МНОГОКЛАССОВАЯ КЛАССИФИКАЦИЯ СЕТЕВЫХ АТАК НА ИНФОРМАЦИОННЫЕ РЕСУРСЫ МЕТОДАМИ МАШИННОГО ОБУЧЕНИЯ

М.А. Кажемский[1], **О.И. Шелухин**[1*]

[1]Московский технический университет связи и информатики,
Москва, 111024, Российская Федерация
*Адрес для переписки: sheluhin@mail.ru

Информация о статье
УДК 004.732.056
Язык статьи – русский

Ссылка для цитирования: Кажемский М.А., Шелухин О.И. Многоклассовая классификация сетевых атак на информационные ресурсы методами машинного обучения // Труды учебных заведений связи. 2019. Т. 5. № 1. С. 107–115. DOI:10.31854/1813-324X-2019-5-1-107-115

Аннотация: *Рассматривается классификация атак на информационные ресурсы с помощью классических алгоритмов машинного обучения: k-ближайших соседей, множественная логистическая регрессия, «наивный» Байес, опорных векторов, а также с помощью ансамблевых методов: дерево решения, «случайный лес» и Ada Boost. Исследование проводилось на наборе данных NSL-KDD с использованием библиотек языка программирования Python: scikit-learn, pandas и jupyter notebook. Произведена подготовка данных для исследования, а также подобраны оптимальные параметры алгоритмов машинного обучения. Все поля в исследуемом наборе были помечены пятью классами, которые соответствуют четырем категориям атак (DoS, U2R, R2L, Probe) и нормальному трафику (normal). Произведен сравнительный анализ результатов классификации каждого алгоритма по разным метрикам оценки. Сделан вывод о том, что все исследуемые алгоритмы показали недостаточную эффективность в условиях несбалансированности данных. Предложено произвести дополнительные действия над исходным набором для качественной классификации. Наилучшие результаты продемонстрировал алгоритм «случайный лес».*

Ключевые слова: *многоклассовая классификация, машинное обучение, сетевые атаки, классические алгоритмы, ансамблевые методы, scikit-learn, NSL-KDD.*

Постановка задачи

В настоящее время в связи с повсеместным развитием и внедрением информационных систем на предприятиях, растет и число возможных уязвимостей в них. Каждый известный тип уязвимости требует определенной последовательности действий от сотрудников, занимающихся безопасностью на предприятии, поэтому важным этапом в противодействии различным угрозам информационной безопасности является именно ее правильная классификация.

Современные системы обнаружения вторжений (СОВ) хорошо справляются с этими задачами [1, 2]. Вместе с тем наиболее эффективные СОВ являются сложными аппаратно-программными комплексами. Так, например, сетевые СОВ для обнаружения аномалий и последующей классификации часто используют накопление статистических данных и технологию DPI (*от англ.* Deep Packet Inspection – проверка и фильтрация сетевых пакетов по их содержимому), что требует серьезных вычислительных мощностей для анализа трафика.

Вместе с тем широкое распространение в выявлении и классификации аномального поведения в информационных системах получили алгоритмы машинного обучения, способные самообучаться и противостоять новому типу угроз. Многие алгоритмы машинного обучения могут работать достаточно быстро при небольших вычислительных мощностях.

Целью статьи является исследование эффективности многоклассовой классификации сетевых атак на информационные ресурсы с помощью наиболее распространенных алгоритмов машинного обучения.

Предшествующие работы

Большое количество исследований по классификации сетевого трафика проводилось на наборах данных KDD'99 и его модификации NSL-KDD (2014).

В работе [3] приведен обзор актуальных работ, в которых рассматривается классификация трафика на наборе NSL-KDD.

В работах [4–7, 8–9] проводится оценка классификации алгоритмов «*наивный» Байес, дерево решений*, «*случайный лес» и метод опорных векторов.* Но при этом ни в одной из проанализированных работ не рассматривается алгоритм *логистическая регрессия*, хотя в настоящей статье именно данный алгоритм показал достаточно высокое качество классификации.

В работах, где проводится сравнительный анализ алгоритмов классификации, в основном руководствуются лишь одной метрикой оценки качества, что на несбалансированных данных может быть не так показательно. Лишь в работе [6] приведен анализ *ROC-кривых*.

В работах [6, 9] оценивается ансамблевый алгоритм «*случайный лес*», но при этом, в этих работах не рассматриваются другие ансамблевые методы классификации.

В работе [5] показано, что лучшие параметры качества алгоритмов достигаются при отборе всех признаков для обучения моделей.

В отличие от представленных работ, в настоящей работе проведен более комплексный анализ алгоритмов классификации.

1. Описание набора данных

Исследование проводилось на наборе данных NSL-KDD (2014), являющемся модернизированной частью KDD n/'99 [10]. Каждая запись в базе KDD представляет собой образ сетевого соединения и включает **41** *информационный признак – индивидуальное измеримое свойство или характеристику наблюдаемого явления* [11, 12] – и промаркирована как «атака» или «не атака».

В исследуемом наборе атаки делятся на четыре основные категории:

DoS (*от англ.* Denial of Service) – отказ в обслуживании; характерна генерация большого объема трафика, что приводит к перегрузке и блокированию сервера;

U2R (*от англ.* User to Root) – получение зарегистрированным пользователем привилегий локального суперпользователя (администратора);

R2L (*от англ.* Remote to Local) – получение доступа незарегистрированного пользователя к компьютеру со стороны удаленной машины;

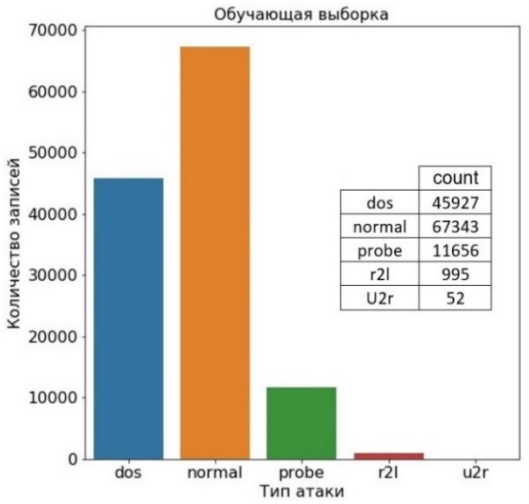
Probe – сканирование системы на наличие уязвимостей с целью их дальнейшей эксплуатации для получения доступа к системе.

