Санкт-Петербургский государственный университет
Математическое обеспечение и администрирование информационных систем

Кафедра системного программирования

Нагаев Артур Рафкатович

Обнаружение изменений в потоке данных с непрерывным распределением

Выпускная квалификационная работа

Научный руководитель:
к.т.н., доц. Литвинов Ю. В.

Рецензент:
Руководитель команды аналитики ООО “Интеллиджей Лабс” Поваров Н. И.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Department of Software Engineering

Arthur Nagaev

Detection of changes in continuously distributed streaming data

Graduation Thesis

Scientific supervisor:
C.Sc., Associate Professor Yurii Litvinov

Reviewer:
Head of Analytics, IntelliJ Labs Co. Ltd. Nikita Povarov

Saint-Petersburg
2019

Оглавление

Введение

Потоковые данные (streaming data) – данные, непрерывно поступающие из большого количества источников, которые одновременно формируют информацию и отправляют её небольшими объемами. Поток может состоять из данных различного вида, например, пользовательской информации в социальных сетях или интернет-магазинах, информации с геопространственных сервисов, финансовых торговых площадок и других источников. Обработка потоковых данных является предпочтительной для большинства сценариев использования, подразумевающих непрерывное формирование новых динамических данных. Также она применяется в большинстве случаев использования, подразумевающих обработку больших данных, анализ которых позволяет компаниям разобраться во многих аспектах своей деятельности. Например, в активности серверов, навигации по веб-сайтам, геолокации устройств, людей или товаров. Различные организации используют машинное обучение для извлечения из данных различных закономерностей. Стандартные модели машинного обучения предсказывают информацию об окружающем мире на основе набора данных, именуемого обучающей выборкой. В связи с изменчивостью анализируемых данных, информация, предсказываемая на основе устаревшего набора данных, может совершенно неверно отражать природу новых данных. Задача обнаружения изменения в потоке данных на основе анализа работы модели машинного обучения называется задачей обнаружения изменения концепции (concept drift detection). Данная задача встречается, во многих сферах. В задачах мониторинга сетей изменения в моделях трафика могут указывать на вторжения, атаки или аномалии, которые требуют внимания сетевых администраторов [8, 5]. В задачах классификации тональности текста в связи с изменчивостью слов, используемых для выражения позитивных и негативных настроений, существует необходимость обновления моделей [2]. Автономные транспортные средства должны быстро реагировать на различные изменения окружающей среды, например, погодные условия, освещение, вид дорожного покрытия

и другие [16]. Не смотря на то, что данная область является хорошо изученной [18], в контексте обработки потоковых данных в распределенных системах возникают дополнительные вопросы. Различные части потока данных могут обрабатываться независимыми вычислительными узлами системы. Для применения известных подходов в таких системах необходимо направлять все данные на один вычислительный узел, что сильно уменьшает пропускную способность и приводит к различным проблемам. Предметом исследования данной работы является распределенное обнаружение изменений концепции в потоке данных.

1. Постановка задачи

Целью данной работы является исследование распределенного подхода к обнаружению изменений концепции в потоке данных. Для достижения этой цели были поставлены следующие задачи.

- Изучить существующие технологии обнаружения изменений концепции в потоке данных.
- Реализовать фреймворк для обработки и обнаружения изменений в потоковых данных.
- Провести эксперименты, сравнить распределенный и нераспределенный случаи и провести анализ результатов.

