Анализ возможностей использования технологий машинного обучения для выявления атак на веб-приложения

Е.А. Юдова, О.Р. Лапонина

Аннотация **– Данная статья посвящена анализу возможности обнаружения атак на веб-приложения с использованием алгоритмов машинного обучения. Рассматривается обучение с учителем. В качестве набора данных используется выборка HTTP DATASET CSIC 2010г. Набор данных создавался автоматически и содержит 36 000 нормальных запросов и более 25 000 аномальных. Все HTTP-запросы помечены как нормальные или аномальные. Аномальные данные содержат такие атаки, как SQL-инъекция, переполнение буфера, сбор информации, раскрытие файлов, CRLF-инъекция, межсайтовое выполнение сценариев (XSS), включение на стороне сервера, подделка параметров и т.д. Используются обучающая и тестовая выборки для анализа эффективности различных алгоритмов машинного обучения, используемых для классификации трафика. Реализовано преобразование всех текстовых значений признаков в численные. Определены метрики качества для пяти алгоритмов машинного обучения, и выбран оптимальный алгоритм, который классифицирует рассматриваемый трафик на аномальный и обычный.**

Ключевые слова **– обучение с учителем, HTTP-уязвимости, метрики качества моделей машинного обучения.**

I. Введение

Вопросы безопасности веб-приложений актуальны по сей день. Не существует универсальных механизмов защиты, а тем более механизмов, которые смогли бы выявлять какие-то новые угрозы для веб-приложений. На сегодняшний день достаточно актуальной задачей является автоматическая классификация большого объема веб-трафика на нормальный и аномальный. Можно с достаточно большой вероятностью предсказать попытку совершения атаки, если анализировать различные параметры трафика между пользовательским агентом и сервером. Признаком совершения атаки могут быть характерные заголовки в запросе, дополнительные файлы или параметры в ссылке, манипуляции с cookie и т.п.

Подобные исследования проводились с использованием различных выборок.

Статья получена 31 октября 2021.

Е.А. Юдова – МГУ имени М.В. Ломоносова (email: KateCate.KY@gmail.com).

О.Р. Лапонина – МГУ имени М.В. Ломоносова (email: laponina@oit.cmc.msu.ru).

В работе [2] использовался набор размеченных данных CICIDS 2017, состоящий из реальных данных, полученных с компьютеров с современными операционными системами. Для применения методов машинного обучения было выделено 85 признаков атак.

В данной статье исследования данных проводились на основе схемы обучения с учителем, предложенной Sebastian Raschka [1].

II. Постановка задачи

A. Цели работы

1. Проанализировать алгоритмы машинного обучения для анализа HTTP-трафика и разработать метрики оценки качества.
2. Выбрать и реализовать оптимальный алгоритм машинного обучения для классификации трафика.

B. Задачи работы

1. Подобрать тренировочную и тестовую выборки. Минимизировать количество признаков, которые будут анализироваться для выявления аномального трафика.
2. Выполнить предварительную обработку данных.
3. Реализовать несколько моделей машинного обучения и выделить самую эффективную из них, с помощью которой будет классифицироваться трафик.
4. Обучить выбранную модель и проверить производительность и качество данной модели на тестовой выборке.

III. Классификация трафика

A. Описание и предварительная обработка набора данных

В данной статье рассматривается обучение с учителем, поэтому необходимо определить обучающую выборку, которая содержит как обычный трафик, так и аномальный трафик. В статье использовалась достаточно обширная выборка из [5]. Данная выборка HTTP DATASET CSIC 2010г. создавалась в искусственной среде, где имитировали обычный и аномальный трафики.

Набор данных HTTP CSIC 2010 содержит трафик веб-приложения электронной коммерции. В этом веб-

приложении пользователи могут покупать товары с помощью корзины покупок и регистрироваться, указав некоторую личную информацию.

Набор данных создается автоматически и содержит 36 000 нормальных запросов и более 25 000 аномальных. Все HTTP-запросы помечаются как нормальные или аномальные. Аномальные данные содержат такие атаки, как SQL-инъекция, переполнение буфера, сбор информации, раскрытие файлов, CRLF-инъекция, межсайтовый скриптинг (XSS), включение на стороне сервера, подделка параметров и т.д.

Трафик генерируется следующим образом.