В наборе данных каждая запись промаркирована, и, если эта запись соответствует вредоносному трафику, ей присваивается определенный тип атаки. Всего представлено 22 основных типа атак, 18 дополнительных типов (не указанных в документации), нормальный и неизвестный трафик. Всего **42** различные **метки** каждой записи. Каждая запись включает в себя поля, информационные признаки, описанные в таблице 1 [10]. Если обозначить каждую метку записи как класс, то каждый признак можно назвать *атрибутом* этого класса.

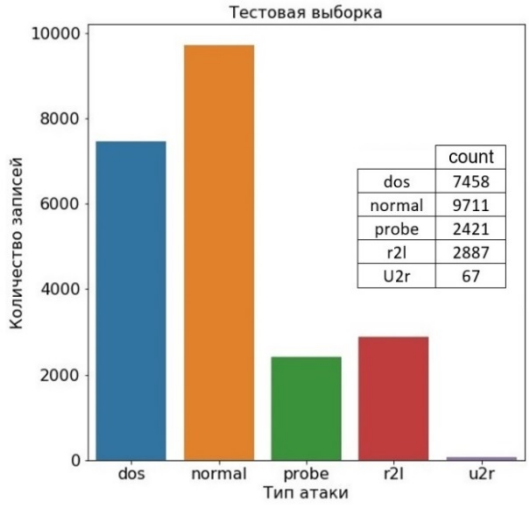
ТАБЛИЦА 1. Описание полей набора данных NSL-KDD

№	Поле	Описание
Данные TCP-соединения		
1.	duration	Продолжительность сессии, с
2.	protocol_type	Тип протокола (TCP, UDP и т. д.)
3.	service	Удаленный сервис (http, telnet и т. д.)
4.	flag	Статус соединения (normal или error)
5.	src_bytes	Количество исходящих байт (источник -> назначение)
6.	dst_bytes	Количество входящих байт (назначение -> источник)
7.	land	1, если подключен с того же хоста/порта, по умолчанию – 0
8.	wrong_fragment	Количество «неправильных» пакетов
9.	urgent	Количество срочных пакетов
Данные домена		
10.	hot	Количество «hot» индикаторов
11.	num_failed_logins	Количество неудачных авторизаций
12.	logged_in	1 при успешной авторизации, 0 – по умолчанию
13.	num_compromised	Количество «скомпрометированных» условий
14.	root_shell	1, если вход выполнен под root, 0 – по умолчанию
15.	su_attempted	1, если была попытка входа под root, 0 – по умолчанию
16.	num_root	Количество доступов суперпользователя
17.	num_file_creations	Количество операций по созданию файла
18.	num_shells	Количество сессий терминала
19.	num_access_files	Количество операций по доступу к файлам
20.	num_outbound_cmds	Количество исходящих команд в ftp-сессии
21.	is_host_login	1, если логин в списке «hosts», 0 – по умолчанию
22.	is_guest_login	1, если логин гостевой, 0 – по умолчанию
Данные, посчитанные в 2-х секундном окне		
23.	count	Количество подключений на один хост в рамках текущей сессии за последние 2 секунды
24.	srv_count	Количество подключений к одному сервису в рамках текущей сессии за последние 2 секунды

№	Поле	Описание
25.	serror_rate	% от подключений с «SYN» ошибкой
26.	srv_serror_rate	% от подключений с «SYN» ошибкой при подключении на один сервис
27.	rerror_rate	% от подключений с «REJ» ошибкой
28.	srv_rerror_rate	% от подключений с «REJ» ошибкой при подключении на один сервис
29.	same_srv_rate	% от подключения к одному и тому же сервису
30.	diff_srv_rate	% от подключения к разным сервисам
31.	srv_diff_host_rate	% от подключения к разным хостам
Данные, посчитанные в 100-секундном окне		
32.	dst_host_count	Количество подключений на один хост в рамках текущей сессии за последние 100 секунд
33.	dst_host_srv_count	Количество подключений на один сервис в рамках текущей сессии за последние 100 секунд
34.	dst_host_same_srv_rate	% от подключения к одному и тому же сервису
35.	dst_host_diff_srv_rate	% от подключения к разным сервисам
36.	dst_host_same_src_port_rate	% от подключения с одного и того же порта источника
37.	dst_host_srv_diff_host_rate	% от подключения к одному и тому же хосту
38.	dst_host_serror_rate	% от подключений с «SYN» ошибкой
39.	dst_host_srv_serror_rate	% от подключений с «SYN» ошибкой при подключении на один сервис
40.	dst_host_rerror_rate	% от подключений с «REJ» ошибкой
41.	dst_host_srv_rerror_rate	% от подключений с «REJ» ошибкой при подключении на один сервис



а)



б)

Рис. 1. Распределение классов в обучающей (а) и тестовой (б) выборках

2. Подготовка данных

Поскольку многие алгоритмы машинного обучения работают только с числовыми признаками, поля с ***символьным*** типом, а именно: *protocol type, service, flag* – были преобразованы к ***числовому*** типу (каждому уникальному значению каждого поля было сопоставлено целое число).

