

АНАЛИЗ ВЛИЯНИЯ ПАРАМЕТРОВ АЛГОРИТМОВ MACHINE LEARNING НА РЕЗУЛЬТАТЫ КЛАССИФИКАЦИИ ТРАФИКА В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

DOI: 10.36724/2072-8735-2021-15-9-24-35

Manuscript received 07 April 2021;
Accepted 26 May 2021

Краснова Ирина Артуровна,
МТУСИ, Москва, Россия,
irina_krasnova-angel@mail.ru

Ключевые слова: машинное обучение, Random Forest, XGBoost, QoS, классификация трафика, настройка параметров, переобучение

В статье анализируется влияние настройки параметров алгоритмов машинного обучения на результаты классификации трафика в режиме реального времени. Рассматриваются алгоритм Random Forest и XGBoost. Даётся краткое описание работы обоих методов и способов оценки результатов классификации. Экспериментальные исследования проводятся на базе данных, полученных на реальной сети, отдельно для TCP и UDP - потоков. Для того чтобы результаты исследования использовались в режиме реального времени, создается специальная матрица признаков, основанная на первых 15 пакетах потока. В качестве основных параметров алгоритма Random Forest (RF) для настройки выбираны количество деревьев, используемый критерий разбиения, максимальное число признаков для построения функции разделения, глубина дерева, минимальное число выборок в узле и в листе. Для XGBoost взяты количество деревьев, глубина дерева, минимальное число выборок в листе, доля признаков и доля выборки, необходимые для построения дерева. Как известно, увеличение числа деревьев приводит к увеличению точности до определенного значения, но как показано в статье, важно следить за тем, чтобы модель не оказалась переобученной. Для борьбы с переобучением используются остальные параметры деревьев. В исследуемом наборе данных за счет устранения переобучения удалось добиться повышения точности классификации для отдельных приложений на 11-12% для Random Forest и на 12-19% для XGBoost. Результаты работы показывают, что настройка параметров является очень важным этапом при построении модели классификации трафика, т.к. помогает бороться с переобучением и значительно повышает точность предсказаний алгоритма. Помимо этого было показано, что при правильной настройке параметров, мало популярный в работах по классификации трафика XGBoost, становится конкурентоспособным алгоритмом и показывает лучшие результаты по сравнению с широко распространенным Random Forest.

Информация об авторе:

Краснова Ирина Артуровна, аспирант, МТУСИ, Москва, Россия

Для цитирования:

Краснова И.А. Анализ влияния параметров алгоритмов Machine Learning на результаты классификации трафика в режиме реального времени // T-Comm: Телекоммуникации и транспорт. 2021. Том 15. №9. С. 24-35.

For citation:

Krasnova I.A. (2021) Analysis of the influence of Machine Learning algorithm parameters on the results of traffic classification in real time. T-Comm, vol. 15, no.9, pp. 24-35. (in Russian)

Введение

В последнее время в различных отраслях науки все большее развитие получают методы интеллектуального анализа данных, в особенности методы машинного обучения (Machine Learning, ML), позволяющие на основе некоторых характеристик об объекте, делать какие-либо прогнозирующие выводы о его природе или свойствах. Отрасль телекоммуникаций не является исключением и также активно применяет ML для решения своих задач. Одной из таких задач является классификация трафика в режиме реального времени с целью определения QoS, которая позволила бы идентифицировать потоки, которые не могут быть выделены на этапе дизайна сети или с применением классификации на основе портов. Более подробно эта задача и общий алгоритм ее решения описаны в [1-2].

Одним из наиболее популярных алгоритмов, используемых для классификации трафика, является Random Forest, позволяющий строить некоторое количество решающих деревьев (Decision Tree) параллельно друг другу. В ряде работы [3-5] он показывал лучшие результаты по сравнению с другими анализируемыми методами. В то же время, результаты бустинга решающих деревьев, т.е. последовательного построения деревьев, показывали не такие успешные результаты. Например, в работе [6] значение F-меры AdaBoost находится в пределах 21-66%, в то время как F-мера Random Forest оказалась выше 90%.

Многие классификаторы имеют настраиваемые параметры, например, для RF – это количество деревьев, глубина дерева и т.д. В большинстве работ, в т.ч. и в выше перечисленных, не приводится подробный анализ влияния настройки параметров классификаторов.

В работе [7] проводится исследование по влиянию настройки параметров на результаты работы алгоритмов. Было выбрано 38 различных наборов данных и 6 методов ML, в т.ч. XGBoost. Показано, что при правильном подборе параметров, результаты классификации обычно улучшаются. Но в упомянутой статье не ставились задачи классификации трафика. В работе [8] анализ влияния настройки параметров SVM (Support Vector Machine) провели применительно к классификации трафика с целью обнаружения вторжения. Для такой же цели в [9] был проанализирован алгоритм XGBoost, которому удалось стать самым быстрым и самым эффективным классификатором из анализируемых, добившись точности 99,54%. Подробный всесторонний обзор работ по этой теме приведен в [10].

В этой статье проводится анализ влияния настройки параметров RF и XGBoost применительно к классификации трафика в режиме реального времени с целью определения QoS.

1. Алгоритм решающего дерева (Decision Tree, CART)

Методы машинного обучения с учителем предполагают наличие определенной базы данных с известными заранее классами. Каждый элемент этой выборки представлен своим вектором признаков и однозначно определен одним из имеющихся классов. База данных разбивается на обучающую и тестовую последовательности, на которых алгоритм обучается и тестируется соответственно.

Одним из наиболее распространенных методов ML является метод решающих деревьев (Decision Trees) [11]. Он позволяет представлять пространство данных в виде разветвленного ациклического графа, элементы которого называют листьями и узлами. В узлах содержится условие разветвления, а в листьях отмечается множество данных, объединенных между собой соблюдением либо несоблюдением выполнения условия, представленного в узле. Лист является концевой вершиной и не имеет дальнейших ответвлений (рис. 1).

Существует три основных типа решающих деревьев: ID3 (Iterative Dichotomiser 3), C4.5/ C5.0 и CART (Classification and Regression Trees). По сравнению с остальными, CART способен решать более широкий спектр задач и принимает разнообразные переменные, поэтому на практике все чаще применяют именно его [12]. При этом строятся бинарные деревья, имеющие на каждом узле по два ответвления.

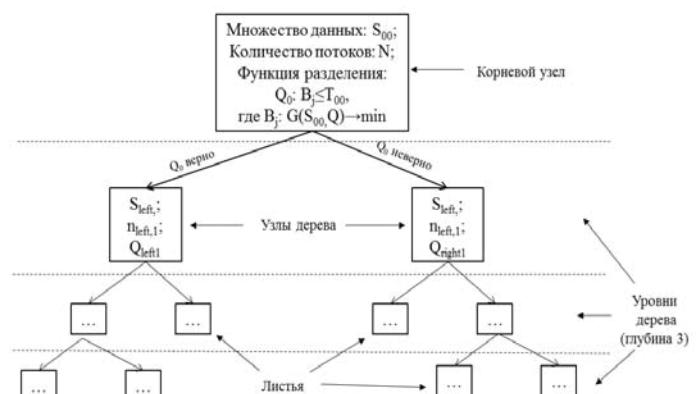


Рис. 1. Дерево решений, построенное по алгоритму CART и его элементы

Алгоритм построения дерева:

1. Пространства обучающей части вектора признаков B и соответствующего ему вектора класса Z представляются в виде множества $S_{side,i} = (B, Z)$, где i обозначает номер уровня дерева и $i = 0, 1, 2, \dots, max_node$, $side$ обозначает сторону разбиения $side = \{left, right, 0\}$. Для начального узла $i=0$, $side=0$, т.к. разделений еще не было, и этот узел называется корневым.

2. Для построения дерева $S_{side,i}$ подается на i -й узел. Выбирается значение $B_j \in B$ в качестве признака и T_i в качестве порогового значения разделятельной функции.

3. Рассчитывается функция i -го узла $Q_i = (B_j, T_i)$, разделяющая множество B на два подмножества (1). Определяется левая сторона (left) и правая (right):

$$S_{left, (i+1)}(Q_i) = S_{side, i} \mid B_j \leq T_i, \quad (1)$$

$$S_{right, (i+1)}(Q_i) = S \setminus S_{left, (i+1)}(Q_i)$$

4. Примесь (impurity) позволяет количественно оценить качество полученной функции разбиения (Q).

