Санкт-Петербургский государственный университет

Кафедра системного программирования

Васенина Анна Игоревна

Использование методов машинного обучения для предсказания сбоев в кластерах хранения данных

Курсовая работа

Научный руководитель:
д.ф.-м.н., проф. Терехов А.Н.

Консультант:
Руководитель исследовательской лаборатории ООО "Рэйдикс", к.т.н. Лазарева С.В.

Санкт-Петербург
2018

Оглавление

Введение

Количество хранимых на электронных устройствах данных постоянно растет, что приводит ко все большей популяризации облачных технологий и систем хранения данных. С ростом объемов информации, содержащейся в кластерах хранения данных, сложнее становится обеспечивать бесперебойную работу и сохранность данных на серверах, однако, потеря данных всё ещё является абсолютно недопустимой. В таких условиях остро встает вопрос поиска новых методов предсказания нарушений в работе кластеров хранения данных.

В силу внешних и внутренних факторов в процессе использования устройств хранения данных может происходить замедление работы или отказ системы. Возникающие ошибки могут быть разделены на две группы: аппаратные и программные. Для поддержания постоянной работы кластера хранения данных необходимо предвосхищать как можно больший круг возможных сбоев.

Борьба с отказами устройств, предназначенных для хранения информации, стала актуальна еще со времен появления первых электронно-вычислительных машин. В 1992 году в дисковых массивах IBM 9337 появляется система диагностики S.M.A.R.T, вошедшая в состав протокола SATA, предназначенная для диагностики механических сбоев. Она используется в компаниях, специализирующихся на системах хранения данных, до сих пор [6].

Однако, современные кластеры хранения данных, включающие не только устройства хранения данных, но также устройства анализа (CPU, RAM) и свою операционную систему, нуждаются в диагностике не только аппаратных, но и программных сбоев. Одним из способов предсказания программных сбоев в кластерах хранения данных является детектирование аномалий [5][10][4] во временных рядах различных показателей. Однако, все существующие в открытом доступе подходы не учитывают одного из существенных аспектов,анализируемых при возникновении сбоя - сообщения, появлявшиеся в логе. На данный момент анализ логов производится исключительно вручную системными адми-

нистраторами.

Возможность собирать и анализировать статистику из логов даёт нам почву для использования методов предиктивной аналитики с целью определять паттерны, предшествующие сбою, и избегать его появления.

Статистический подход к предсказанию хорошо показывает себя в применении к веб-сервисам, таким как ebay [3] и twitter [9]. Данная работа является попыткой выделить некоторые специфические признаки и использовать их для предсказания возможных сбоев в кластере хранения данных.

Актуальность ансамблей решающих деревьев

Мы изначально предполагаем наличие в наших данных неявных закономерностей, из-за которых анализ невозможно проводить тривиальными методами. Это делает для нас актуальным использование методов машинного обучения. Однако, так как сбои в работе систем хранения данных происходят достаточно редко, одна из трудностей, с которыми нам пришлось столкнуться при выполнении этой работы - невозможность использования методов глубокого обучения из-за недостаточного количества данных. На этом фоне наиболее подходящим для нас являлось решение на основе ансамблей решающих деревьев.

1. Постановка задачи

Целью работы является исследование применимости алгоритмов машинного обучения на базе ансамблей решающих деревьев к анализу логов, собираемых в кластерах хранения данных RAIDIX .

Для её выполнения были поставлены следующие задачи:

- Изучить существующие методы машинного обучения на базе решающих деревьев;
- Выработать методику анализа сообщений в логах;
- Подготовить обучающую выборку;
- Провести эксперименты;
- Проанализировать результаты.

2. Обзор

В данной работе мы рассматриваем задачу классификации логов на “хорошие” и “плохие” при помощи методов машинного обучения. При этом стоит иметь ввиду значительный перевес класса “хороших” относительно “плохих” по количеству элементов. Для решения данного вопроса можно использовать алгоритмы различных типов, краткий обзор которых представлен в данном разделе.

2.1. Обучение без учителя

Обучением без учителя называется процесс машинного обучения, при котором система самостоятельно обучается выполнять поставленную задачу без вмешательства со стороны экспериментатора.