Если каждую возможную метку записи представить как класс, то получится **42** возможных класса, что даст слишком большую вариацию, поэтому *40 меток* атак были *преобразованы* к их *категориям*, *нормальный* трафик был оставлен без изменений, а *неизвестный* был исключен из-за отсутствия записей с данной меткой в наборе. Таким образом в результате преобразования все поля были промаркированы *пятью* классами.

В NSL-KDD данные разбиты на обучающую (125 973 записи) и тестовую (22 544 записи) выборки, и классы в них распределены так, как это показано на рисунке 1.

Как видно из рисунка 1, классы в наборе данных NSL-KDD не сбалансированы, т. е. количество записей с классом «normal» и «dos» в несколько раз превосходит количество записей по другим классам. Это может в дальнейшем с большой вероятностью привести к «недообучению» алгоритмов и, следовательно, к ошибочной классификации редких категорий атак.

Некоторые алгоритмы машинного обучения работают со всеми признаками как с одним вектором, а поскольку значения признаков неоднородны, это может привести к некорректности вычислений [13–15]. Для устранения этого недостатка использована *нормализация* набора данных в интервале от 0 до 1 по принципу мини-макс в соответствии с формулой:

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}, \quad (1)$$

где $\min(X)$ и $\max(X)$ – минимальное и максимальное значение поля из всего набора данных.

Алгоритмы классификации

Для классификации рассматриваемого набора данных NSL-KDD были использованы следующие алгоритмы машинного обучения.

1) «Классические» алгоритмы машинного обучения, для которых использовался *нормализованный* набор данных:

– ***Метод k-ближайших соседей*** (*k*-Nearest Neighbors, neighbors); параметром алгоритма было выбрано число соседей $k = 6$;

– ***Множественная логистическая регрессия*** (Logistic Regression, regression); для решения уравнения логистической регрессии использовался алгоритм SAGA [14–16];

– ***Мультиномиальный «наивный» Байес*** (Multinomial Naive Bayes, nb);

– ***Метод опорных векторов*** (Support Vector Machines, SVM, svc); для исследуемого набора данных применялось ядро *радиальной базисной функции* ($k(x, x') = e^{(-\gamma\|x-x\prime\|^2)}$) с параметром $\gamma = 2$.

2) Ансамблевые методы, для которых нормализация не требуется из-за того, что основой алгоритмов является дерево решений:

– ***Дерево решений*** (Decision Tree Classifier, dtc); в качестве оценочной функции использовался коэффициент неопределенности Gini; в ходе эмпирического анализа было выяснено, что наилучший результат алгоритма достигается при *количестве признаков* – 28 и *глубине дерева* – 23*;*

– «***Случайный лес»*** (Random Forest); наилучший результат для рассматриваемого набора данных был получен при разбивке данных на 100 подвыборок;

– ***Ada Boost***; наилучший результат для рассматриваемого набора данных был получен при разбивке данных на 1000 подвыборок.

Метрики оценки алгоритмов классификации

В задачах машинного обучения наиболее часто используются следующие метрики для оценки эффективности построенных моделей [11]: точность (*precision*), полнота (*recall*), *F*-мера (*F-score*), ROC-кривые (*от англ.* Receiver Operating Characteristic curve – кривая ошибок), AUC-ROC и AUC-PR (*от англ*. Area Under Curve – площадь под кривой ошибок и площадь под кривой *precision-recall*) [12, 14].

После проведения классификации возможно получение четырех видов результатов: TP (*от англ.* True Positive – истинно положительный), TN (*от англ.* True Negative – истинно отрицательный), FP (*от англ.* False Positive – ложно положительный) и FN (*от англ.* False Negative – ложно отрицательный). Эти результаты можно представить в виде матрицы ошибок в таблице 2, где y'– ответ алгоритма на объекте, а y – истинная метка класса на этом объекте.

ТАБЛИЦА 2. Матрица ошибок

	$y = 1$	$y = 0$
$y' = 1$	True Positive (TP)	False Positive (FP)
$y' = 0$	False Negative (FN)	True Negative (TN)

Точность (precision) показывает долю объектов, названных классификатором положительными и при этом действительно являющимися положительными:

$$precision = \frac{TP}{TP + FP}. \quad (2)$$

Полнота (recall) показывает долю правильно помеченных положительных объектов среди всех объектов положительного класса:

$$recall = \frac{TP}{TP + FN}. \quad (3)$$

Точность чувствительна к распределению данных, в то время как полнота – нет. Полнота не отражает, сколько объектов помечены как положительные неверно, а точность не дает никакой информации о том, сколько положительных объектов помечены неправильно [11].

F-мера (F-score, F_β) сочетает в себе вышеупомянутые две метрики – среднее гармоническое точности и полноты:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}, \quad (4)$$

где β – принимает значения в диапазоне $0 < \beta < 1$, если приоритет отдается точности, и $\beta > 1$, если приоритет отдается полноте.

F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

ROC-кривая или **кривая ошибок** – график, позволяющий оценить качество классификации, который отображает соотношение между *чувствительностью (TPR, True Positive Rate)* алгоритма и долей из объектов отрицательного класса, которые алгоритм предсказал неверно *(FPR, False Positive Rate)* при варьировании порога решающего правила:

$$FPR = \frac{FP}{FP + TN}. \quad (5)$$

Количественную интерпретацию ROC-кривой дает **показатель AUC-ROC**. В идеальном случае, когда классификатор не делает ошибок (*FPR* = 0, *TPR* = 1), получается площадь под кривой, равная 1; если классификатор «гадает», то AUC-ROC будет стремиться к 0,5, так как классификатор будет выдавать одинаковое количество TP и FP. Площадь под кривой в этом случае показывает качество алгоритма (больше – лучше); кроме этого, важной является крутизна самой кривой (желательно максимизировать *TPR*, минимизируя *FPR*), а значит, она в идеале должна стремиться к точке (0,1) [14].