Если обозначить за N_i общее количество потоков трафика в i -м узле, n_{left} – количество потоков трафика, передаваемых в левый $i+1$ узел, а n_{right} в правый, то для узла i находится примесь (2):

$$G(S_{side, i}, Q_i) = \frac{n_{left}}{N_i} H(S_{left, (i+1)}(Q_i)) + \frac{n_{right}}{N_i} H(S_{right, (i+1)}(Q_i)), \quad (2)$$

исходя из которой, выбирается значение j для B_j так, чтобы минимизировать примеси:

$$G(S_{side, i}, Q_i) \rightarrow \min$$

При $G(S_{side, i}, Q_i) = 0$ в листе остаются потоки, принадлежащие только одному классу.

В качестве функции примеси могут быть использованы различные критерии. Наиболее известными являются: индекс Джини или индекс энтропии.

Пусть частота наблюдений класса Z_k в узле i $p_{i,k}$ (3):

$$p_{i,k} = \frac{1}{n_i} \sum_{B_i} I(Z_i = Z_k) \quad (3)$$

По умолчанию, все веса классов считаются равнозначными.

Индекс Джини показывает, насколько часто поток в узле классифицируется неверно. Функция примеси H на основе индекса Джини определяется по формуле (4):

$$H(S_{side,(i+1)}(Q_i)) = \sum_k p_{i,k} (1 - p_{i,k}) \quad (4)$$

Энтропия показывает наиболее редкие классы в узле и оценивается через логарифм правдоподобия (5):

$$H(S_{side,(i+1)}(Q_i)) = - \sum_k p_{i,k} \log(p_{i,k}) \quad (5)$$

5. $i = i+1$ и пункты 2-4 повторяются для $side=\{\text{left}, \text{right}\}$ пока не выполнится критерий останова.

По умолчанию, деревья решений строятся до тех пор, пока примесь в каждом узле не станет равной 0, т.е. такой алгоритм не будет содержать никаких ошибок. Такой подход плохо работает на новых данных, так как он является переобученным. В этом случае, нужно использовать какие-то дополнительные критерии остановки, стрижки либо применять ансамбли деревьев. При этом метод стрижки практически не используют, т.к. он довольно сложный и деревья редко строят по отдельности.

Решающие деревья по отдельности являются слабыми, плохо сбалансированными алгоритмами и успешно спрашиваются только с относительно несложными задачами. Но они могут быть важными компонентами при работе в ансамблевых алгоритмах, поэтому рекомендуется настраивать лучшим образом параметры модели [12].

2. Случайный лес (Random Forest, RF)

Случайный лес (RF, Random Forest) [13] представляет собой баггинг деревьев Бреймана, т.е. метод, при котором строится множество различных деревьев на разных подпространствах вектора признаков (рис. 2). При баггинге деревья строятся параллельно друг другу, а результат классификации определяется голосованием.

Алгоритм построения случайного леса:

1. Обучающее множество S разбивается случайным образом на $\{S_{t1}, S_{t2}, S_{tn_estimators}\} \subseteq S$ подмножества, где $n_estimators$ означает количество деревьев, участвующих в ансамбле. При этом подмножества S_t могут между собой пересекаться, и в сумме не образуют полное множество S . Выборка, не вошедшая в подмножество S_t , называется ОOB (out-of-bag) и обозначается $S_{ti,OOB}$.

2. На основании каждого из подмножеств S_t независимо друг от друга строится по одному дереву. Построение отдельных деревьев проходит по тем же принципам, что и

построение по алгоритму CART, но имеет две специфические особенности. Во-первых, в качестве просмотренных признаков в каждом узле по умолчанию ставится значение: $\max_features = \sqrt{m}$. Во-вторых, по умолчанию деревья строятся полными, без стрижки.

3. Для каждого отдельного потока y_j каждое из деревьев дает свой прогноз Tr_i (S_{ti}), где $i=1, 2, \dots, n_estimators$. Итоговое решение по классификации определяется как усредненное значение между результатами отдельных деревьев (6):

$$S_y(y) = \frac{1}{n_estimators} \sum_{i=1}^{n_estimators} Tr_i(y) \quad (6)$$

Оценка качества алгоритма RF

Для оценки качества алгоритма RF помимо стандартных параметров, полученных с помощью матрицы ошибок, также применяются параметры, рассчитанные с помощью метода OOB (out-of-bag). Метод OOB заключается в построении случайного леса на основе потоков, не участвующих в обучении выбранных для тестирования деревьев и последующем подсчете количества ошибочек классификации (7) по OOB (ER_{OOB}):

$$\forall S_t \in S_{t,OOB}:$$

$$ER_{OOB} = \frac{1}{N_{OOB}} \sum_{i=1}^{N_{OOB}} I(S_y(Y) \neq f(S, Y)), \quad (7)$$

где N_{OOB} – размер множества $S_{t,OOB}$.

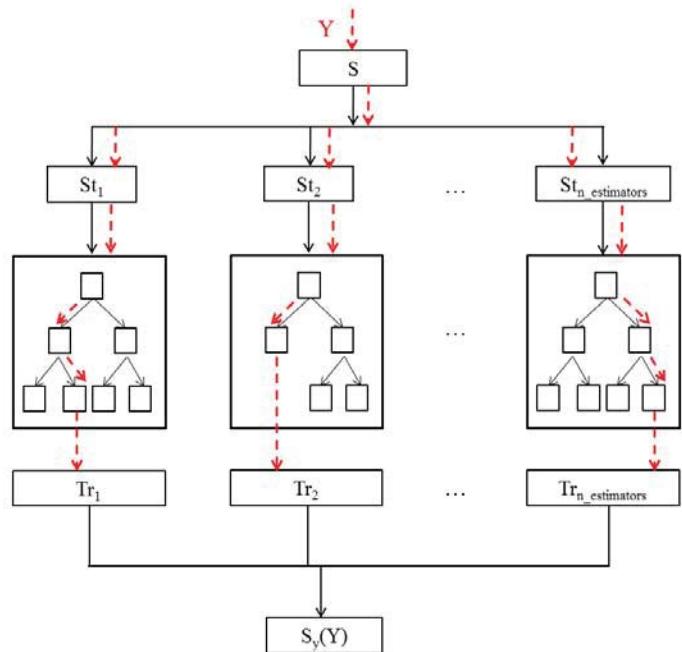


Рис. 2. Классификация сетевого трафика по алгоритму Random Forest

В работе для реализации алгоритма Random Forest используется библиотека Scikit-learn языка программирования Python.

В таблице 1 перечислены наиболее важные настраиваемые параметры RF, которые исследовались в статье.

Таблица 1

Основные настраиваемые параметры Random Forest

№	Параметр	Описание параметра
1	N_estimators	Количество деревьев
2	Критерий примеси	Критерий, используемый для получения функции разбиения (индекс Джини или Энтропия)
3	max_feature	Максимальное количество просмотренных в ветвях признаков для разбиения
4	Max_dept	Глубина дерева – максимальное количество уровней
5	min_samples_1st	Минимальное число выборок в узле, необходимое для разветвления
6	min_samples_leaf	Минимальное число выборок в листе

3. Экстремальный градиентный бустинг (XGBoost)

XGBoost представляет собой бустинг деревьев. При бустинге деревья обучаются последовательно, исправляя ошибки друг друга. Общий алгоритм XGBoost выглядит следующим образом:

1. Предсказание ансамбля алгоритмов (\hat{y}_i) на основе деревьев имеет вид (8):

$$\hat{y}_i = \sum_{k=1}^{n_{\text{estimators}}} f_k(x_i), \quad f_k \in \mathcal{F}, \quad (8)$$

где $n_{\text{estimators}}$ – количество деревьев;

f_k – базовые алгоритмы;

\mathcal{F} – функциональное пространство всех возможных деревьев модели.

2. После получения предсказания от модели, оптимизируется целевая функция (obj) или функция оптимизации бустинга, которая имеет вид (9):

$$obj(\theta) = L(\theta) + \Omega(\theta), \quad (9)$$

где

θ – шаг модели;

$L(\theta) = \sum_{i=1}^n L(y_i, \hat{y}_i^{(t)})$ – специфичная функция потерь (может оцениваться как СКО);

y_i – значение i -го элемента обучающей выборки;

$\hat{y}_i^{(t)}$ – предсказание для первых t деревьев;

$\Omega(\theta) = \sum_{i=1}^t \Omega(f_i)$ – регуляризатор для каждого из деревьев (10):

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2, \quad (10)$$

T – число листьев;

γ – параметр, позволяющий регулярировать деление листа на поддеревья;

λ – параметр регуляризации для суммы весов деревьев;

w_j – вес листьев;

$\frac{1}{2} \lambda \sum_{j=1}^T w_j^2$ – функция, контролирующая сумму весов модели.