2. Обзор литературы

Введем основные обозначения. Под X_i^j будем подразумевать поток данных, приходящих с момента времени i по момент времени j. Момент изменения концепции в потоке данных, характеризующийся изменением распределения F_0 исходных данных на неизвестное распределение F_1, обозначим за w. Задача машинного обучения заключается в поиске решающей функции $y : X \to Y$, сопоставляющей объектам исследования $x_i \in X$ значения $y_i \in Y$ из множества допустимых ответов на основе имеющихся описаний прецедентов, именуемых обучающей выборкой $\{(x_i, y_i)\}_{i=1}^{l}$. Процесс поиска данной функции заключается в выборе модели восстанавливаемой зависимости и подборе её оптимальных параметров с помощью выбранного алгоритма обучения, на которых достигается оптимальное значение заданного функционала качества, значения которого показывают, насколько хорошо модель описывает наблюдаемые данные. Задачи машинного обучения на потоковых данных часто называют задачами онлайн-обучения.

Изменения в данных могут происходить по-разному. Они могут происходить постепенно или резко, а также появляться периодически. Для каждого случая применяются различные методы. Например, повторяющиеся каждый сезон изменения могут быть определены явным образом и обработаны заранее. Однако в задачах прогнозирования изменения могут иметь более случайный характер.

Расмотрим основные методы обработки изменений концепции.

1. Тривиальным является случай, при котором отсутствует обнаружение изменений. Используются предобученные модели.

2. Периодическое повторное обучение модели. В данном случае модели повторно обучаются через определенное количество времени на новых данных.

3. Использование моделей, которые можно итеративно обучать на новых данных.

4. Использование экспоненциального сглаживания весов позволяет итеративно обучаемым моделям уделять большее внимание новым данным.

5. Использование ансамбля моделей, при котором несколько моделей учатся исправлять предсказания статичных моделей.

6. Использование статистических методов определения изменения. Принятие гипотезы об изменении сигнализирует о необходимости повторного обучения модели на новых данных.

Специфика обработки большого потока данных заключается в постоянном анализе качества работы модели в режиме реального времени. Резкое ухудшение качества может сигнализировать об изменении распределения потока поступающих данных. Определение резкого изменения оптимально осуществлять с помощью статистических методов последовательной проверки гипотез [18]. Рассмотрим некоторые из них.

Скользящие окна (sliding windows) [17, 12, 14]. Данные методы характеризуются использованием массивов данных (окон), хранящих информацию о потоке, например качество модели. Обычно используется одно окно, суммирующее прошлую информацию и скользящее окно, хранящее последние пришедшие элементы потока. Распределения по этим двум окнам сравниваются с использованием статистических тестов с нулевой гипотезой о том, что распределения равны. Если нулевая гипотеза отклоняется, изменение объявляется в начале окна обнаружения. Основной недостаток данных методов заключается в выборе размера окон [6]. Окна с фиксированным размером работают лишь в редких случаях, когда известно, с какой скоростью происходят изменения. Большой размер окна ведет к задержкам в определении изменения, связанным с необходимостью его заполнения, в то время, как небольшой размер окна может привести к пропуску обнаружения изменения или к позднему обнаружению, если изменения постепенно накапливаются со временем. Возможным компромиссным решением данной проблемы является параллельное использование окон различного размера, что влечет дополнительные затраты времени и памяти на анализ.

[12, 17]. Использование окон с затухающими весами (decaying-weighted) является альтернативным подходом к решению данной проблемы [13]. Каждому элементу потока сопоставляется убывающий с его возрастом вес, зависящий от заранее заданной постоянной затухания. В данном случае выбор постоянной должен соответствовать скорости изменения потока, которая зачастую неизвестна.

Последовательный критерий отношения вероятностей (Sequential Probability Ratio Test, SPRT) – классический метод определения изменений в потоке данных, использующийся в большом количестве алгоритмов [7]. В момент изменения потока данных, когда исходное распределение P_0 меняется на распределение P_1, вероятность того, что наблюдаемые примеры распределятся по P_1 ($P(x_w...x_n|P_1)$) становится значительно выше вероятности их распределения по P_0 ($P(x_w...x_n|P_0)$). Решение об изменении потока принимается в момент, когда отношение данных вероятностей становится не меньше заданного порога L. Предполагая независимость наблюдений X_i, статистика для проверки гипотезы H_1 о том, что изменение произошло в момент времени w, против нулевой гипотезы H_0 определяется следующим образом.