Критерий AUC-ROC устойчив к несбалансированным классам и может быть интерпретирован как вероятность того, что случайно выбранный положительный объект будет ранжирован классификатором выше (будет иметь более высокую вероятность быть положительным), чем случайно выбранный отрицательный объект [14].

Помимо ROC-кривой существует PR-кривая (*от англ.* Precision-Recall Curve), показывающая отношение *точности* (2) от *полноты* (3). Соответственно, количественный показатель в данном случае такой же – площадь по кривой – **AUC-PR** (больше – лучше). В идеальном случае кривая должна стремиться к точке (1,1), где классификатор получает только истинно положительные результаты, без отрицательных. PR-анализ также применяется на несбалансированных данных.

3. Оценка алгоритмов классификации

На рисунке 2 показаны матрицы ошибок для каждого из рассматриваемых алгоритмов машинного обучения. Они показывают количество меток, присвоенных алгоритмом (Predicted label), и их соответствие истинным меткам (True label). Значения в матрице ошибок нормализованы относительно количества записей каждого класса.

Из них следует, что большинством достаточно хорошо классифицируются классы *dos*, *normal* и *probe*, а классы *r2l* и *u2r* чаще всего классифицируются как *normal*. Последнее является следствием несбалансированности (недостаточности) обучающих данных. Лучшие характеристики для верного оп-ределения этих двух классов показал алгоритм *Ada Boost* (рисунок 2ж), где при обучении идет работа над исправлением неверно классифицированных данных. По главной диагонали отображается значение метрики *полнота*.

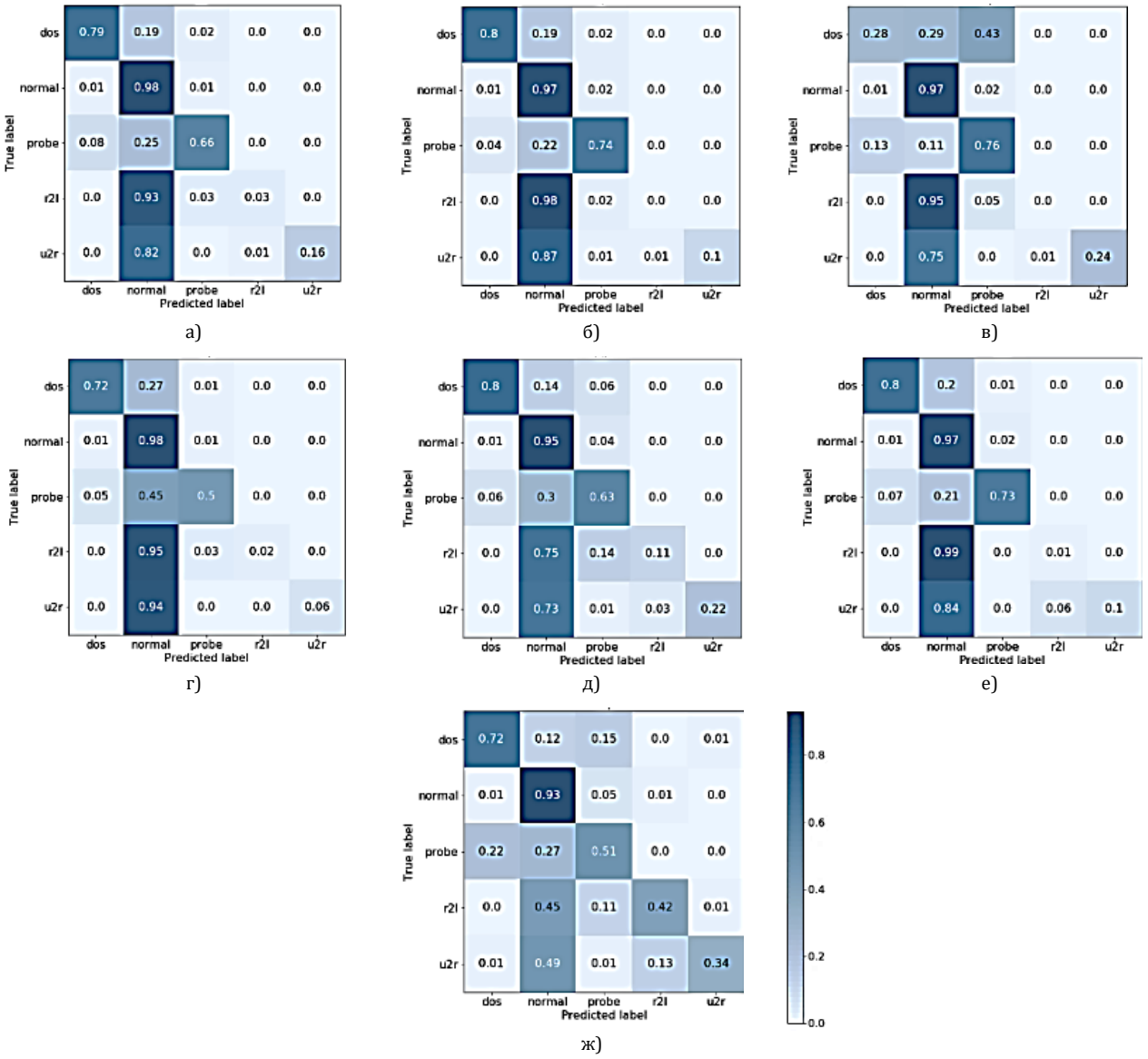


Рис. 2. Матрицы ошибок для алгоритмов: а) метод *k*-ближайших соседей; б) множественная логистическая регрессия; в) мультиномиальный «наивный» Байес; г) метод опорных векторов; д) дерево решений; е) «случайный лес»; ж) Ada Boost

Из анализа таблиц 3–6 можно сделать вывод о том, что хорошо определяются классы *dos*, *normal* и *probe*, а классы *r2l* и *u2r* – плохо. Основываясь на метрике *F*-score, можно сделать вывод о том, что алгоритмы *логистическая регрессия* и «*случайный лес*» хорошо определяют классы *dos, normal* и *probe,* а алгоритм *Ada Boost* может сбалансировать результат при несбалансированных данных.