В отличие от CART, при котором деревья либо урезаются после их полного построения, либо вводятся ограничения на глубину дерева/листья/вершины, при XGBoost более гибкая система регуляризации. При построении дерева используется функция, на основании которой принимается решение целесообразности о разделении листа на ветки (11):

$$Gain = \frac{1}{2} [H_L(\lambda) + H_R(\lambda) - H_{\text{исх}}(\lambda)] - \gamma, \quad (11)$$

где $H_L(\lambda)$ – информационное усиление на новом левом листе; $H_R(\lambda)$ – информационное усиление на новом правом листе; $H_{\text{исх}}(\lambda)$ – информационное усиление на исходном листе.

Если $Gain < 0$, то деление листа на ветки не происходит, и лист становится терминальным.

3. В модель добавляется новое дерево f_k , которое строится на основании ошибок предыдущего шага [14].

При использовании различных функций потерь получаются различные виды бустинга (AdaBoost при экспоненциальной, LogitBoost при логарифмической, GentleBoost и др.). Градиентный бустинг является обобщением всех этих функций, а экстремальный градиентный бустинг позволяет проводить более гибкую регуляризацию, благодаря чему, XGBoost достигает лучших результатов по сравнению с AdaBoost и прочих подобных методов и применяется значительно чаще в современных задачах.

Таблица 2
Основные настраиваемые параметры XGBoost

№	Параметр	Описание параметра
1	N_estimators	Количество деревьев
2	Max_dept	Глубина дерева – максимальное количество уровней
3	min_child_weight	Минимальное число выборок в листе
4	colsample_bytree	Доля признаков, используемая для построения дерева
5	subsample	Доля выборки, используемая для построения дерева

4. Оценка результатов классификации

На этапе тестирования алгоритма, результаты классификации, полученные с помощью построенной модели, сравниваются с истинными результатами классификации. Для каждого класса подсчитывают количество верно и неверно идентифицированных потоков, и составляется матрица ошибок (рис. 3). На рисунке количество верно идентифицированных потоков обозначено как TP_i , а – неверно F_{ij} , причем i – номер предсказанного класса, а j – номер истинного класса.

Истинные классы: $S=f(B, Z, Y)$

	S_1	S_2	...	S_n
S_1	TP_1	F_{12}	...	F_{1n}
S_2	F_{21}	TP_2	...	F_{2n}
...
S_n	F_{n1}	F_{n2}	...	TP_n

Рис. 3. Матрица ошибок (Confusion matrix)

По составленной матрице ошибок рассчитываются основные параметры алгоритма:

Доля правильных ответов показывает, сколько было правильно идентифицированных потоков относительно всех потоков (12):

$$ACCURACY = \frac{\sum_{i=1}^n TP_{ii}}{\sum_{i=1}^n TP_{ii} + \sum_{i=1}^n \sum_{j=1}^n F_{ij}}; \quad (12)$$

Точность для i -го класса – это доля верно классифицированных потоков от всех потоков, которым была присвоена метка i -го класса (13):

$$PRECISION_i = \frac{TP_{ii}}{TP_{ii} + \sum_j^n F_{ij}}; \quad (13)$$

Полнота для j -го класса – мера выделения одного потока среди других (14):

$$RECALL_j = \frac{TP_{ii}}{TP_{ii} + \sum_i^n F_{ij}}; \quad (14)$$

F_1 -мера – гармоническое среднее между точностью и полнотой (15):

$$F1 = \frac{2 \cdot PRECISION \cdot RECALL}{PRECISION + RECALL}. \quad (15)$$

Все перечисленные параметры находятся в диапазоне [0;1], где 0 – это наихудший, а 1 – наилучший результат соответственно. На практике 1 достигается довольно редко.

4. Составление базы данных трафика

Для проведения экспериментальной части исследования была выбрана открытая база данных исследовательской группы MAWI, разработанная в рамках проекта WIDE [15]. Трафик снимался на реальной сети в Токио, в контрольной точке с соединением 10 Гбит/с. Для актуальности анализируемых трасс, были выбраны 6 записей, сделанные в январе–феврале 2020 года, каждая длительностью 15 минут. Таким образом, общее время измерений составило 1,5 ч, а общий объем трафика – 200,25 Гб.

С помощью автоматического анализатора трафика nDPI и модуля dpkt языка программирования Python удалось разделить данные на отдельные выборки по следующим правилам:

- все пакеты, имеющие одинаковый набор 5-tuple (IP – адрес, порты источника и назначения и используемый транспортный протокол), объединялись в одну выборку, названную **потоком**;

- потоки, представляющие обратное направление (в которых IP-адреса и порты источника и назначения записаны в обратном порядке), не учитывались в итоговой выборке;

- каждый поток содержал в себе ровно 15 пакетов: потоки с меньшим количеством пакетов удалялись, потоки с большим – обрезались;

- для всех выборок записаны размеры каждого из 15 пакетов, а также разность между временем прихода двух последовательно поступающих пакетов, итого 29 параметров;

- для каждого потока было зафиксировано название его источника-приложения, определенного nDPI.

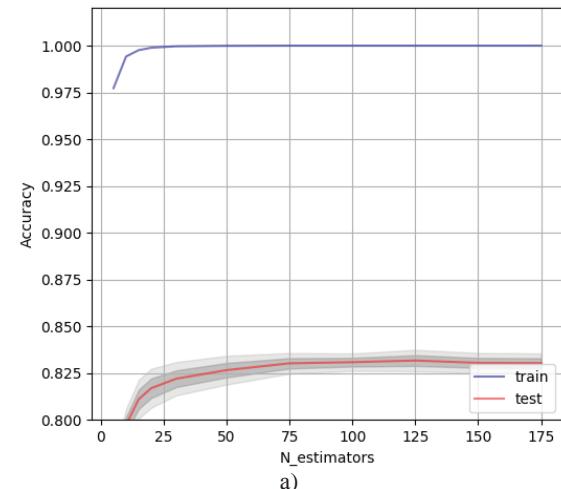
Дальнейший анализ проводился отдельно на базах данных, собранных по TCP и UDP-приложениям. Базу данных TCP представляли по 1500 потоков в каждом из 13 приложений: SSL, HTTP_Proxy, DNS, Apple, IMAPS, HTTP, Skype, SSH, SMTP, RTMP, Telnet, POP3 и IMAP. Базу данных UDP составляли по 250 потоков каждого из 8 приложений: DNS, NTP, Quic, IPsec, SNMP, NetBIOS, STUN и UPnP. Подробно

о предварительной обработке данных, в т.ч. о нормализации параметров модели и работе с выбросами описано в [16].

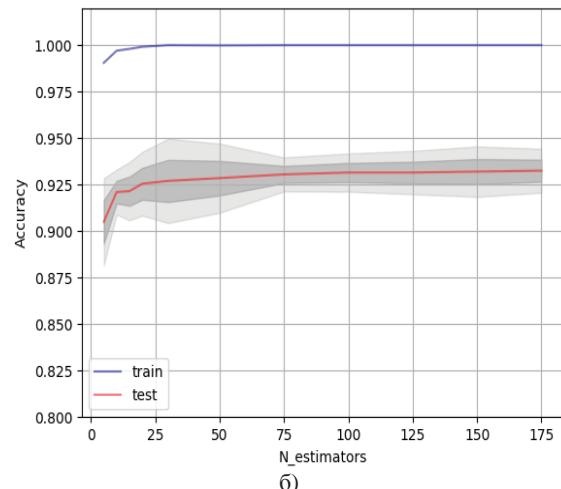
Для обеих баз данных была рассчитана матрица признаков, состоящая из 45 параметров: размеры и время между приходом двух соседних пакетов (1-29), средний, минимальный, максимальный, медианный, суммарный, СКО и дисперсия размера пакета (30-36), то же для времени между поступлением пакетов (37-43), пакетная и байтовые скорости (44-45) [17]. Такой набор параметров удобно собрать и рассчитать в режиме реального времени, его эффективность была показана автором в работе [18], а способ снятия, не вносящий дополнительных задержек в сеть в статьях [19-20].

5. Настройка параметров модели Random Forest

На рисунке 4 изображена зависимость точности по методу ОOB для тестовой и обучающей выборки в зависимости от количества деревьев ($N_{estimators}$), а на рисунке 5 – зависимость ошибки ОOB от количества деревьев, максимальному количеству просмотренных в ветвях признаках ($max_features$) и критерию примеси (индекс Джини или Энтропия). Темно-серым отмечена область, в которой находятся значения в пределах стандартного отклонения, а светло – серым в пределах максимального. В качестве возможных вариантов $max_features$ рассматриваются стандартные варианты: $max_features=\sqrt{m}$, $max_features=\log_2 m$ и $max_features=m$.