Собираются данные, касающиеся всех параметров веб-приложения (имена, фамилии, адреса и т.д.) из базы данных приложения. Собранные значения параметров записываются в две базы данных: в одну значения из нормальных запросов, в другую из аномальных. Кроме того, имеется список всех общедоступных страниц веб-приложения.

Затем для каждой веб-страницы генерируются нормальные и аномальные запросы. В случае, если нормальные запросы имеют параметры, значения параметров заполняются данными, взятыми из базы данных нормальных значений случайным образом. Аналогично для аномальных запросов значения параметров берутся из базы данных аномальных значений.

Существуют три типа аномальных запросов (три типа атак):

1) Статические атаки, которые пытаются запросить скрытые или несуществующие ресурсы или устаревшие параметры, такие как идентификаторы сессии, которые указываются с помощью перезаписи URL. Скрытые файлы включают файлы конфигурации, файлы по умолчанию и т.д.
2) Динамические атаки, которые изменяют корректные аргументы запроса: SQL-инъекция, CRLF-инъекция, межсайтовый скриптинг, переполнение буфера и т.д.
3) Непреднамеренные некорректные запросы. Эти запросы не являются атаками, однако они не соответствуют нормальному поведению веб-приложения и не имеют той же структуры, что и обычные значения параметров (например, телефонный номер, состоящий из букв).

Атаки создавались с помощью таких инструментов, как ZAP (Paros) [13] и W3AF [14].

Данные межсетевые экраны используют аномальный подход для обнаружения атак, то есть им должно быть известно нормальное поведение веб-приложения, а поведение, отличное от нормального, считается атакой. При таком подходе для фазы обучения необходим только нормальный трафик.

Набор данных разделен на три разных подмножества. Одно подмножество используется в фазе обучения и содержит только нормальный трафик. Два других подмножества используются в фазе тестирования, одно содержит нормальный трафик, другое аномальный (вредоносный).

В используемый нами набор данных CSIC 2010 разработчиками было добавлено два параметра.

Во-первых, параметр **index**, который используется для отслеживания номера HTTP-пакета. Разработчики набора данных разделили каждый HTTP-пакет на отдельные записи, так как URL пакета мог иметь несколько значимых для классификации данных формата <ключ> = <значение>. В результате index позволяет определить принадлежность записи в наборе данных исходному HTTP-пакету.

Во-вторых, параметр **label**, который определяет пакет как нормальный (значение равно 1) или аномальный (значение рано 0).

Также используемый набор данных содержит перечисленные ниже заголовки HTTP-запроса и HTTP-ответа.

Для выполнения предварительной обработки данных, нужно понимать семантику этих заголовков, так как в дальнейшем они будут использоваться как признаки в выборке данных.

Method - метод запроса клиента к серверу.

Url - запрашиваемый ресурс.

User-Agent - пользовательский агент, то есть браузер пользователя.

Pragma - запрет кэширования, используемый протоколом HTTP/1.0.

Cache-Control - HTTP-заголовок, используемый протоколом HTTP/1.1, который определяет количество времени и способ кэширования файла.

Accept - типы контента, выраженные как MIME-типы, которые клиент может принимать.

Accept-Encoding – указывает, было ли возвращаемое сожержимое сжато или нет.

Accept-Charset – список поддерживаемых клиентом кодировок содержимого.

Accept-Language – список поддерживаемых естественных языков.

Host – доменное имя хоста, которому предназначен запрос (и, как правило, номер порта).

Connection - заголовок, который определяет, остаётся ли сетевое соединение активным после завершения текущей транзакции или запроса.

Content-Length - размер тела объекта.

Content-Type - заголовок, который используется для того, чтобы определить MIME-тип ресурса.

Cookie - небольшой фрагмент данных, который отправляется сервером браузеру пользователя. Браузер может его сохранить и обязан отсылать обратно в новом запросе к данному серверу (в формате <key>=<value>).

Payload – отправляется как обычный текст (формат: <key>=<value>).

На этапе предварительной обработки данных были выявлены признаки, которые имеют неопределенные (пропущенные) значения (NaN): contentLength (123450 пропущенных данных), contentType (123450 пропущенных данных), payload (25491 пропущенных

данных). У признаков contentLength и contentType есть пропущенные значения, так как данные заголовки присутствуют только в POST запросах. Следовательно, для нас не критично отсутствие части значений у этих признаков. Признак payload имеет пропущенные значения из-за того, что не во всех запросах передается текстовая информация (в данном случае аномальный трафик может определяться другим признаком).