ТАБЛИЦА 3. Значение метрики *precision*

Алгоритмы\Классы	dos	normal	probe	r2l	u2r
k-ближайших соседей	0,96	0,66	0,81	0,93	0,79
Логистическая регрессия	**0,97**	0,66	0,83	0,50	**0,88**
«Наивный» Байес	0,85	0,64	0,34	0,44	0,67
Метод опорных векторов	**0,97**	0,62	0,80	0,95	0,80
Дерево решений	0,96	0,70	0,56	**0,97**	0,58
«Случайный лес»	0,96	0,67	**0,86**	0,96	0,75
Ada Boost	0,89	**0,76**	0,40	0,92	0,17

ТАБЛИЦА 4. Значение метрики *recall*

Алгоритмы\Классы	dos	normal	probe	r2l	u2r
k-ближайших соседей	0,79	**0,98**	0,66	0,03	0,16
Логистическая регрессия	0,80	0,97	0,74	0	0,1
«Наивный» Байес	0,28	0,97	**0,76**	0	0,24
Метод опорных векторов	0,72	**0,98**	0,5	0,02	0,06
Дерево решений	0,80	0,95	0,63	0,11	0,22
«Случайный лес»	**0,82**	0,97	0,72	0,03	0,09
Ada Boost	0,71	0,93	0,54	**0,43**	**0,34**

ТАБЛИЦА 5. Значение метрики *F-score*

Алгоритмы\Классы	dos	normal	probe	r2l	u2r
k-ближайших соседей	0,87	0,79	0,73	0,07	0,27
Логистическая регрессия	**0,88**	0,79	**0,78**	0,01	0,19
«Наивный» Байес	0,42	0,77	0,47	0	**0,35**
Метод опорных векторов	0,83	0,76	0,61	0,04	0,11
Дерево решений	0,87	0,81	0,59	0,2	0,32
«Случайный лес»	**0,88**	0,79	**0,78**	0,08	0,16
Ada Boost	0,71	**0,84**	0,46	**0,46**	0,23

ТАБЛИЦА 6. Среднее взвешенное значение метрик

Алгоритмы\Метрики	precision	recall	*F*-score
k-ближайших соседей	**0,81**	0,76	0,71
Логистическая регрессия	0,76	0,76	0,71
«Наивный» Байес	0,65	0,59	0,52
Метод опорных векторов	0,8	0,72	0,67
Дерево решений	0,8	0,76	0,73
«Случайный лес»	**0,81**	**0,77**	0,72
AdaBoost	0,78	0,75	**0,75**

После расчета метрик AUC-ROC и AUC-PR для сравнительного анализа кривых было оставлено четыре алгоритма, которые показали лучшие значения для этих метрик: *k-ближайших соседей*, *логистическая регрессия*, *метод опорных векторов* и *«случайный лес».*

Из зависимостей метрики AUC-ROC, представленных на рисунке 3 и в таблице 7, следует, что лучше всего себя показали алгоритмы «случайный лес», метод опорных векторов и логистическая регрессия. При этом «*случайный лес*» дал схожие с *методом опорных векторов* параметры. Тем не менее метрика AUC-ROC показывает, что эти алгоритмы хорошо определяют классы *r2l* и *u2r*, что, однако, не соответствует матрице ошибок на рисунке 2, а также значениям метрик *precision, recall и F-score* в таблицах 3–5, и обусловлено несбалансированностью классов.

ТАБЛИЦА7. Значения AUC-ROC для всех алгоритмов машинного обучения

Алгоритмы\Классы	dos	normal	probe	r2l	u2r
k-ближайших соседей	0,9	0,84	0,86	0,53	0,71
Логистическая регрессия	0,96	0,89	**0,97**	0,77	**0,96**
Метод опорных векторов	**0,97**	0,94	0,94	0,71	0,85
«Случайный лес»	**0,97**	**0,96**	0,96	0,72	0,89

Из зависимостей метрики PR-AUC, представленных на рисунке 4 и в таблице 8 следует, что лучше всего себя показал алгоритм «*случайный лес*» (см. рисунок 4г). При этом видно, что по этой метрике алгоритмы плохо справляются с определением классов *r2l* и *u2r,* что не исключается метриками *precision* и *recall*.

Из этого можно сделать вывод о том, что метрика AUC-PR лучше подходит для оценки несбалансированных данных, чем AUC-ROC.

ТАБЛИЦА 8. Значения PR-AUC для всех алгоритмов машинного обучения

Алгоритм\Классы	dos	normal	probe	r2l	u2r
k-ближайших соседей	0,85	0,705	0,628	0,175	0,242
Логистическая регрессия	0,931	0,83	**0,834**	0,363	0,38
Метод опорных векторов	**0,952**	0,9	0,781	0,313	0,209
«Случайный лес»	0,944	**0,94**	0,798	**0,475**	**0,461**

Выводы

Из анализа представленных данных можно сделать вывод, что из-за несбалансированности обучающих данных алгоритмы машинного обучения показали себя плохо на исследуемом наборе. Для решения этой проблемы требуется сбалансировать классы, например, путем дублирования или перейти к бинарной классификации – «нормальный» vs «не нормальный».

Однако если рассмотреть определение сбалансированных классов (*dos*, *normal* и *probe*), то лучше всего себя показали алгоритмы «*случайный лес*», *метод опорных векторов* и *логистическая регрессия*. При схожих выходных характеристиках, метод опорных векторов и регрессия требуют подготовки входных данных, в частности нормализации, тогда как случайный лес этого не предполагает.