Одним из примеров обучения без учителя, применимых к нашей теме являются алгоритмы, разработанные для задачи поиска аномалий во временных рядах, подробный обзор которых можно найти в специализированной литературе [1]. Хорошим примером такого алгоритма является Isolation forest [8], разбивающий пространство признаков, прибегая к построению некоторого количества случайных бинарных решающих деревьев. Ответом дерева (anomaly_score) считается глубина листа в построенном дереве. Утверждается, что аномальным точкам свойственно оказываться в листьях, близких к корню. Существенными для нас преимуществами этого алгоритма являются:

- Алгоритм распознаёт аномалии различных видов: как изолированные точки с низкой локальной плотностью, так и кластеры аномалий малых размеров;
- Сложность изолирующего дерева – O(n log n), что эффективнее большинства других алгоритмов;
- Алгоритм не требует существенных затрат по памяти, в отличие от, например, метрических методов, зачастую требующих построения матрицы попарных расстояний;

- Отсутствуют параметры, требующие подбора;
- Алгоритм не требует задания метрики или другой априорной информации об устройстве данных.

2.2. Обучение с учителем

В противовес обучению без учителя можно поставить алгоритмы обучения с учителем, когда для каждого элемента обучающей выборки мы принудительно задаём результат. А от алгоритма требуется найти закономерности между стимулами и реакциями системы. Примером таких методов машинного обучения являются ансамбли деревьев решений XGBoost [2] и RandomForest [7], демонстрирующие хорошие результаты во многих задачах анализа данных. Эти методы решения также основаны на построении некоторого количества случайных бинарных решающих деревьев. В отличии от алгоритма Isolation Forest, здесь абсолютно не важна глубина листа ответа, важен только класс, который выдаёт дерево как ответ. Общее решение алгоритма определяется голосованием всех решающих деревьев. Существенными для нас преимуществами данных алгоритмов являются:

- Высокая скорость обучения;
- Высокое качество получаемых моделей (сравнимое с нейронными сетями и ансамблями нейронных сетей);
- Малое количество настраиваемых параметров.

3. Описание работы

3.1. Инструментарий

Для выполнения данной работы был выбран язык программирования python, являющийся стандартным инструментом при анализе данных и машинном обучении ввиду наличия крайне удобных инструментов, а именно:

- Библиотек для визуализации: matplotlib, seaborn;
- Библиотек для анализа данных: numpy, pandas;
- Библиотек для машинного обучения: sklearn, XGBoost.

3.2. Обработка данных

Данные - это сообщения в логах, содержащие в себе:

- Время появления сообщения;
- Источник сообщения;
- Содержательную часть.

Однако, на ранних этапах работы нам стало ясно, что для нас существенными являются не отдельные сообщения, а серии сообщений, так как информации для анализа каждого отдельного сообщения нам не хватает (например, некоторые ошибки даже не появлялись в представленных логах), а также из-за наличия между сообщениями сложных связей, которые тяжело и затратно вычислять при анализе в онлайн-режиме. При этом подходе существенным является способ разделения текста на блоки.

3.2.1. Ошибки первого рода

Назовём отрицательными (negativ, 1) результатами, блоки, за которыми последовал сбой системы. Положительными (positive, 0) результатами - блоки, в следующем после которых участке сбоя не произошло.

Для нас крайне важным является с высокой точностью предсказывать отрицательные результаты, то есть не допускать ошибок первого рода, то есть ложных положительных (FP, false positives) прогнозов. Поэтому основной метрикой оценки нашего предсказания будет являться параметр TNR (true negative rate).

$$TNR = \frac{TN}{TN + FP} \tag{1}$$

Второстепенным критерием будет являться TPR (true positive rate).

$$TPR = \frac{TP}{TP + FN} \tag{2}$$

В этом случае для нас оказывается выгодным при использовании классификатора задать некоторый порог подозрительности. в зависимости от которого мы будем определять вектор как отрицательный. Зависимость качества предсказания от выбранного порога можно увидеть на рис.1