а)



б)

Рис. 4. Зависимость точности классификации от количества деревьев ($N_{estimators}$): а) TCP-потоки, б) UDP-потоки

Как видно из рисунка 4, для TCP-потоков наивысшая точность 0,83 достигается при 125 деревьях и выше, а для UDP – 0,93 при 175 деревьях и выше. При дальнейшем росте числа деревьев точность практически не меняется, но значительно увеличивается сложность расчетов и время построения модели. Результаты тренировочного TCP-набора принимают значение 1, а тестового – незначительно колеблются вблизи 0,82. Это говорит о том, что разброс в ошибке составляет около 18%, а модель является переобученной. В случае с UDP ошибка 8%, но модель также является переобученной. Для того чтобы справиться с этой проблемой и уменьшить разброс между тестовыми и тренировочными результатами, необходимо воспользоваться механизмами регуляризации параметров.

При 125 деревьях для TCP-потоков небольшой ОOB-ошибки можно достичь при $\max_features=\sqrt{m}$ с индексом Джини или $\max_features=m$ с Энтропией. При увеличении количества деревьев ошибка незначительно снижается, но сложность линейно возрастает. Для UDP-потоков при 175 деревьях наименьшая ОOB – ошибка достигается при $\max_features=\sqrt{m}$ с Энтропией или $\max_features=\log_2 m$ с индексом Джини.

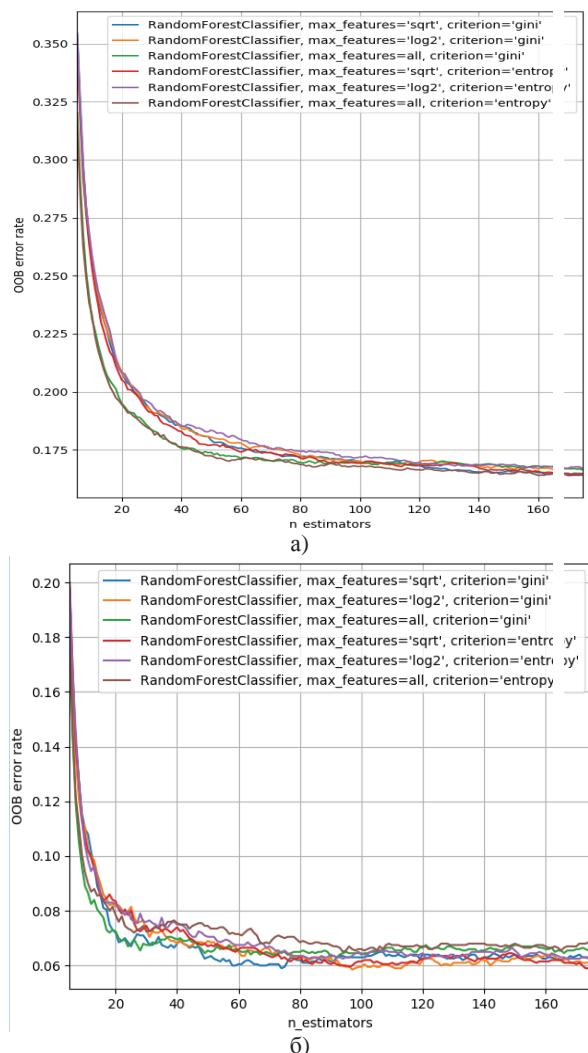


Рис. 5. Зависимость ОOB-ошибки классификации от количества деревьев ($n_{estimators}$), максимального числа признаков для разветвления и критерия примеси (индекс Джини или Энтропия):
а) TCP-потоки, б) UDP – потоки

На рисунке 6 изображены валидационные кривые для TCP и UDP-потоков в зависимости от глубины деревьев. С увеличением глубины деревьев точность классификации возрастает. Для TCP-потоков $\max_depth=25$, а для UDP-потоков с 17.

На рисунке 7 кривые точности для минимального числа выборок в узле, а на рисунке 8 – для минимального числа выборок в листе. При увеличении этих параметров точность классификации незначительно уменьшается. Для TCP-потоков $\min_samples_split=2$, а для UDP – 4. Для обоих наборов данных $\min_samples_leaf=1$.

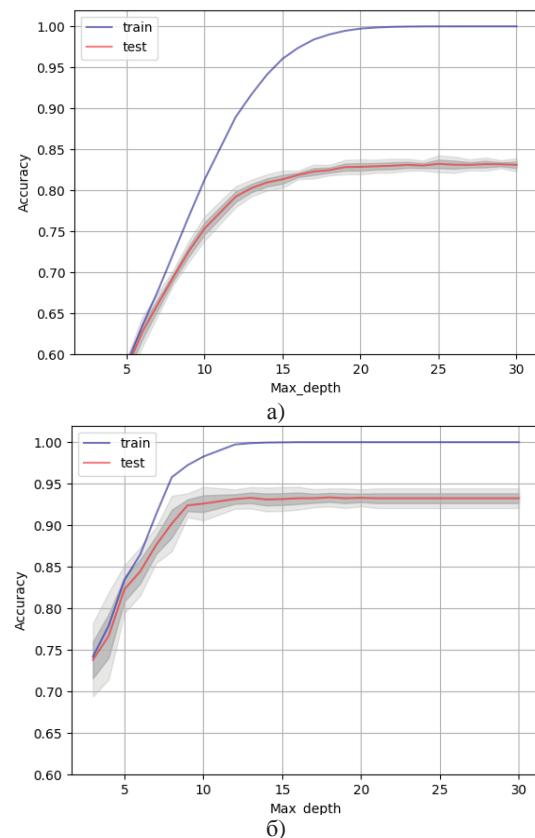
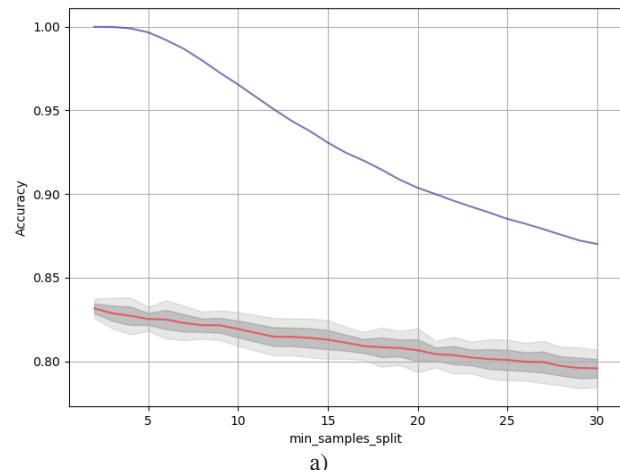


Рис. 6. Зависимость точности классификации от максимальной глубины деревьев (\max_depth): а) TCP-потоки, б) UDP – потоки



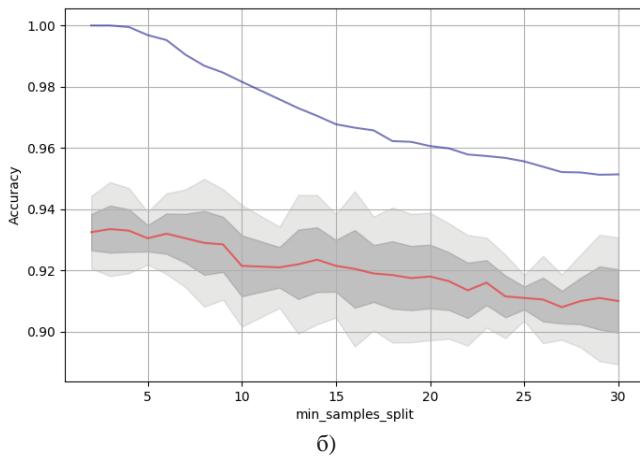


Рис. 7. Зависимость точности классификации минимального числа выборок в узле (min_samples_split): а) TCP-потоки, б) UDP – потоки