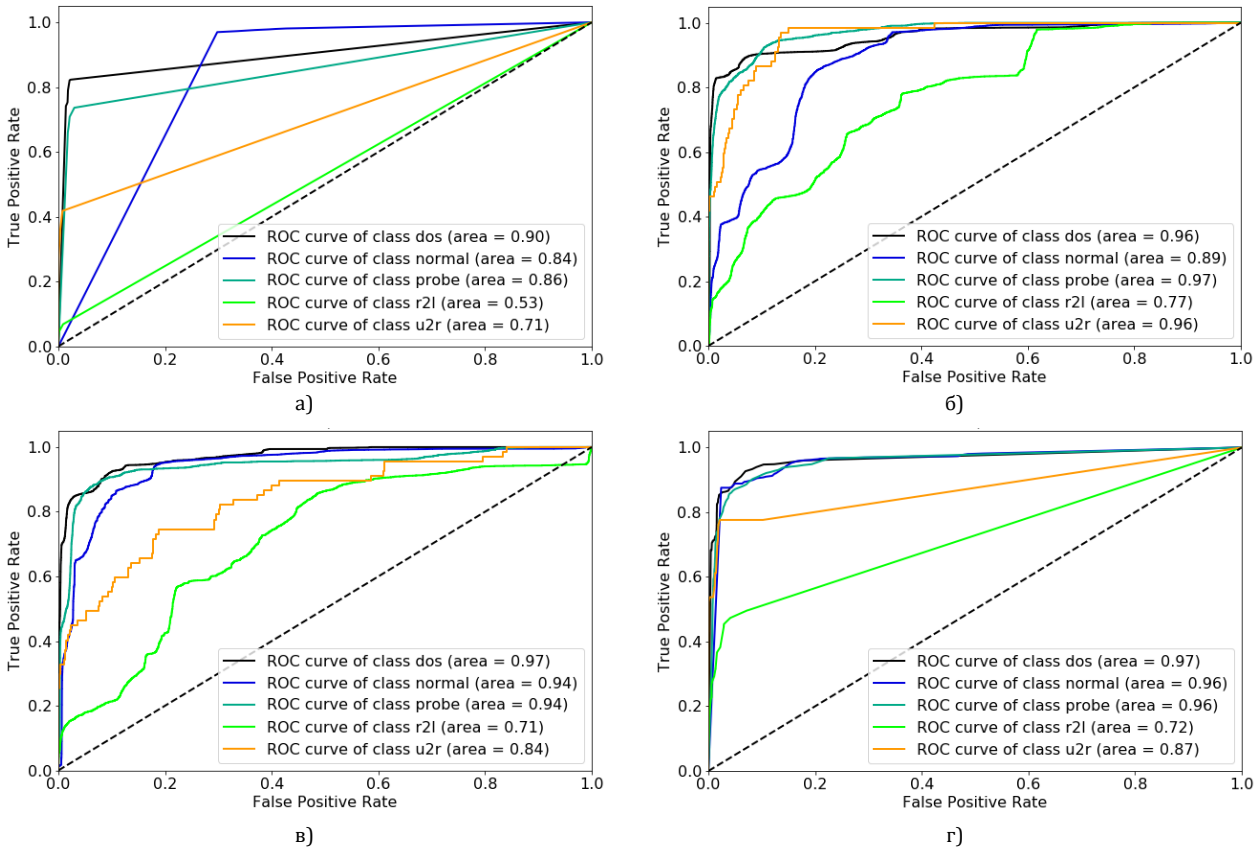


Рис. 3. ROC-кривые и ROC-AUC для алгоритмов: а) метод *k*-ближайших соседей; б) множественная логистическая регрессия; в) метод опорных векторов; г) «случайный лес»