$$T[w,n] = log\frac{P(x_w...x_n|P_1)}{P(x_w...x_n|P_0)} = \sum_{i=w}^{n} log\frac{P_1[x_i]}{P_0[x_i]} = T[w,n-1] + log\frac{P_1[x_n]}{P_0[x_n]}$$

Кумулятивная сумма (CUSUM) – метод последовательного анализа, использующий принципы метода SPRT, часто применяющийся в разных подходах к обнаружению изменений [4]. На каждой итерации теста вычисляется статистика

$$S_n = max(0, S_{n-1} + (X_n - q)), S_0 = 0$$

где q – максимальная величина допустимых изменений X. Так как статистика за прошлую итерацию уже известна, вычисление занимает $O(1)$ времени. Тест сигнализирует об изменении, когда статистика S_n начинает отклоняться вверх от заданного порога L. Для учета отрицательных отклонений статистики S от L нужно использовать функцию min при

её вычислении.

Рассмотрим распределенные методы адаптации к изменению концепции. В статье [1] предлагается использовать метод Online MapReduce Drift Detection Method (OMR-DDM), заключающийся в параллельном использовании моделей (элементы map), которые передают вектора с качеством работы модели на заданном скользящем окне элементу reduce, на котором определяется изменение концепции на основе качества всех моделей с последующей рассылкой информации об изменении во все map элементы, вызывающей обновление их моделей. Однако в статье не описно, какие именно модели использовались в экспериментах. В статье [15] предложен метод Micro-Cluster Nearest Neighbour (MC-NN), являющийся распределенной модификацией метода k ближайших соседей, позволяющей адаптироваться к изменениям концепции. Однако данный метод далеко не всегда наиболее лучшим образом справляется с задачами классификации. Распределенная схема, предложенная в статье, позволяет естественным образом ускорить работу классического метода k-NN за счет распараллеливания.

3. Модель

Для проведения экспериментов на языке Python был реализован фреймворк для моделирования потока и его обработки. На рисунке 1 представлена диаграмма классов.

