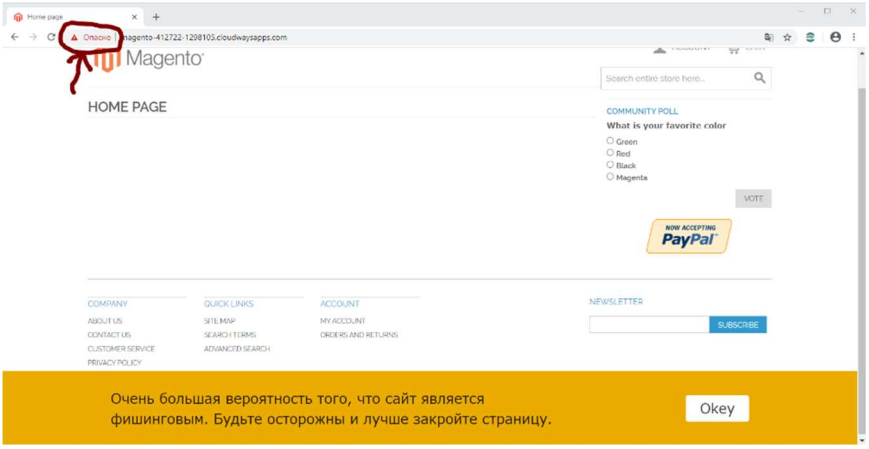
Теперь для наглядности плюсов работы данного расширения рассмотрим пример сайта, который уже есть в черном списке браузера Google Chrome. Мы скопируем его и запустим на другом домене, которого нет в черном списке. На Рис.15а показано предупреждение при попытке перейти на сайт http://magento-412722-1298105.cloudwaysapps.com/. Это означает что сайт уже внесен в черный

список. Проигнорировав предупреждение, заходим на страницу сайта Рис.15б. Видно, что в левом верхнем углу есть значок «Опасно», который сообщает, что сайт является поддельным, и расширение тоже распознает его как фишинговый.

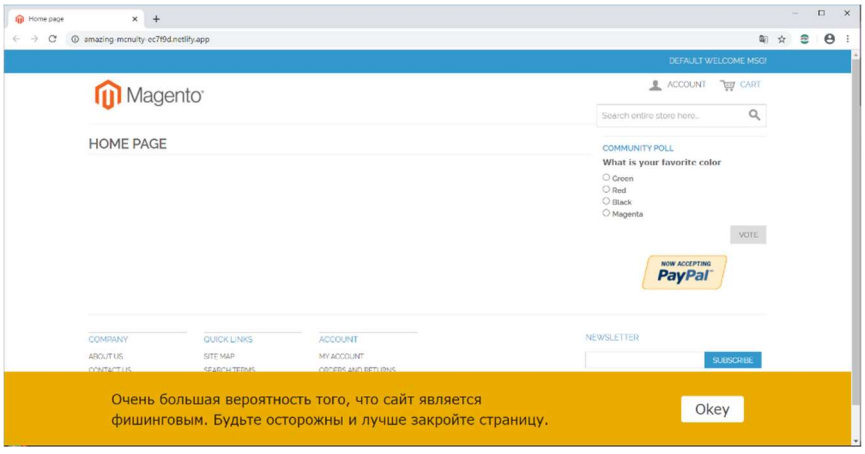
Теперь копируем сайт со всеми ссылками и зависимостями и запускаем его на новом домене (Рис.15в). Можно сразу заметить, что в левом верхнем углу нету значка «Опасно», а значит браузер уже не распознает этот сайт как поддельный, но EPSD все равно определяет его как фишинговый.



а)



б)



в)

Рис.15 Сравнение расширения с антифишинговой защитой браузера Google Chrome

Заключение

Все поставленные задачи были выполнены. В процессе работы стало понятно, что это расширение является хорошей основой для будущих исследований и улучшений. Как пример, с его помощью можно будет собирать данные с различных сайтов. Если пользователь соглашается или отрицает статус сайта, то эти данные сохраняются, и на их основе вновь обучается нейронная сеть. В этом случае возможны два варианта реализации:

1. У каждого пользователя будет своя нейронная сеть, обучаемая на их истории.
2. Одна общая нейронная сеть, для которой выделяется сервер для хранения отзывов других пользователей. На основе всех получаемых данных и будет происходить обучение.

И все-таки, подводя итоги, удалось создать такое браузерное расширение, которое может компенсировать некоторые минусы антифишиговой зашиты популярных браузеров, тем самым облегчить жизнь рядовому пользователю Интернета, обезопасив его от кражи данных.

Список литературы

1. Безмалый В. Современные браузеры. Защита от фишинга [Электронный ресурс]. URL: https://www.osp.ru/pcworld/2011/07/13009498/.

2. Hill R. uBlock Wiki [Электронный ресурс]. URL: https://github.com/gorhill/uBlock/wiki.

3. Vayansky I., Kumar S. Phishing – challenges and solutions [Электронный ресурс]. URL: https://www.researchgate.net/publication/322823383_Phishing_-_challenges_and_solutions.

4. Нежников С. Фишинг в интернете: как не попасть в сети мошенников: [Электронный ресурс]. URL: https://sales-generator.ru/blog/fishing-v-internete/.

5. PyTorch documentation — PyTorch master documentation [Электронный ресурс]. URL: https://pytorch.org/docs/stable/index.html.

6. Brain.js documentation [Электронный ресурс]. URL: https://github.com/BrainJS/brain.js#about.

7. OpenPhish - Phishing Intelligence [Электронный ресурс]. URL: https://openphish.com/

Приложения

1. Исходный код расширения: https://github.com/Romamart/EPSDfree.
2. Репозиторий со всеми исследованиями: https://github.com/Romamart/DipTest.