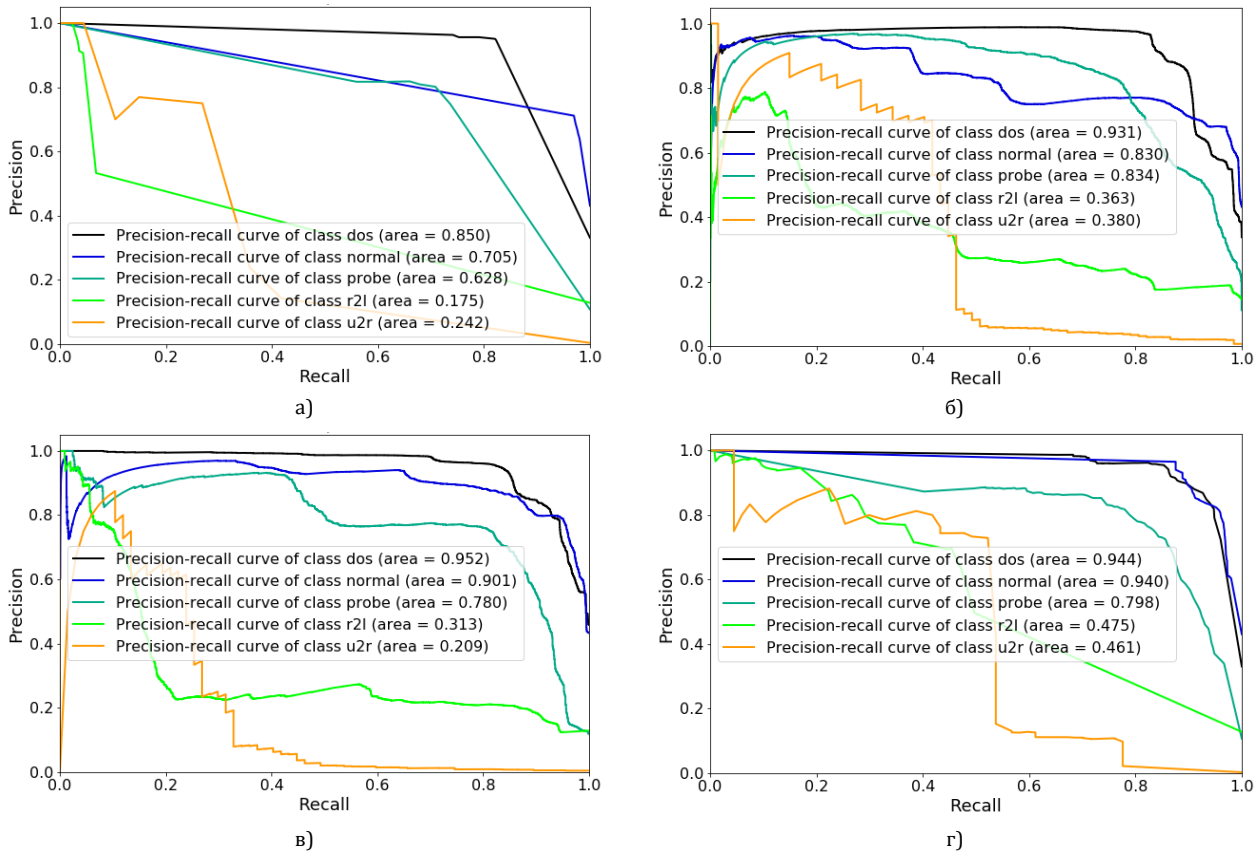


Рис. 4. PR-кривые и PR-AUC для алгоритмов: а) метод *k*-ближайших соседей; б) множественная логистическая регрессия; в) метод опорных векторов; г) «случайный лес»

Алгоритм «*случайный лес*» кратно превосходит по скорости работы метод опорных векторов, т. к. в первом в основе лежат вычислительно легкие деревья решений, и имеется возможность распараллеливать процессы подсчета, а во втором – вычислительно сложный алгоритм бинарной классификации, который сравнивает каждый класс с каждым, что сильно увеличивает время работы.

Таким образом, для многоклассовой классификации вредоносного трафика из исследованных алгоритмов машинного обучения лучше всего подходит алгоритм «*случайный лес*».

Список используемых источников

1. Шелухин О.И. Сетевые аномалии. Обнаружение, локализация, прогнозирование. М.: Горячая линия–Телеком, 2019. 448 с.

2. Шелухин О.И., Сакалема Д.Ж., Филинова А.С. Обнаружение вторжений в компьютерные сети (сетевые аномалии). М: Горячая линия–Телеком, 2016. 220 с.

3. Thomas R., Pavithran D. A Survey of Intrusion Detection Models based on NSL-KDD Data Set // Proceedings of the 5th HCT Information Technology Trends (ITT, Dubai, United Arab Emirates, 28–29 November 2018). Piscataway, NJ: IEEE, 2018. PP. 286–291. DOI:10.1109/CTIT.2018.8649498

4. Dhanabal L., Shantharajah S.P. A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms // International Journal of Advanced Research in Computer and Communication Engineering. 2015. Vol. 4. Iss. 6. PP. 446–452. DOI:10.17148/IJARCCE.2015.4696

5. Pervez M.S., Farid D.M. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs // Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA, Dhaka, Bangladesh, 18–20 December 2014). Piscataway, NJ: IEEE, 2014. DOI:10.1109/SKIMA.2014.7083539

6. Revathi S., Malathi A. A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection // International Journal of Engineering Research & Technology. 2013. Vol. 2. Iss. 12. PP. 1848–1853.

7. Paulauskas N., Auskalnis J. Analysis of data pre-processing influence on intrusion detection using NSL-KDD dataset // Proceedings of the Open Conference of Electrical, Electronic and Information Sciences (eStream, Vilnius, Lithuania, 27 April 2017). Piscataway, NJ: IEEE, 2017. DOI:10.1109/eStream.017.7950325

8. Meena G., Choudhary R.R. A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA // Proceedings of the International Conference on Computer, Communications and Electronics (Comptelix, Jaipur, India, 1–2 July 2017). Piscataway, NJ: IEEE, 2017. PP. 553–558. DOI:10.1109/COMPTELIX.2017.8004032

9. Ingre B., Yadav A., Soni A.K. Decision Tree Based Intrusion Detection System for NSL-KDD Dataset // Proceedings of the International Conference on Information and Communication Technology for Intelligent Systems (ICTIS, Ahmedabad, India, 25–26 March 2017). Cham: Springer, 2017. Vol. 2. PP. 207–218. DOI:10.1007/978-3-319-63645-0_23

10. Protic D.D. Review of KDD CUP '99, NSL-KDD and KYOTO 2006+ datasets // Vojnotehnički Glasnik. 2018. Vol. 66. Iss. 3. PP. 580–596. DOI:10.5937/vojtehg66-16670

11. Bishop C.M. Pattern Recognition and Machine Learning. Berlin: Springer, 2006.

12. Шелухин О.И., Симонян А.Г., Ванюшина А.В. Влияние структуры обучающей выборки на эффективность классификации приложений трафика методами машинного обучения // T-Comm: Телекоммуникации и транспорт. 2017. Т. 11. № 2. С. 25–31.

13. Knowledge Discovery in Databases – обнаружение знаний в базах данных // BaseGroup Labs. Технологии анализа данных. URL: https://basegroup.ru/community/articles/kdd (дата обращения 04.03.2019)

14. Шелухин О.И., Ерохин С.Д., Ванюшина А.В. Классификация IP-трафика методами машинного обучения. М.: Горячая линия–Телеком, 2018. 284 с.

15. Mitchell T. Machine Learning. NY: McGraw-Hill, 1997. 414 p.

16. Defazio A., Bach F., Lacoste-Julien S. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives // Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS, Montreal, Canada, 08–13 December 2014). Cambridge: MIT Press, 2014. Vol. 1. PP. 1646–1654.