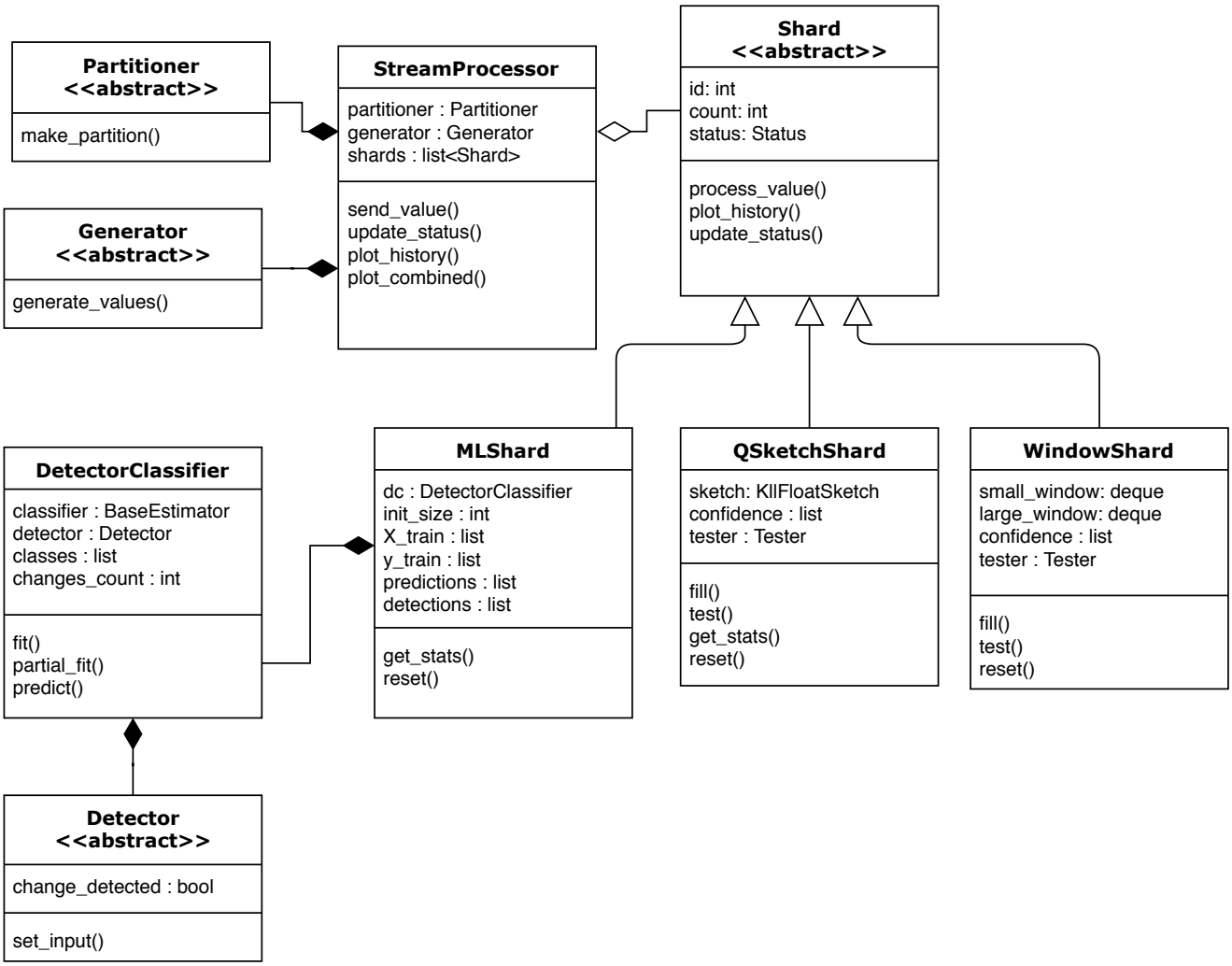

Рис. 1: UML диаграмма классов модели

- **StreamProcessor** является классом, рассылающим данные в обработчики в соответствии с распределением, предоставляемым наследником класса Partitioner.
- **Partitioner** – это абстрактный класс, потомки которого реализуют партицию данных по обработчикам.

- **Generator** – это абстрактный класс, потомки которого реализуют генерацию (считывание), а также передачу данных в StreamProcessor.
- **Shard** – это абстрактный класс, выступающий в роли интерфейса к обработчикам потока. Обработчик должен уметь принимать элемент потока, а также выполнять проверку на изменение в потоке.
- **DetectorClassifier** – это класс, выполняющий проверку на изменение концепции, а также онлайн-обучение модели.
- **Detector** – это абстрактный класс, предназначенный для определения изменения концепции.

Генерация данных. Для генерации данных в наследниках класса Generator была использована библиотека numpy. Для каждого элемента потока генерируется целочисленный идентификатор id для моделирования распределения по обработчикам пользовательских данных по их уникальным идентификаторам.

Партиция данных. Поскольку в реальной ситуации влиять на распределение данных в потоке невозможно, были рассмотрены три вида распределения данных по обработчикам.

1. Round robin – чередование обработчиков.
2. Hashing by id – распределение по значению хэш-функции от id.
3. Hashing by value – распределение по значению хэш-функции от элемента потока.

QSketchShard. Данный вид обработчика обрабатывает поток с помощью структуры данных KllQuantileSketch [11]. Данная структура данных позволяет получить выборочные квантили входящего набора данных с точностью e, с помощью которых можно эффективно создавать выборку, апроксимирующую поток данных. С помощью заданного статистического теста, например, критерия Колмогорова-Смирнова из

библиотеки scipy, оценивается уровень значимости, являющийся вероятностью ошибки первого рода. Гипотеза об изменении потока принимается в том случае, если уровень значимости окажется ниже заданного порога.