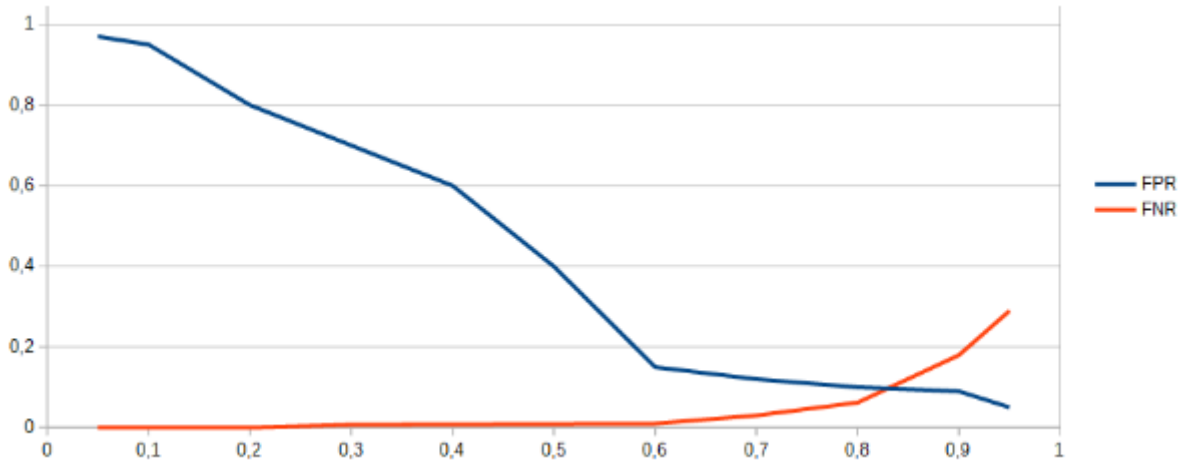

Рис. 1: Зависимость вероятности выдачи ложного положительного и ложного отрицательного прогнозов в зависимости от барьера вероятности, после которого прогноз считается положительным. Ось x: барьер вероятности, ось y: вероятность совершения ошибки. Синяя линия — вероятность совершения ложной положительной ошибки, красная линия — вероятность совершения ложной отрицательной ошибки.

3.2.2. Представление данных

При разбиении логов на блоки, важным параметром для нас является критерий разбиения. Это может быть либо время, либо стековое расстояние. Так как мы решаем задачу предсказания сбоя в следующем участке, а участки внутри одного блока по стековому расстоянию могут крайне разниться по времени, или напротив укладываться в очень короткий промежуток, в качестве параметра разбиения был выбран некоторый промежуток времени T. Результатом такого разбиения является некоторый, возможно пустой, набор сообщений.

Вектором X для набора сообщений назовём следующую совокупность:

- Процентное содержание плохих сообщений;
- Среднее расстояние между плохими сообщениями (в минутах)
- Среднее расстояние между хорошими сообщениями (в минутах)
- Среднее количество хороших сообщений между плохими
- Однородность (homogeneous) плохих сообщений
- Однородность (homogeneous) хороших сообщений

$$Homogeneous = max \frac{Repeats\_of\_message\_}{Total\_messages} \tag{3}$$

3.2.3. Анализ признаков

Одна из первых наших идей была такой: появляющиеся в логах ошибки накапливаются и в итоге выливаются в сбой. Для этого были построены графики зависимости накопления количества ошибок. Мы ожидали увидеть на таких графиках (рис. 3.2.3) два возможных варианта, подтвердивших бы нашу гипотезу:

- Резкое возрастание ошибок к концу графика, свидетельствующее о том, что уже произошедшие ошибки провоцируют новые, что в итоге выливается в краш;

- Выход графиков на асимптоту, свидетельствующий о том что когда произошло уже достаточно ошибок в более ранний период, каждая следующая ошибка увеличивает вероятность краша значительно существеннее.

Однако, наша гипотеза не подтвердилась и вместо этого мы увидели ступенчатые функции на всем протяжении времени. Таким образом мы пришли к следующим выводам:

- Анализировать следует не количество, а процентное содержание плохих сообщений;

- Следует подробнее посмотреть на эти серии сообщений, являющиеся на графике (рис. 3.2.3) вертикальными скачками.

Составив графики процентного содержания "плохих" сообщений в промежутке времени, мы сразу же смогли увидеть в них наличие выделяющихся пиков (рис. 3.2.3).

Посмотрев на конкретные участки логов, содержащие серии плохих сообщений, мы увидели, что часто сообщения идут об одинаковых ошибках. И появляются они подряд друг за другом. Это натолкнуло нас на мысль ввести в вектор показатели расстояния (как стекового, так и временного) и однородности. Успешность составления нашего вектора X хорошо иллюстрируют диаграммы размаха (boxplots)(рис. 3.2.3).