После составления списка уникальных значений для различных признаков из выборки были исключены признаки, которые принимают всегда одно и то же значение. Исключены признаки: connection, userAgent, protocol, pragma, cacheControl, acceptLanguage, accept, acceptEncoding, acceptCharset, contentType. Большая часть признаков имеет одинаковые значения, так как взаимодействие сервера и клиента производилось в одной и той же искусственной среде.

Далее признаки label, method и host были достаточно просто приведены к численным характеристикам, так как в используемом наборе данных принимают только два уникальных значения. Для обработки cookie был создан список уникальных значений (cook), каждое значение этого параметра в столбце DataFrame заменилось индексом соответствующего параметра в списке 'cook'.

Обработка Payload производилась другим способом. Для значения левой части данного запроса был создан список и проделаны шаги, аналогично работе с характеристикой cookie. Создан список уникальных слов (ke) и колонка с новой характеристикой - key, в которой значения соответствовали порядковому номеру параметра в списке ke. Если такой параметр отсутствует, то соответствующее значение равно -1. Было посчитано количество символов '%' и слов в правой части текстового запроса. Затем реализация алгоритма поиска наиболее частых слов в правой части payload позволила выделить слова, которые могут быть частью атаки в данных запросах. Это слова: cookie, delay, cmd, waitfor, exec, from, set, del, alert, any, sessionid, select, user, url, script, document, confirm, hacker, and, proxy, table, drop, id, passwd, inject, steal, bin, dir, include. В соответствующих столбцах указано количество данных слов в правой части запроса payload.

Последний признак, который требует обработки - url. В начале пришлось выяснить максимальную длину пути до файла, она получилась равна 4, а количество файлов, которые имеют разные разрешения - 3. Соответственно, получились такие колонки с признаками: хост, 4 директории, 3 имени файла, 3 типа файла; тип файла определяется по его расширению. Имена файлов и их разрешения начинали заполняться с конца, при этом использовался метод, который был ранее описан для cookie.

После всех преобразований и удаления столбцов url и payload, получилась требуемая выборка в формате .csv. Далее выполнено разбиение получившейся выборки на тестовую и обучающую, где доля тестового набора составляет 0.3 от всей выборки.

B. Уменьшение количества анализируемых признаков

Проведем анализ важности признаков, чтобы исключить признаки, значимость которых намного меньше остальных. Анализ важности производится с помощью классификаторов Дерева решений и Случайного леса (Decision Tree Classifier и Random Forest Classifier). Рассмотрим данные классификаторы.

1. Классификатор Дерево решений (Decision Tree) [6]. На основе доступного набора данных Дерево решений изучает иерархию if/else, которая в итоге приводит к принятию решения. Данный классификатор широко используются в машинном обучении.

В данном случае стоит задача - восстановить зависимость $y : X \to Y,\ |Y| < \infty$ по точкам обучающей выборки $(x_i, y_i), i = 1 \ldots l$.

Приведем более формальное описание данного классификатора.

Дано: векторы $x_i = (x_i^1, \ldots, x_i^n)$ - объекты обучающей выборки,

$y_i = y(x_i)$, $i = 1 \ldots l$ - классификации (ответы учителя).

$$\begin{pmatrix} x_1^1 & \cdots & x_1^n \\ \ldots & \ldots & \ldots \\ x_l^1 & \cdots & x_l^n \end{pmatrix} \to \begin{pmatrix} y_1 \\ \ldots \\ y_l \end{pmatrix} \quad (1)$$

Найти: функцию $a(x)$, способную классифицировать объекты произвольной тестовой выборки $\tilde{x}_i = (\tilde{x}_i^1, \ldots, \tilde{x}_i^n)$, $i = 1, k$.

$$\begin{pmatrix} \tilde{x}_1^1 & \cdots & \tilde{x}_1^n \\ \ldots & \ldots & \ldots \\ \tilde{x}_k^1 & \cdots & \tilde{x}_k^n \end{pmatrix} \to \begin{pmatrix} a(\tilde{x}_1) \\ \ldots \\ a(\tilde{x}_k) \end{pmatrix} \quad (2)$$

Дерево решений состоит из вершин, в которых записаны объекты обучающей выборки (признаки), и листьев, в которых записаны результаты классификации. Алгоритм классификации $a(x)$ определяется с помощью бинарного дерева решений, которое создается на основе обучающей выборки.