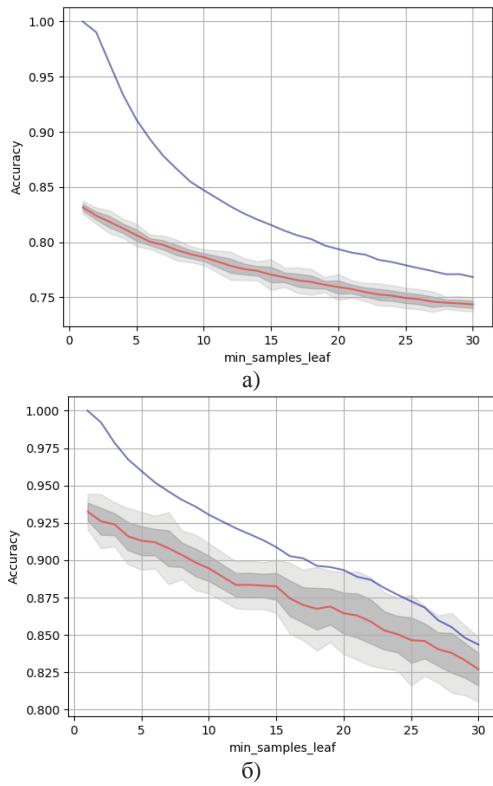


Рис. 8. Зависимость точности классификации от минимального количества выборок в листе (min_samples_leaf): а) TCP-потоки, б) UDP – потоки

Но, несмотря на уменьшение точности, разброс между результатами тестовой и обучающей выборки тоже уменьшается и модель уже не такая переобученная. Поэтому для TCP-потоков предлагается использовать max_depth=19, min_samples_split=7, min_samples_leaf=1, сохраняя при этом точность 0,82, но уменьшая переобучение на 18%. Немного увеличив количество деревьев и количество максимальных признаков до max_features=13, n_estimators=175, точность станет равной 0,87, т.е. увеличится на 5%.

Аналогичным образом для UDP-потоков можно определить: n_estimators=190, max_depth=8, min_samples_split=6 и min_samples_leaf=1.

6. Настройка параметров модели XGBoost

Для реализации алгоритма XGBoost используется одноименная сторонняя библиотека [14].

На рисунке 9 изображены валидационные кривые, показывающие точность классификации в зависимости от числа деревьев для TCP и UDP-потоков. В отличие от Random Forest, для достижения высокой точности классификации не требуется большого количества деревьев и разброс между тестовой и обучающей выборкой не так велик. При построении деревьев с максимальной глубиной равной 5, точность 80% достигается при 50 деревьях для TCP и 90% при 5 деревьях для UDP.

На рисунке 10 изображены зависимости точности классификации от максимальной глубины деревьев. Графики для XGBoost очень похожи на аналогичные графики для Random Forest (рис. 6), но рост точности для XGBoost начинается при значительно меньших значениях max_depth: 7-10 для TCP и 3-5 для UDP.

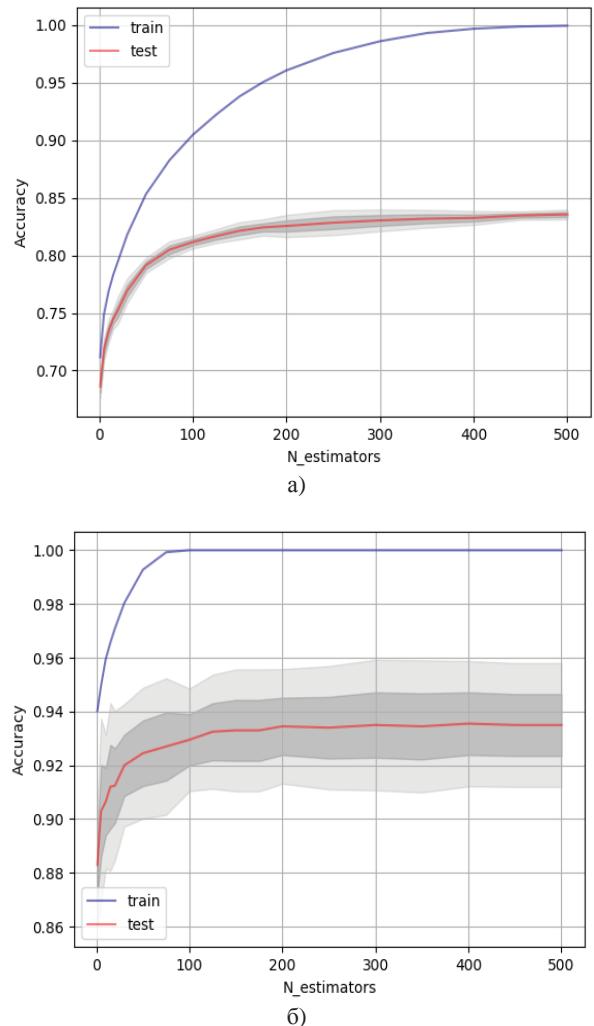


Рис. 9. Зависимость точности классификации от количества деревьев (N_estimators): а) TCP-потоки, б) UDP – потоки

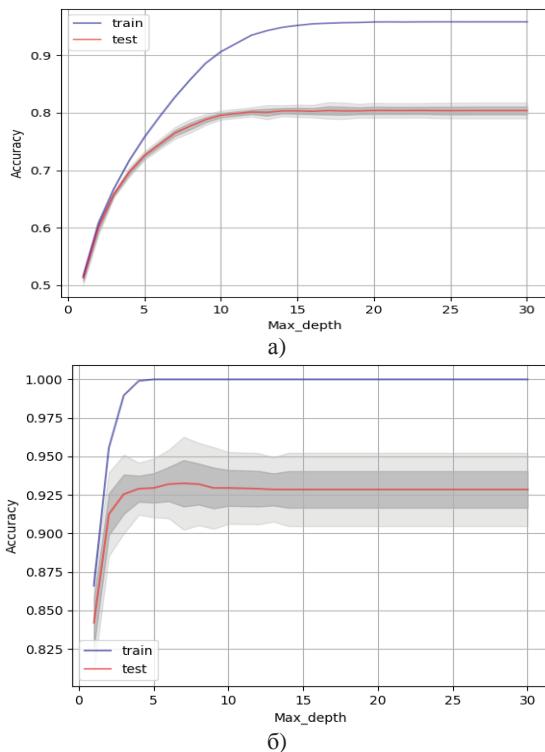


Рис. 10. Зависимость точности классификации от максимальной глубины деревьев (Max_depth): а) TCP-потоки, б) UDP – потоки

На рисунке 11 показаны валидационные прямые для min_child_weight – параметра, отвечающего за минимальное число выборок в каждой вершине – аналогично min_samples_leaf для Random Forest. Также как и в случаях с Random Forest, этот параметр помогает справиться с переобучением.

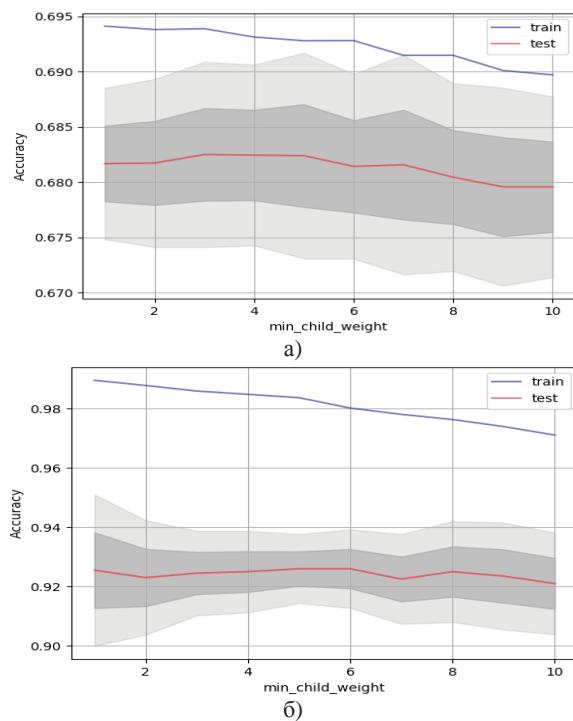


Рис. 11. Зависимость точности классификации от минимального числа выборок в каждой вершине (min_child_weight):
а) TCP-потоки, б) UDP – потоки

На рисунке 12 изображены зависимости точности классификации трафика от доли признаков, используемых для построения деревьев (colsample_bytree), аналогично max_features для Random Forest, а на рисунке 13 – зависимости точности классификации от доли выборки для построения деревьев (subsample), аналогично методу ОOB.

Параметры colsample_bytree и subsample добавляют случайность в процесс обучения, благодаря этому, при изменении этих долей, можно добиться лучших результатов моделирования.