* * *

MULTICLASS CLASSIFICATION OF ATTACKS TO INFORMATION RESOURCES WITH MACHINE LEARNING TECHNIQUES

M. Kazhemskiy[1], **O. Sheluhin**[1]

[1]Moscow Technical University of Communication and Informatics,
Moscow, 111024, Russian Federation

Article info
Article in Russian

For citation: Kazhemskiy M., Sheluhin O. Multiclass Classification of Attacks to Information Resources with Machine Learning Techniques. *Proceedings of Telecommunication Universities*. 2019;5(1):107–115. (in Russ.) Available from: https://doi.org/10.31854/1813-324X-2019-5-1-107-115

Abstract: *The article considers the classification of attacks on information resources using "classic" machine learning algorithms: k-Nearest Neighbors, Logistic Regression, Naive Bayes, Support Vectors, also ensemble methods: Decision Tree, Random Forest and Ada Boost. The research was conducted on the NSL-KDD data set using Python programming language libraries: scikit-learn, pandas and jupyter notebook. Data in the dataset were prepared for the research along with optimization of machine learning algorithm parameters. All fields in the dataset were marked with five classes, which correspond to four categories of attacks (DoS, U2R, R2L, Probe) and normal traffic (normal). A comparative analysis of the classification of each algorithm were made using different evaluation metrics. It was concluded that all the reasearched algorithms have shown insufficient efficiency in the conditions of data imbalance. It was proposed to perform additional actions on the initial dataset for better classification. The best results were demonstrated by the Random Forest algorithm.*

Keywords: *multiclass classification, machine learning, attacks, classic algorithms, ensemble methods, scikit-learn, NSL-KDD.*

References

1. Sheluhin O.I. *Setevye anomalii. Obnaruzhenie, lokalizatsiia, prognozirovanie* [Network Anomalies. Detection, localization, prediction]. Moscow: Goriachaia liniia-Telekom Publ.; 2019. 448 p. (in Russ.)
2. Sheluhin O.I., Sakalema D.Z., Filinova A.S. *Obnaruzhenie vtorzhenii v kompiuternye seti (setevye anomalii)*. [Intrusion Detection in Computer Networks: Network Anomalies]. Moscow: Goriachaia liniia-Telekom Publ.; 2016. 220 p. (in Russ.)
3. Thomas R., Pavithran D. A Survey of Intrusion Detection Models based on NSL-KDD Data Set. *Proceedings of the 5th HCT Information Technology Trends, ITT, 28–29 November 2018, Dubai, United Arab Emirates*. Piscataway, NJ: IEEE; 2018. p.286–291. Available from: https://doi.org/10.1109/CTIT.2018.8649498
4. Dhanabal L., Shantharajah S.P. A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*. 2015;4(6):446–452. Available from: https://doi.org/10.17148/IJARCCE.2015.4696
5. Pervez M.S., Farid D.M. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. *Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications, SKIMA, 18–20 December 2014, Dhaka, Bangladesh*. Piscataway, NJ: IEEE; 2014. Available from: https://doi.org/10.1109/SKIMA.2014.7083539
6. Revathi S., Malathi A. A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection. *International Journal of Engineering Research & Technology*. 2013;2(12):1848–1853.
7. Paulauskas N., Auskalnis J. Analysis of data pre-processing influence on intrusion detection using NSL-KDD dataset. *Proceedings of the Open Conference of Electrical, Electronic and Information Sciences, eStream, 27 April 2017, Vilnius, Lithuania*. Piscataway, NJ: IEEE; 2017. Available from: https://doi.org/10.1109/eStream.017.7950325
8. Meena G., Choudhary R.R. A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. *Proceedings of the International Conference on Computer, Communications and Electronics, Comptelix, 1–2 July 2017, Jaipur, India*. Piscataway, NJ: IEEE; 2017. p.553–558. Available from: https://doi.org/10.1109/COMPTELIX.2017.8004032
9. Ingre B., Yadav A., Soni A.K. Decision Tree Based Intrusion Detection System for NSL-KDD Dataset. Proceedings of the International Conference on Information and Communication Technology for Intelligent Systems, ICTIS, 25–26 March 2017, Ahmedabad, India. Cham: Springer; 2017. Vol. 2. p.207–218. Available from: https://doi.org/10.1007/978-3-319-63645-0_23
10. Protic D.D. Review of KDD CUP '99, NSL-KDD and KYOTO 2006+ datasets. *Vojnotehnički Glasnik*. 2018;66(3):580–596. Available from: https://doi.org/10.5937/vojtehg66-16670
11. Bishop C.M. *Pattern Recognition and Machine Learning*. Berlin: Springer; 2006.
12. Sheluhin O.I., Simonyan A.G., Vanyushina A.V. Influence of training sample structure on traffic application efficiency classification using machine-learning methods. *T-Comm.* 2017;11(2):25–31. (in Russ.)
13. BaseGroupLabs. Tekhnologii analiza dannykh [BaseGroupLabs. Data Analysis Technologies]. *Knowledge Discovery in Databases.* (in Russ.) Available from: https://basegroup.ru/community/articles/kdd [Accessed 4th March 2019]
14. Sheluhin O.I., Erokhin S.D., Vaniushina A.V. *Klassifikatsiia IP-trafika metodami mashinnogo obucheniia* [IP-Traffic Classification by Machine Learning Methods]. Moscow: Goriachaia liniia-Telekom Publ.; 2018. 284 p. (in Russ.)
15. Mitchell T. *Machine Learning*. NY: McGraw-Hill; 1997. 414 p.
16. Defazio A., Bach F., Lacoste-Julien S. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. *Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS, 08–13 December 2014, Montreal, Canada*. Cambridge: MIT Press; 2014. Vol. 1. p.1646–1654.