2. Классификатор Случайный лес (Random Forest) [7]. Это мета-оценка, которая соответствует ряду классификаторов дерева решений на различных подвыборках набора данных и использует усреднение для повышения точности прогнозирования и контроля над подгонкой ответов.

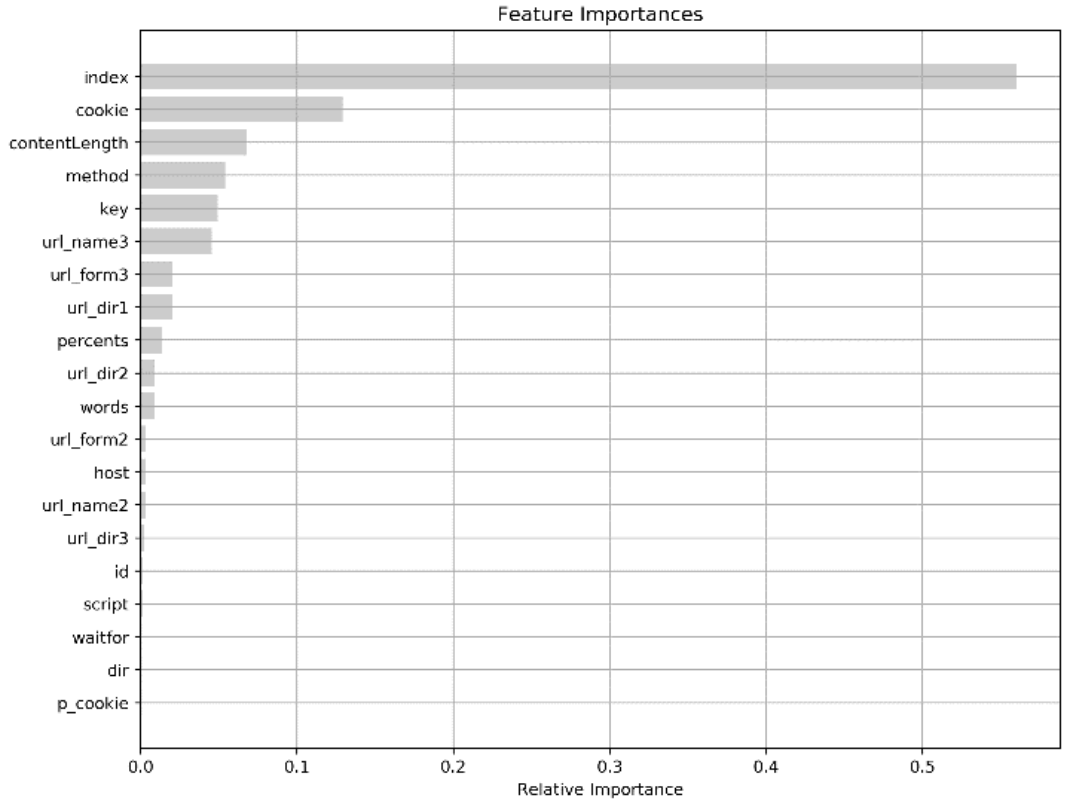

Рис. 1. Важность признаков для классификатора Случайный лес

После реализации, обучения и проверки корректности обоих методов получился признак, который превалирует над другими – index (коэффициент его важности составляет 0.56 и 0.57 для разных моделей), данные для случайного леса можно увидеть на рисунке 1. Так как данный параметр используется для отслеживания номера HTTP-пакета, то его можно удалить из признаков, по значениям которых будет приниматься решение.

3. Корреляция Пирсона

Для более эффективного отбора признаков используем корреляцию Пирсона. Она позволяет определить наличие или отсутствие линейной связи между двумя численными характеристиками. Коэффициент линейной корреляции Пирсона:

$$r = \frac{\overline{cov}(X+Y)}{s_x s_y} \quad (4)$$

$$\overline{cov}(X+Y) = \frac{1}{n-1}\sum_{i=1}(x_i - \tilde{x})(y_i - \tilde{y}) \quad (5)$$