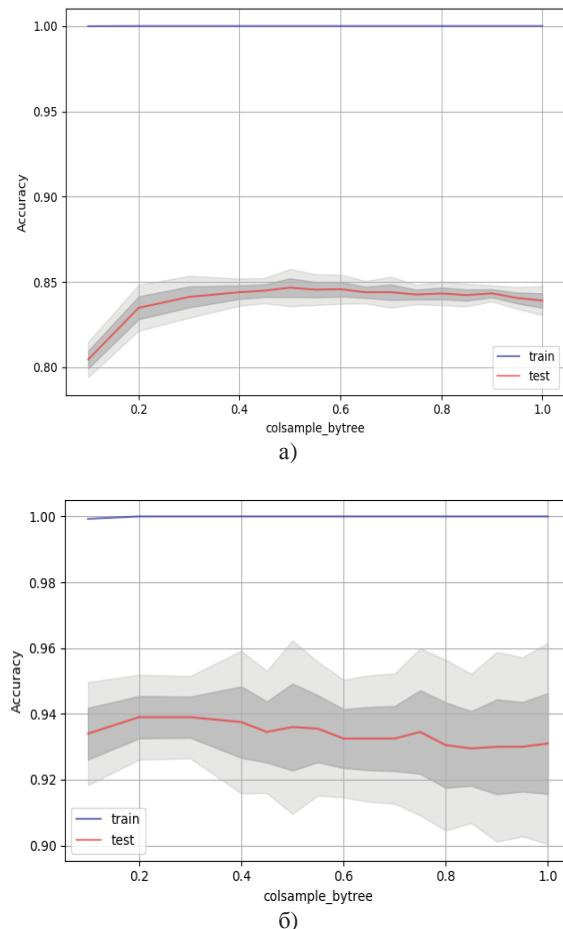


Рис. 12. Зависимость точности классификации от доли признаков, используемых для обучения каждого дерева (colsample_bytree):
а) TCP-потоки, б) UDP – потоки

Путем подбора параметров, можно определить те, при которых достигается наилучшая точность. Для TCP-потоков точность 0,9005 может быть получена при n_estimators=155, max_depth=10, colsample_bytree=0,65, min_child_weight=1, subsample=0,65; а для UDP-потоков точность 0,9675 – при n_estimators=85, max_depth=7, colsample_bytree=0,4, min_child_weight=2 и subsample=0,65.

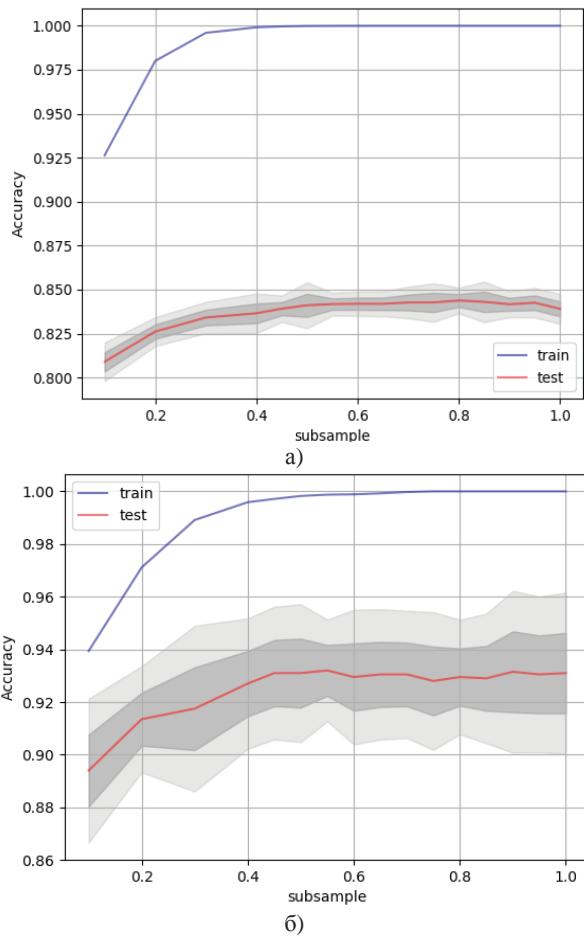


Рис. 13. Зависимость точности классификации от доли выборки, используемой для обучения каждого дерева (subsample):
а) TCP-потоки, б) UDP – потоки

7. Результаты классификации трафика до и после настройки параметров

На рисунках 14-17 представлены результаты классификации TCP и UDP потоков с параметрами по умолчанию и параметрами, которые были получены при изучении математической модели Random Forest и XGBoost.

Для TCP-потоков улучшение результатов точности при настройке RF 3,3%, а при настройке XGBoost – 6,7%. В случаях с UDP-потоками, настройка RF повышает точность классификации на 1,5%, а настройка XGBoost – на 2%. При рассмотрении результатов отдельных приложений можно заметить, что настройка RF повышает Precision до 12% (для POP3), а UDP – до 11% (для NTP). Настройка XGBoost для TCP повышает Recall до 19% для POP3 и SSL, а для UDP – до 12% (для NTP). На основе этих значений можно сделать вывод, что настройка математических параметров модели является важным этапом при создании классификатора трафика. Для классификации TCP-потоков влияние настройки параметров оказалось большим, чем при классификации UDP-потоков. Это объясняется тем, что и при ненастроенных параметрах для UDP-потоков были показаны хорошие результаты, иногда достигающие 100%. С одной стороны, классификаторы для UDP учатся быстрее, а с другой стороны – для TCP выбрано 13 приложений против 8 для UDP, что усложняет задачу классификации и обучения.

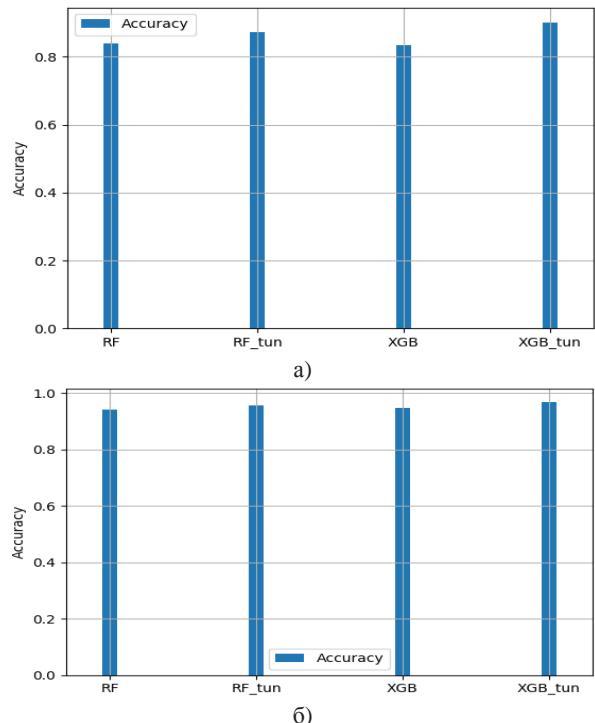


Рис. 14. Точность классификации (Accuracy) при использовании алгоритмов Random Forest и XGBoost до и после настройки модели:
а) TCP-потоки, б) UDP – потоки