3.2.4. Разбиение логов на блоки

Актуальным был вопрос выбора значения параметра разбиения T. Для решения этой задачи была проведена серия экспериментов:

- Были составлены вектора X для T = 15 минут, 30 минут, 1 час, 2 часа, 4 часа, 8 часов;

- Каждый набор был случайно перетасован и разделён в стандартных пропорциях 80 к 20;

- На 80 процентах мы обучали алгоритм классификатора Random Forest, на 20 оценивали эффективность предсказания (TNR, TPR)

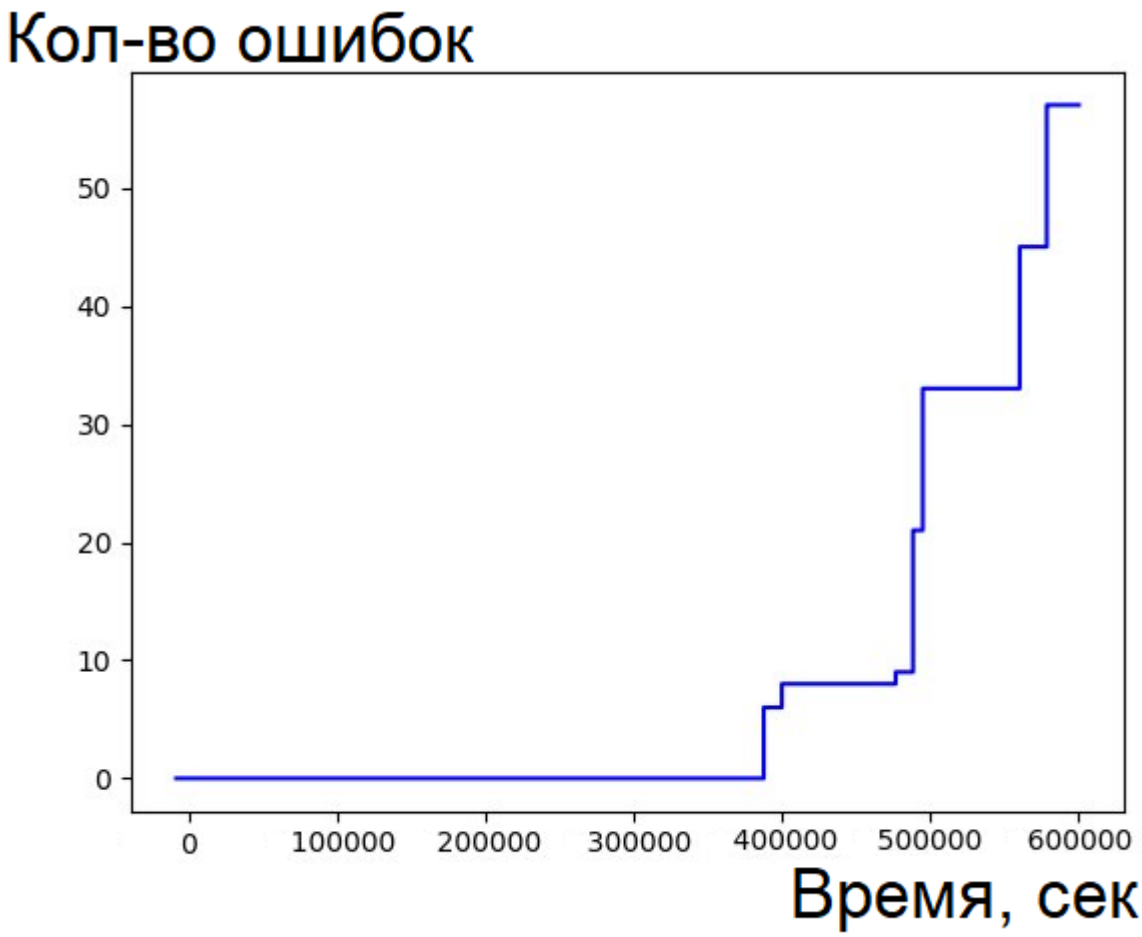

Рис. 2: Пример графика зависимости накопления количества ошибок от времени

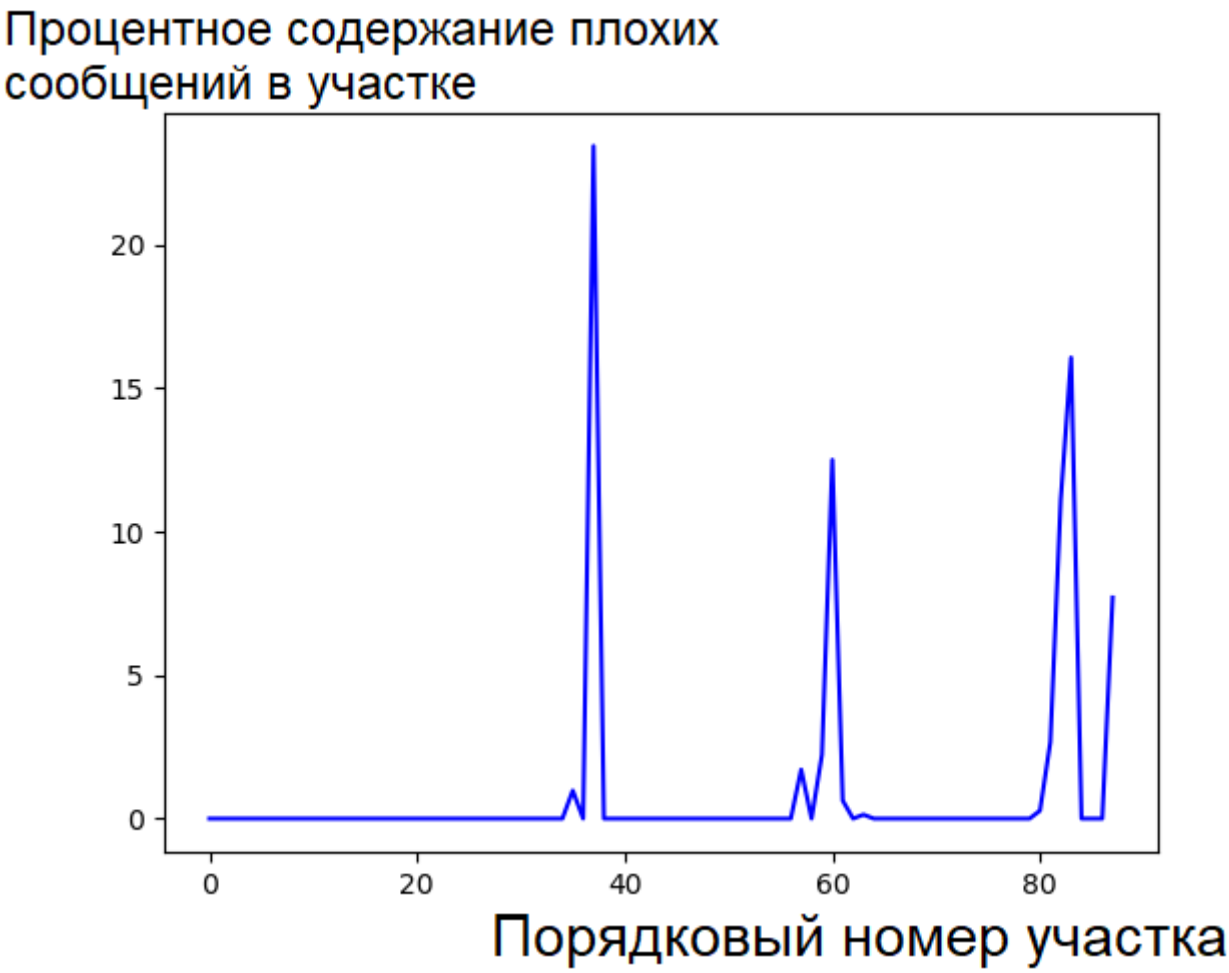

Рис. 3: Пример графика содержания ”плохих” сообщений в участке

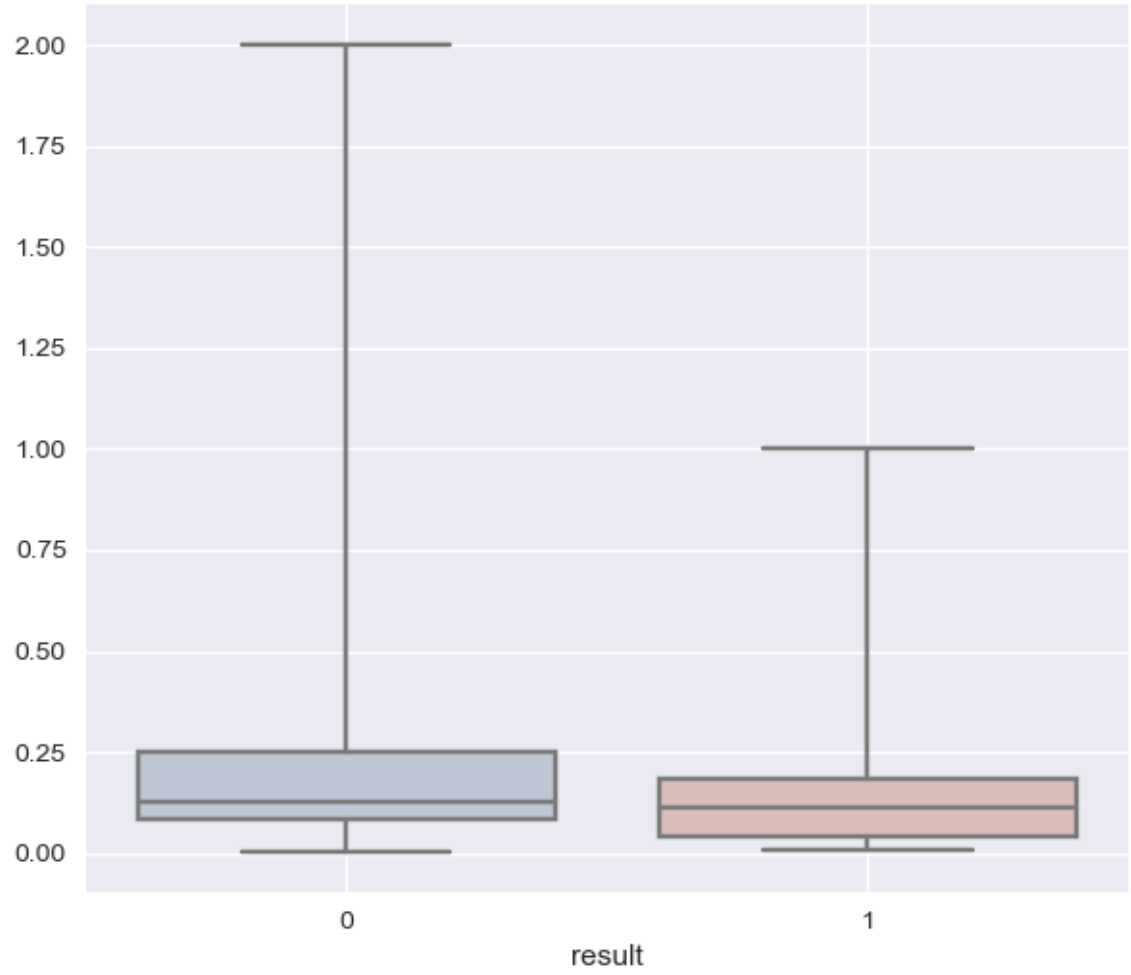

Рис. 4: Пример различия между значениями признака однородности хороших сообщений в участке для хороших (0) и плохих (1) векторов для T=2 часа. Значение 2 присуждается, если в участке хороших сообщений нет

T	15 минут	30 минут	1 час	2 часа	4 часа	8 часов
TPR	0,4	0,5	0,7	0,95	0,8	0,75
TNR	0,63	0,64	0,68	0,73	0,71	0,69

Рис. 5: Таблица 1. Оценка эффективности алгоритма предсказания в зависимости от выбранного параметра T

- Мы повторяли пункты 2-3 10 раз и выбирали из них лучший результат, который заносили в таблицу (рис. 3.2.4).