- выборочный коэффициент ковариации

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}(x_i - \tilde{x})^2} \quad (6)$$

$$s_y = \sqrt{\frac{1}{n-1}\sum_{i=1}(y_i - \tilde{y})^2} \quad (7)$$

- выборочные средние квадратичные отклонения, где n - размер выборки, а x_i, y_i - числовые значения. После подстановки и преобразований получается:

$$r = \frac{\sum_{i=1}^{n}(x_i - \tilde{x})(y_i - \tilde{y})}{\sqrt{[\sum_{i=1}^{n}(x_i - \tilde{x})^2]\cdot[\sum_{i=1}^{n}(y_i - \tilde{y})^2]}} \quad (8)$$

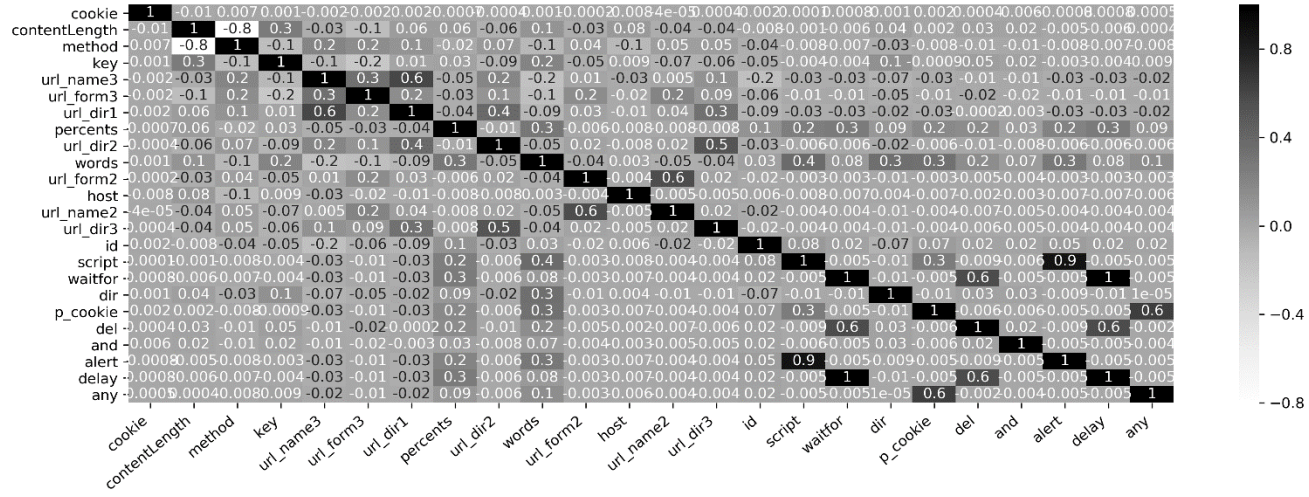

Рис. 2. Коэффициенты корреляции Пирсона

Величина линейного коэффициента корреляции Пирсона изменяется в пределах от -1 до +1. Чем ближе этот показатель для двух признаков к значению -1, тем меньше данные признаки коррелируют между собой. Аналогично, чем ближе коэффициент корреляции Пирсона к значению +1, тем больше признаки коррелируют друг с другом.

Для рассматриваемой выборки в данной работе получаются различные коэффициенты Пирсона, которые можно увидеть на рисунке 2. Очевидно, что признаки wait for и delay сильно коррелируют между собой, поэтому один из признаков можно удалить из рассматриваемой выборки. Удалим признак delay. Далее выделим 23 самых значимых признака из рассматриваемой выборки данных.

В результате в выборке остались признаки: cookie, contentLength, method, key, url_name3, url_form3, url_dir1, percents, url_dir2, words, url_form2, host, url_name2, url_dir3, id, script, waitfor, dir, p_cookie, del, and, alert, any.

C. Выбор модели машинного обучения

Модели и метрики качества

В качестве метрик качества каждой модели используются следующие показатели.

True Positive (TP) – количество (процент) правильно классифицированного нормального трафика.

True Negative (TN) - количество (процент) правильно классифицированного аномального трафика.

False Negative (FN) - количество (процент) неправильно классифицированного нормального трафика (нормальный трафик классифицирован как аномальный).

False Positive (FP) - количество (процент) неправильно классифицированного аномального трафика (аномальный трафик классифицирован как нормальный).