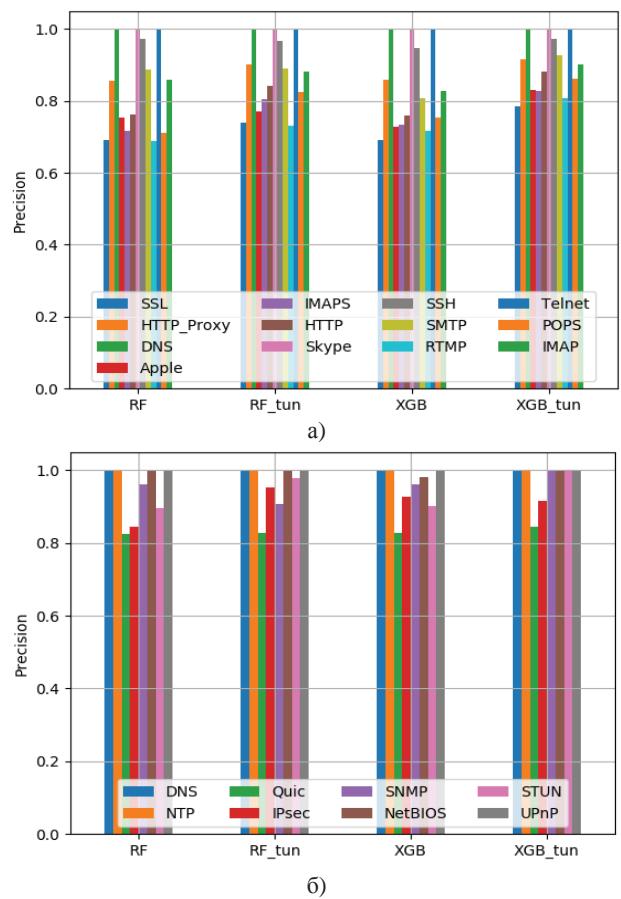
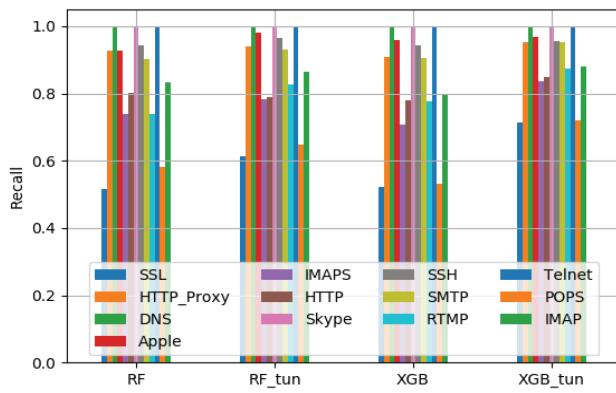
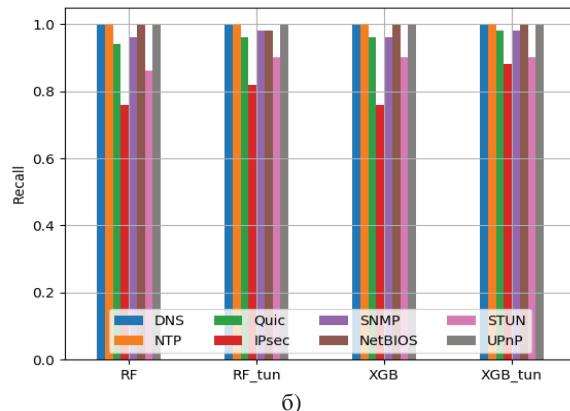


Рис. 15. Точность классификации каждого класса (Precision) при использовании алгоритмов Random Forest и XGBoost до и после настройки модели. а) TCP-потоки, б) UDP – потоки

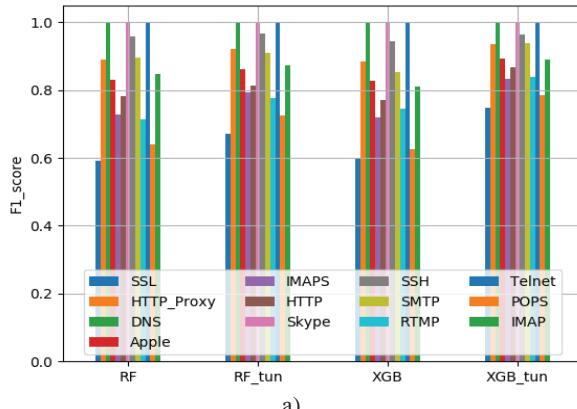


а)

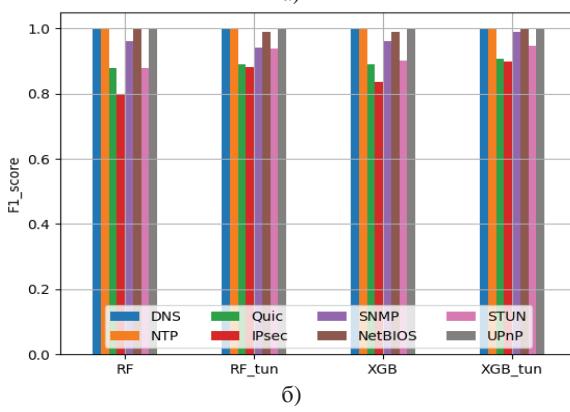


б)

Рис. 16. Полнота классификации каждого класса (Recall) при использовании алгоритмов Random Forest и XGBoost до и после настройки модели. а) TCP-потоки, б) UDP – потоки



а)



б)

Рис.17. F1-мера классификации каждого класса (F1) при использовании алгоритмов Random Forest и XGBoost до и после настройки модели. а) TCP-потоки, б) UDP – потоки

В большинстве работ, связанных с классификацией трафика, наилучшие результаты перед всеми остальными алгоритмами показывает алгоритм Random Forest [3-5]. В [6] показано, что один из алгоритмов бустинга на основе деревьев (AdaBoost) в целом показывает результаты хуже, по сравнению с Random Forest. Но, судя по проведенному эксперименту заметно, что до настройки параметров модели XGBoost по результатам уступал даже ненастроенному RF, а после настройки – опередил и настроенный, и ненастроенный RF.

Заключение

Одной из проблем классификаторов, построенных на основе решающего дерева, является склонность к переобучению, т.е. возможность хорошо работать на обучающей выборке и не так хорошо на тестовой. Определить наличие переобучения можно, построив валидационные кривые для обучающей и тестовой выборки. Понизить переобученность модели позволяет уменьшение глубины дерева, а также увеличение минимального числа выборок в листьях и узлах. Увеличение числа деревьев приводит к росту точности классификации.

Настройка гиперпараметров моделей позволяет повысить точность классификации отдельных приложений на 19% для TCP и на 12% для UDP – потоков, как для RF, так и для XGBoost. Несмотря на то, что в большинстве работ, посвященных классификации трафика методами машинного обучения, наиболее эффективным оказался алгоритм Random Forest, после настройки параметров выяснилось, что алгоритм XGBoost показал лучшие результаты по сравнению с Random Forest.

Результаты статьи использовались для построения классификаторов трафика в режиме реального времени с целью определения QoS [16-20].

Литература

1. Маньков В.А., Краснова И.А. Алгоритм динамической классификации потоков в мультисервисной SDN-сети // Т-Comm: Телекоммуникации и транспорт. 2017. Том 11. №12. С. 37-42.
2. Маньков В.А., Краснова И.А. Задача управления трафиком с динамическим определением QoS в мультисервисных SDN сетях // Сборник трудов XI Международной отраслевой научно-технической конференции «Технологии информационного общества». (15-16 марта 2017 г. Москва, МТУСИ). М.: ООО «ИД Медиа Паблишер», 2017 г., с.67-68
3. Iwai T., Nakao A. (2016). Adaptive mobile application identification through in-network machine learning. 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), 1-6. <https://doi.org/10.1109/APNOMS.2016.7737226>
4. Anantavrasilp I., Scholer T. (2007). Automatic flow classification using machine learning. 2007 15th International Conference on Software, Telecommunications and Computer Networks, 1-6. <https://doi.org/10.1109/SOFTCOM.2007.4446129>.
5. Шелухин О.И., Ерохин С.Д., Ванюшина А.В. Классификация IP-трафика методами машинного обучения / Под ред. О.И.Шелухина. М.: Горячая линия – Телеком, 2018. 282 с: ил.
6. Шелухин О.И., Симонян А.Г., Ванюшина А.В. Влияние структуры обучающей выборки на эффективность классификации приложений трафика методами машинного обучения // Т-Comm: Телекоммуникации и транспорт. 2017. Том 11. №2. С. 25-31.