WindowShard. Обработчик хранит новые данные в двух скользящих окнах разного размера. С помощью заданного статистического теста из библиотеки scipy сравнивается похожесть окон, либо, аналогично QSketchShard, выполняется проверка статистических гипотез. Флаг об изменении потока выставляется в случае, если гипотеза об изменении была принята для каждого окна. Данный класс был использован для сравнения.

MLShard. Данный обработчик собирает обучающую выборку заданного размера, инициирует обучение модели, а затем отправляет в детектор входящие данные, обновляя информацию с детектора об изменении на каждом шаге.

DetectorClassifier. Данный класс удовлетворяет интерфейсу BaseEstimator из библиотеки scikit-learn. Данному интерфейсу удовлетворяют все модели машинного обучения данной библиотеки, работа с которыми в большей степени заключается в вызове методов fit для обучения и predict для предсказания. Благодаря этому его можно было заменить на обычную модель офлайн-обучения без определения изменения в концепции. Помимо этого класс DetectorClassifier имеет метод partial fit, что позволяет использовать вместо него итеративно обучаемые модели.

Для обнаружения изменения концепции на языке Python были реализованы два детектора: PageHinkley [9] и Adwin [3]. Данные детекторы часто упоминаются в исследованиях, а также достаточно просты для реализации.

PageHinkley – тест, позволяющий эффективно вычислять статистику S используя предыдущие вычисления.

$$M_n = \frac{x + M_{n-1} \cdot (n-1)}{n} \;\; S_n = S_{n-1} \cdot \alpha + (x - M_n - \delta),$$

где M_n – среднее, а S_n – статистика, характеризующая отклонение от среднего. Гипотеза об изменении потока принимается, когда данная статистика значительно отклоняется от заданного порога, $S > \lambda$.

Адаптивное окно (ADWIN) – метод, основанный на использовании скользящего окна, которое расширяется, если изменения маловероятны и сужается, если происходит изменение распределения данных. Входными данными для алгоритма являются последовательность чисел $x \in [0, 1]$ и уровень доверия γ. Обозначим средние арифметическое окна W_j за $\hat{\mu}W_j$, порог $e = \sqrt{\frac{1}{m} \cdot ln\frac{4|W|}{\gamma}}$, где m – гармоническое среднее окон W_0 и W_1.

Algorithm 1 ADWIN

```
1: Инициализировать окно W
2: for each t > 0 do
3:     W ← W ∪ x_t
4:     repeat
5:         Исключить элемент из W
6:     until |μ̂W_0 − μ̂W_1| < e для каждого разделения W на W_0 ∪ W_1
7: end for
       return μ̂W
```

4. Эксперименты

Для эксперимента был использован реальный набор данных electricity. Набор данных состоит из сорока пяти тысяч записей, каждая из которых имеет 6 атрибутов и 2 класса, характеризующие возрастание и падение цены на электричество в штате Новый Южный Уэльс в Австралии [10]. Изменения концепции данных вызваны изменениями в привычках потребления электроэнергии, событиях и временах года.

В качестве модели для онлайн-обучения был выбран наивный Байесовский классификатор GaussianNB из scikit-learn в связи с тем, что он является широко используемой моделью, демонстрирующей неплохие результаты [15]. Для того, чтобы проверить работу детекторов, был также выбран линейный классификатор SGDClassifier из scikit-learn.

Так как потоковые данные нельзя разделить на тестовую и обучающую выборку, а также в связи с тем, что набор данных является сбалансированным, то есть содержит экземпляры разных классов в приблизительно равной пропорции, в качестве метрики оценивания работы использовалась метрика accuracy для скользящего окна размера w, равная отношению числа верно классифицированных объектов в окне к длине окна w. Сравнивались детекторы по усреднению данной метрики на всем потоке. На рисунке 2 изображена данная метрика для трех детекторов с классификатором GaussianNB для окон с различными размерами. Синим выделен пустой детектор, который не обнаруживает изменений. Оранжевым выделен детектор PageHinkley. Зеленым выделен детектор ADWIN.