Рассматриваемые метрики качества [8]:

Accuracy (доля верных ответов). Вычисляется по формуле:

$$Accuracy = \frac{TP}{TP+TN+FP+FN} \quad (9)$$

Precision (точность). Определяет, на сколько можно доверять классификатору. Определяется формулой:

$$Precision = \frac{TP}{TP+FP} \quad (10)$$

Recall (полнота). Показывает, как много объектов класса «аномальный трафик» распознает классификатор. Для вычисления используется формула:

$$Recall = \frac{TP}{TP+FN} \quad (11)$$

F1-metrics - среднее гармоническое точности и полноты, которое вычисляется по формуле:

$$F1 = 2\frac{Precision*Recall}{Precision+Recall} \quad (12)$$

При обучении и тестировании рассматриваемых моделей метрика F1 очень важна. Когда происходит попытка повышения Recall, то метрика Precision падает (увеличивается число ложно-положительных ответов) и наоборот. Поэтому для характеристики классификатора используют метрику качества F1, с помощью которой выявляется оптимальная модель.

В данной работе рассмотрены пять наиболее простых для реализации моделей:

1. **Метод К-ближайших соседей** (k-nearest neighbors method, kNN) [9]. Данный метод решает задачи классификации. Предполагается, что в начале уже имеется определенное количество классифицированных объектов. Требуется выработать правило, которое позволит отнести новый объект к одному из имеющихся классов. В основе данного метода лежит правило: объект принадлежит тому классу, к которому относится большая часть его ближайших соседей.
2. **Бэггинг решающих деревьев** (bagging decision trees, Bagging) [10]. Базовые классификаторы $a_i(x)$ решающих деревьев обучаются независимо по случайным подвыборкам длины ℓ с повторениями. При этом доля объектов,

попадающих в выборку 1-$\mathcal{U}$. Средняя итоговая композиция вычисляется как среднее данных базовых алгоритмов:

(13)

$$a_N(x) = \frac{1}{N}\sum_{i=1}^{N} a_n(x)$$

3. **Бустинг решающих деревьев** (boosting decision trees, AdaBoost) [11]. Устанавливает классификатор на исходный набор данных. Затем производит корректировку весов неправильно классифицированных экземпляров класса за счет помещения дополнительных копий классификатора на тот же набор данных. Корректировка производится таким образом, что последующие классификаторы больше фокусируются на сложных случаях.

Пусть уже построен алгоритм $a(x)$ для решающего дерева, теперь нужно построить такой алгоритм $b(x)$, что выполняется:

$$a(x_i) + b(x_i) = y_i, i = 1 \ldots l \quad (14)$$

Рассмотрим кусочно-постоянный алгоритм, то есть решающее дерево:

$$b(x) = \sum_j \beta_j I[x \in X_j] \quad (15)$$

то есть пространство признаков делится на конечное число областей $\{X_j\}$. В области X_j алгоритм выдает ответ β_j. Каждому листу решающего дерева соответствует своя область. Для данного алгоритма можно независимо для каждой области подбирать значение β_j, минимизируя функцию ошибок (потерь)[1]. Получается следующая функция ошибок:

$$\sum_{i=1}^{m} L(y_i, a(x_i) + \sum_j \beta_j I[x \in X_j]) \to min \quad (16)$$

После выбора и фиксации областей X_j, решается задача минимизации по переменной β_j, ее можно решать независимо для каждой области, тогда получаем такую задачу:

$$\sum_{x_i \in X_j} L(y_i, a(x_i) + \beta_j) \to \min_{\beta_j} \quad (17)$$

4. **Случайный лес** (random forest, RFC) [7]. Данная модель состоит из большого количества отдельных решающих деревьев, которые работают как совокупность методов. Каждое дерево возвращает прогноз класса, и класс с наибольшим количеством голосов становится прогнозом данного леса.

[1] в AdaBoost используется экспоненциальная функция потерь

5. **Логистическая регрессия** (logistic regression, LR) [12]. Метод построения и обучения линейных классификаторов.

Обучающая выборка:

$$X^l = (x_i, y_i)_{i=1}^l, x_i \in R, y_i \in \{-1, +1\} \quad (18)$$

Линейная модель классификации:

$$a(x, w) = sign <x, w> \quad (19)$$

Непрерывная аппроксимация бинарной функции потерь:

$$Q(w) = \sum_{i=1}^{l} I[a(x_i, w)y_i < 0] \leq L(\langle x_i, w\rangle y_i) \to \min_w \quad (20)$$

В данном случае используется логарифмическая функция потерь:

$$L(M) = log(1 + e^{-M}) \quad (21)$$

Метод опорных векторов (support vector machine – SVM) в данной работе не рассматривался, по причине его долгой работы в сравнении с другими моделями.