7. Probst P., Boulesteix A., Bischl B. (2019). Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *J. Mach. Learn. Res.*, 20, 53:1-53:32.
8. Hong Y., Huang C., Nandy B., Seddigh N. (2015). Iterative-tuning support vector machine for network traffic classification. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 458-466.
9. Bansal A.K., Kaur S. (2018). Extreme Gradient Boosting Based Tuning for Classification in Intrusion Detection Systems.
10. Деарт В.Ю., Маньков В.А., Краснова И.А. «Анализ перспективных подходов и исследований по классификации потоков трафика для поддержания QoS методами ML в SDN-сетях» // Вестник СибГУТИ. 2021. №1.
11. Breiman L., Friedman J., Olshen R., Stone C. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.
12. Buitinck L., Louppe G., Blondel M., Pedregosa F., Mueller A., Grisel O., Niculae V., Prettenhofer P., Gramfort A., Grobler J., Layton R., Vander Plas J., Joly A., Holt B., Varoquaux G. (2013). API design for machine learning software: experiences from the scikit-learn project. ArXiv, abs/1309.0238.
13. Breiman L. (2001). Random Forests. *Machine Learning*, 45, 5-32.
14. Introduction to Boosted Trees // URL: <https://xgboost.readthedocs.io/en/latest/tutorials/model.html> (Дата обращения 03.03.2020)
15. Cho K., Mitsuya K., Kato A. (2000). Traffic Data Repository at the WIDE Project. USENIX Annual Technical Conference, FREENIX Track.
16. Mankov V.A., Deart V.Y., Krasnova I.A. (2021) Evaluation of the Effect of Preprocessing Data on Network Traffic Classifier Based on ML Methods for QoS Predication in Real-Time. In: Hu Z., Petoukhov S., He M. (eds) Advances in Artificial Systems for Medicine and Education IV. AIMEE 2020. Advances in Intelligent Systems and Computing, vol 1315. Springer, Cham. https://doi.org/10.1007/978-3-030-67133-4_5
17. Deart V., Mankov, Krasnova I., Development of a Feature Matrix for Classifying Network Traffic in SDN in Real-Time Based on Machine Learning Algorithms, 2020 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC), Moscow, Russia, 2020, pp. 1-9, doi: 10.1109/MoNeTeC49726.2020.9258314
18. Deart V., Mankov, Krasnova I., Agglomerative Clustering of Network Traffic Based on Various Approaches to Determining the Distance Matrix, 2021 28th Conference of Open Innovations Association (FRUCT), Moscow, Russia, 2021, pp. 81-88, doi: 10.23919/FRUCT50888.2021.9347616
19. Маньков В.А., Краснова И.А. (2019). «Классификация потоков трафика SDN-сетей методами машинного обучения в режиме реального времени» // Труды международной научно-технической конференции «Информационные технологии и математическое моделирование систем 2019», С.65-68. <https://doi.org/10.36581/CITP.2019.31.51.016>
20. Mankov V.A., Krasnova I.A. (2020) Collection of Individual Packet Statistical Information in a Flow Based on P4-switch. In: Hu Z., Petoukhov S., He M. (eds) Advances in Intelligent Systems, Computer Science and Digital Economics. CSDEIS 2019. Advances in Intelligent Systems and Computing, vol 1127. Springer, Cham. https://doi.org/10.1007/978-3-030-39216-1_11

ANALYSIS OF THE INFLUENCE OF MACHINE LEARNING ALGORITHM PARAMETERS ON THE RESULTS OF TRAFFIC CLASSIFICATION IN REAL TIME

Irina A. Krasnova, MTUCI, Moscow, Russia, irina_krasnova-angel@mail.ru

Abstract

The paper analyzes the impact of setting the parameters of Machine Learning algorithms on the results of traffic classification in real-time. The Random Forest and XGBoost algorithms are considered. A brief description of the work of both methods and methods for evaluating the results of classification is given. Experimental studies are conducted on a database obtained on a real network, separately for TCP and UDP flows. In order for the results of the study to be used in real time, a special feature matrix is created based on the first 15 packets of the flow. The main parameters of the Random Forest (RF) algorithm for configuration are the number of trees, the partition criterion used, the maximum number of features for constructing the partition function, the depth of the tree, and the minimum number of samples in the node and in the leaf. For XGBoost, the number of trees, the depth of the tree, the minimum number of samples in the leaf, for features, and the percentage of samples needed to build the tree are taken. Increasing the number of trees leads to an increase in accuracy to a certain value, but as shown in the article, it is important to make sure that the model is not overfitted. To combat overfitting, the remaining parameters of the trees are used. In the data set under study, by eliminating overfitting, it was possible to achieve an increase in classification accuracy for individual applications by 11-12% for Random Forest and by 12-19% for XGBoost. The results show that setting the parameters is a very important step in building a traffic classification model, because it helps to combat overfitting and significantly increases the accuracy of the algorithm's predictions. In addition, it was shown that if the parameters are properly configured, XGBoost, which is not very popular in traffic classification works, becomes a competitive algorithm and shows better results compared to the widespread Random Forest.

Keywords: Machine Learning, Random Forest, XGBoost, QoS, traffic classification, parameter tuning, overtraining.

References

1. V.A. Mankov, I.A. Krasnova (2017), "Algorithm for dynamic classification of flows in a multiservice software defined network", *T-Comm*, vol. 11, no. 12, pp. 37-42.
2. V.A. Mankov and I.A. Krasnova (2017), "Zadacha upravleniya trafikom s dinamicheskim opredeleniem QoS v multiservisnyh SDN setyah", *Sbornik trudov XI Mezhdunarodnoj otrazhenoj nauchno-tehnicheskoy konferencii "Tekhnologii informacionnogo obshchestva"*, pp. 67-68.
3. T. Iwai, A. Nakao (2016). Adaptive mobile application identification through in-network machine learning. *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 1-6. <https://doi.org/10.1109/APNOMS.2016.7737226>
4. I. Anantavrasilp, T. Scholer (2007). Automatic flow classification using machine learning. *2007 15th International Conference on Software, Telecommunications and Computer Networks*, 1-6. <https://doi.org/10.1109/SOFTCOM.2007.4446129>.
5. O.I. Sheluhin, S.D. Erohin and A.V. Vanyushina (2018), "Klassifikaciya IP-trafika metodami mashinnogo obucheniya", Moscow: Goryachaya liniya – Telekom.
6. O.I. Sheluhin, A.G. Simonyan and A.V. Vanyushina (2017), "Influence of training sample structure on traffic application efficiency classification using machine-learning methods", *T-Comm*, vol. 11, no. 2, pp. 25-31.
7. P. Probst, A. Boulesteix, B. Bischl (2019). Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *J. Mach. Learn. Res.*, 20, 53:1-53:32.
8. Y. Hong, C. Huang, B. Nandy, N. Seddigh (2015). Iterative-tuning support vector machine for network traffic classification. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 458-466.
9. A.K. Bansal, S. Kaur (2018). Extreme Gradient Boosting Based Tuning for Classification in Intrusion Detection Systems.
10. V. Yu. Deart, V. A. Mankov, I. A. Krasnova (2021), "Analysis of promising approaches and research on traffic flow classification for maintain QoS using ML methods in SDN networks". *Vestnik SibGUTI*. No.1.
11. L. Breiman, J. Friedman, R. Olshen, C. Stone (1984). *Classification and Regression Trees*. Wadsworth, Belmont, CA.
12. L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vander Plas, A. Joly, B. Holt, G. Varoquaux (2013). API design for machine learning software: experiences from the scikit-learn project. *ArXiv*, abs/1309.0238.
13. L. Breiman (2001). Random Forests. *Machine Learning*, 45, pp. 5-32.
14. Introduction to Boosted Trees // URL: <https://xgboost.readthedocs.io/en/latest/tutorials/model.html> (Date of access 03.03.2020)
15. K. Cho, K. Mitsuya, A. Kato (2000). Traffic Data Repository at the WIDE Project. *USENIX Annual Technical Conference*, FREENIX Track.
16. V.A. Mankov, V.Y. Deart, I.A. Krasnova (2021). Evaluation of the Effect of Preprocessing Data on Network Traffic Classifier Based on ML Methods for Qos Predication in Real-Time. In: Hu Z., Petoukhov S., He M. (eds) *Advances in Artificial Systems for Medicine and Education IV. AIMEE 2020. Advances in Intelligent Systems and Computing*, vol 1315. Springer, Cham. https://doi.org/10.1007/978-3-030-67133-4_5
17. V. Deart, Mankov, I. Krasnova (2020), Development of a Feature Matrix for Classifying Network Traffic in SDN in Real-Time Based on Machine Learning Algorithms, *2020 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC)*, Moscow, Russia, pp. 1-9, doi: [10.1109/MoNeTeC49726.2020.9258314](https://doi.org/10.1109/MoNeTeC49726.2020.9258314)
18. V. Deart, Mankov, I. Krasnova (2021), Agglomerative Clustering of Network Traffic Based on Various Approaches to Determining the Distance Matrix, *2021 28th Conference of Open Innovations Association (FRUCT)*, Moscow, Russia, pp. 81-88, doi: [10.23919/FRUCT50888.2021.9347616](https://doi.org/10.23919/FRUCT50888.2021.9347616)
19. V.A. Mankov and I.A. Krasnova (2019), "Klassifikatsiya potokov trafika SDN-setei metodami mashinnogo obucheniya v rezhime real'nogo vremeni", *Informatsionnye Tekhnologii i Matematicheskoe Modelirovaniye Sistem 2019*, [online] Available: <https://doi.org/10.36581/CITP.2019.31.51.016>.
20. V.A. Mankov and I.A. Krasnova (2020) Collection of Individual Packet Statistical Information in a Flow Based on P4-switch. In: Hu Z., Petoukhov S., He M. (eds) *Advances in Intelligent Systems, Computer Science and Digital Economics. CSDEIS 2019. Advances in Intelligent Systems and Computing*, vol 1127. Springer, Cham. https://doi.org/10.1007/978-3-030-39216-1_11

Information about author:

Irina A. Krasnova, graduate student, MTUCI, Moscow, Russia