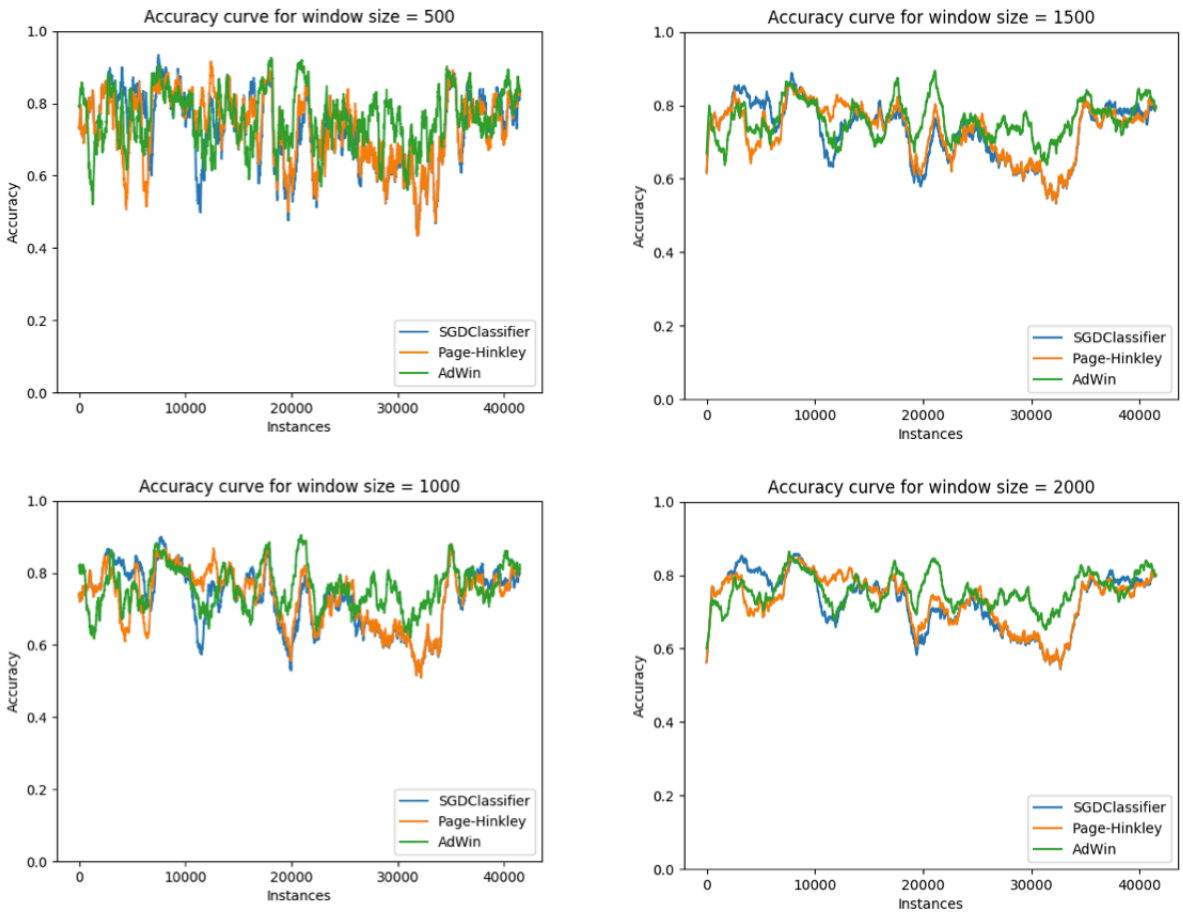

Рис. 2: Значения accuracy для различных окон

В таблице 1 приведено сравнение результатов работы моделей для реализованных детекторов. Под mean accuracy понимается среднее значение accuracy для окна шириной 1000.