Выбор параметров и качественные показатели моделей

Для каждой модели производилась подборка оптимальных параметров с помощью инструмента sklearn.model_selection.GridSearchCV. В качестве важных показателей рассматривались такие метрики качества, как F1 и Accuracy. По результатам работы данного метода получились следующие оптимальные параметры для моделей:

1. **kNN**: n_neighbors = 3, weights = 'distance', p = 2.
2. **Bagging**: n_estimators = 150, max_samples = 0.9.
3. **AdaBoost**: n_estimators = 10, learning_rate = 0.1.
4. **RFC**: n_estimators = 50, min_samples_leaf = 3, max_features = 11, max_depth = 25.
5. **LR**: tol = 0.00001, C = 1.0, intercept_scaling = 1.3, max_iter = 50.

На основе полученных данных производилось обучение и тестирование каждой модели на выборке данных. В таблице 1 представлены показатели метрик качества и время работы каждой модели.

Таблица 1. Метрики качества и время работы каждой модели

Model	Accuracy	Precision	Recall	F1	Time, s
kNN	0.914	0.926	0.912	0.919	02.324877
Bagging	0.910	0.914	0.918	0.916	136.016121
AdaBoost	0.901	0.907	0.908	0.907	03.294212
RFC	0.737	0.708	0.859	0.776	18.763315

LR	0.609	0.602	0.785	0.681	04.459142

Принимая во внимание минимальное время выполнения и показатели метрик качества, в качестве основной модели выбран метод k ближайших соседей. Также определены оптимальные параметры для данного метода. В результате получена итоговая модель для анализа данных определенного вида, которые рассматривались в работе. Текст программы находится в открытом доступе, в файле Diplom.ipynb [13].

IV. Выводы

В ходе работы для выборки данных рассмотренного типа [5] был описан механизм сведения текстовых признаков к числовым. Так же, на основе показателей, которые были получены с помощью случайного леса, были отброшены признаки, которые оказывали низкое влияние на классификацию трафика. С помощью корреляции Пирсона произведен отброс коррелирующих между собой признаков. После преобразования выборки данных была выделена оптимальная модель для их обработки и классификации.

Следовательно, для обработки данных такого типа следует использовать **метод K-ближайших соседей** с параметрами: n_neighbors = 3, weights = ’distance’, p = 2. При этом на рассматриваемых данных метрики качества принимают следующие значения: Accuracy = 0.914, Precision = 0.926, Recall = 0.912, F1 = 0.919. Время работы данной модели получилось минимальным, поэтому, можно считать, что это самая оптимальная модель для обработки рассматриваемой выборки данных.

V. Заключение

Что было сделано в ходе работы:

- Выделены основные признаки HTTP-трафика, которые представляют наибольшую важность при классификации трафика.
- Выбраны и обработаны обучающая и тестовая выборки.
- Реализованы различные модели машинного обучения для классификации аномального и не аномального трафиков.
- Выявлен и реализован самый эффективный по времени и точный по метрикам качества алгоритм для рассматриваемой выборки данных.

Библиография

[1] Raschka S. Python machine learning. — Packt publishing ltd, 2015.

[2] Kostas K. Anomaly Detection in Networks Using Machine Learning // Research Proposal. — 2018. — т. 23.

[3] Siles R. Session Management Cheat Sheet // URL https://www.owasp.org/index. php/Session_Management_Cheat_Sheet. — 2014.

[4] Scambray J., Shema M. Hacking Exposed Web Applications. — Brandon A. Nordin, 2002. — с. 416.

[5] Gimenez C. T., Villegas A. P., Maran˜on G. A. HTTP data set CSIC 2010 // Information Security Institute of CSIC (Spanish Research National Council). – 2010.

[6] Кафтанников И. Л., Парасич А. В. Особенности применения деревьев решений в задачах классификации // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. — 2015. — т. 15, № 3.

[7] Чистяков С. П. Случайные леса: обзор // Труды Карельского научного центра Российской академии наук. — 2013. — № 1.