По результатам эксперимента был выбран параметр T=2 часа Также в процессе этого экспериментов мы заметили, что наилучший и наихудший результаты сильно разнятся, что навело нас на мысль о существовании в выборке некоторых ключевых “хороших” векторов и возможности уменьшения обучающей выборки за счет их нахождения

3.2.5. Разбиение наборов данных

Ввиду огромного неравенства классов, нам нет необходимости обучать модель на 80% хороших векторов, так как это дает улучшение всего в 1-2% в TPR по сравнению с обучением на 50%, содержащих некоторые ключевые вектора.

Для поиска этих векторов мы использовали метод половинного деления и смотрели, как каждая половинка показывает себя в классификации. По итогам был отобран некоторый набор “хороших” векторов, который вошел в обучающую выборку.

Для построения вероятностных моделей мы разделили наш набор подобным образом:

- 50% хороших векторов в обучающую выборку 50% хороших векторов в тестирующую;
- 80% плохих векторов в обучающую выборку, 20% в тестирующую.

3.3. Эксперименты

3.3.1. Random Forest

Алгоритм классификации показал следующие результаты:

- TNR = 0.95
- TPR = 0.7

3.3.2. XGBoost

Алгоритм классификации показал следующие результаты:

- TNR = 0.95
- TPR = 0.72

3.3.3. Isolation Forest

Алгоритм поиска аномалий показал следующие результаты:

- TNR = 0.62
- TPR = 0.87

3.4. Оценка результатов

Алгоритм поиска аномалий показывает значительно лучший TPR, однако, недостаточно качественно ищет “плохие участки”. Алгоритм XGBoost не даёт существенного улучшения результатов по сравнению с Random Forest, но является более тяжелым, чем Random forest. В качестве алгоритма решения нашей задачи был выбран Random Forest, который хорошо показал себя на поиске плохих участков и достаточно хорошо на отсеивании хороших участков.

4. Заключение

В рамках данной работы были выполнены следующие задачи:

- Изучены существующие методы машинного обучения на базе решающих деревьев;
- Выработана методика анализа сообщений в логах;
- Подготовлена обучающая выборка;
- Проведены эксперименты

  Получен положительный результат.

Список литературы

[1] Chandola V.; Banerjee A.; Kumar V. Anomaly detection: A survey // article. — 2009. — URL: http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf (online; accessed: 14.05.2018).

[2] Chen Tianqi Guestrin Carlos. XGBoost: A Scalable Tree Boosting System // article. — 2016. — URL: https://arxiv.org/abs/1603.02754 (online; accessed: 14.05.2018).

[3] Goldberg David. Statistical Anomaly Detection // article. — 2015. — URL: https://www.ebayinc.com/stories/blogs/tech/statistical-anomaly-detection (online; accessed: 14.05.2018).

[4] Guthemberg Silvestre Carla Sauvanaud Mohamed Kaâniche Karama Kanoun. An anomaly detection approach for scale-out storage systems // article. — 2014. — URL: https://hal.archives-ouvertes.fr/hal-01076212 (online; accessed: 14.05.2018).

[5] Ira Cohen Moises Goldszmidt Terence Kelly Julie Symons. Correlating instrumentation data to system states: A building block for automated diagnosis and control // article. — 2004. — URL: https://www.usenix.org/legacy/event/osdi04/tech/full_papers/cohen/cohen.pdf (online; accessed: 14.05.2018).

[6] Klein Andrew. What Can We Learn From 100,000 Spinning Hard Drives? // SDC 2017. — 2017. — URL: https://www.youtube.com/watch?v=A5pm6VGMkn8 (online; accessed: 14.05.2018).

[7] L. Breiman. Random forests. Machine Learning. — 2001. — Vol. 45.

[8] Liu F. T. Ting K. M. Zhou Z. H. Isolation Forest // article. — 2008. — URL: https://arxiv.org/abs/1603.02754 (online; accessed: 14.05.2018).

[9] Owen Vallis Jordan Hochenbaum Arun Kejariwal. A Novel Technique for Long-Term Anomaly Detection in the Cloud // Article. — 2014. — URL: https://www.usenix.org/system/files/conference/hotcloud14/hotcloud14-vallis.pdf (online; accessed: 14.05.2018).

[10] Shivkumar Shivaji Ryan Sudhakaran Yong Bum Lee Joel Barajas Zamora. Storage anomaly detection // article. — 2018. — URL: http://www.freshpatents.com/-dt20180201ptan20180032385.php (online; accessed: 14.05.2018).