[8] Виноградова Е., Головин Е. Метрики качества алгоритмов машинного обучения в задачах классификации // Научная сессия ГУАП. — 2017. — с. 202—206.

[9] Lantz B. Machine learning with R. — Packt publishing ltd, 2013.

[10] AbellanJ., MasegosaA.R. Bagging decision trees on data sets with classification noise // International Symposium on Foundations of Information and Knowledge Systems. — Springer. 2010. — с. 248—265.

[11] DruckerH.,CortesC. Boosting decision trees //Advances in neural information processing systems. — 1996. — с. 479—485.

[12] Паклин Н. Логистическая регрессия и ROC-анализ - математический аппарат // Режим доступа: https://basegroup. ru/community/articles/logistic. Дата доступа. — 2015. — т. 9.

[13] OWASP Zed Attack Proxy (ZAP). https://www.zaproxy.org/docs/ (2021).

[14] Andrés Riancho: Web Application Attack and Audit Framework. http://w3af.sourceforge.net (2007).

[15] Kate Yudova Diploma github.com/KateYudova/Diplom.

Analysis of the possibilities of using machine learning technologies to detect attacks on web applications

E.A. Yudova, O.R. Laponina

Abstract - **This article is devoted to the analysis of the possibility of detecting attacks on web applications using machine learning algorithms. Supervised learning is considered. A sample of HTTP DATASET CSIC 2010 is used as a data set. The dataset was automatically generated and contains 36,000 normal queries and over 25,000 anomalous. All HTTP requests are marked as normal or abnormal. The anomalous data contains attacks such as SQL injection, buffer overflow, information gathering, file expansion, CRLF injection, cross-site scripting (XSS), server-side inclusion, parameter spoofing, etc. The training and test samples are selected to analyze the effectiveness of various machine learning algorithms used for traffic classification. Conversion of all text values of attributes to numerical ones was realized. The quality metrics for five machine learning algorithms are determined, and the optimal algorithm is selected that classifies the traffic under consideration into abnormal and normal.**

Keywords - **supervised learning, HTTP vulnerabilities, quality metrics of machine learning models.**

REFERENCES

[1] Raschka S. Python machine learning. — Packt publishing ltd, 2015.
[2] Kostas K. Anomaly Detection in Networks Using Machine Learning // Research Proposal. — 2018. — t. 23.
[3] Siles R. Session Management Cheat Sheet // URL https://www.owasp.org/index. php/Session_Management_Cheat_Sheet. — 2014.
[4] Scambray J., Shema M. Hacking Exposed Web Applications. — Brandon A. Nordin, 2002. — s. 416.
[5] Gimenez C. T., Villegas A. P., Maran˜on G. A. HTTP data set CSIC 2010 // Information Security Institute of CSIC (Spanish Research National Council). – 2010.
[6] Kaftannikov I. L., Parasich A. V. Osobennosti primenenija derev'ev reshenij v zadachah klassifikacii // Vestnik Juzhno-Ural'skogo gosudarstvennogo universiteta. Serija: Komp'juternye tehnologii, upravlenie, radiojelektronika. — 2015. — t. 15, # 3.
[7] Chistjakov S. P. Sluchajnye lesa: obzor // Trudy Karel'skogo nauchnogo centra Rossijskoj akademii nauk. — 2013. — # 1.
[8] Vinogradova E., Golovin E. Metriki kachestva algoritmov mashinnogo obuchenija v zadachah klassifikacii // Nauchnaja sessija GUAP. — 2017. — s. 202—206.
[9] Lantz B. Machine learning with R. — Packt publishing ltd, 2013.
[10] AbellanJ., MasegosaA.R. Bagging decision trees on data sets with classification noise // International Symposium on Foundations of Information and Knowledge Systems. — Springer. 2010. — s. 248—265.
[11] DruckerH.,CortesC. Boosting decision trees //Advances in neural information processing systems. — 1996. — s. 479—485.
[12] Paklin N. Logisticheskaja regressija i ROC-analiz - matematicheskij apparat // Rezhim dostupa: https://basegroup. ru/community/articles/logistic. Data dostupa. — 2015. — t. 9.
[13] OWASP Zed Attack Proxy (ZAP). https://www.zaproxy.org/docs/ (2021).
[14] Andrés Riancho: Web Application Attack and Audit Framework. http://w3af.sourceforge.net (2007).
[15] Kate Yudova Diploma github.com/KateYudova/Diplom.