Таблица 1: Сравнение детекторов

Модель	Детектор	Mean accuracy	Число изменений
GaussianNB	None	0.718	0
GaussianNB	PageHinkley	0.736	25
GaussianNB	ADWIN	0.75	139
SGDClassifier	None	0.84	0
SGDClassifier	PageHinkley	0.844	2
SGDClassifier	ADWIN	0.846	14

В таблице 2 приведено сравнение результатов работы детектора ADWIN с моделью SGDClassifier для различного числа обработчиков. Эксперимент проводился на Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz.

Таблица 2: Сравнение результатов для различного числа обработчиков

Число обработчиков	1	2	3	4	5	6	7	8
Mean accuracy	0.846	0.844	0.841	0.839	0.837	0.834	0.832	0.83
Время (сек)	15.97	8.34	6.61	5.23	4.29	3.87	3.66	3.43

Вывод

- Повторное обучение модели в местах изменения концепции позволяет улучшить её качество.
- Не смотря на то, что качество моделей, используемых параллельно, уменьшается с ростом числа обработчиков, можно значительно сократить время обработки благодаря естественной их масштабируемости.

5. Заключение

В ходе работы были выполнены следующие задачи.

- Изучены существующие технологии обнаружения изменений концепции в потоке данных.
- Реализован фреймворк для обработки и обнаружения изменений потоковых данных.
- Проведены эксперименты, проведено сравнение распределенного и нераспределенного случаев, а также проведен анализ результатов.

Список литературы

[1] Andrzejak Artur, Gomes Joao Bartolo. Parallel concept drift detection with online map-reduce // 2012 IEEE 12th International Conference on Data Mining Workshops / IEEE. — 2012. — P. 402–407.

[2] Bifet Albert, Frank Eibe. Sentiment knowledge discovery in twitter streaming data // International conference on discovery science / Springer. — 2010. — P. 1–15.

[3] Bifet Albert, Gavalda Ricard. Learning from time-changing data with adaptive windowing // Proceedings of the 2007 SIAM international conference on data mining / SIAM. — 2007. — P. 443–448.

[4] Chetouani Yahya. Use of cumulative sum (CUSUM) test for detecting abrupt changes in the process dynamics // International Journal of Reliability, Quality and Safety Engineering. — 2007. — Vol. 14, no. 01. — P. 65–80.

[5] Cormode Graham, Muthukrishnan Shan. What's new: Finding significant differences in network data streams // IEEE/ACM Transactions on Networking (TON). — 2005. — Vol. 13, no. 6. — P. 1219–1232.

[6] Dries Anton, Rückert Ulrich. Adaptive concept drift detection // Statistical Analysis and Data Mining: The ASA Data Science Journal. — 2009. — Vol. 2, no. 5-6. — P. 311–327.

[7] Exponentially weighted moving average charts for detecting concept drift / Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, David J Hand // Pattern recognition letters. — 2012. — Vol. 33, no. 2. — P. 191–198.

[8] Fast portscan detection using sequential hypothesis testing / Jaeyeon Jung, Vern Paxson, Arthur W Berger, Hari Balakrishnan // IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004 / IEEE. — 2004. — P. 211–225.

[9] Gama João, Sebastião Raquel, Rodrigues Pedro Pereira. On evaluating stream learning algorithms // Machine learning. — 2013. — Vol. 90, no. 3. — P. 317–346.

[10] Harries Michael, Wales New South. Splice-2 comparative evaluation: Electricity pricing. — 1999.

[11] Karnin Zohar, Lang Kevin, Liberty Edo. Optimal quantile approximation in streams // 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS) / IEEE. — 2016. — P. 71–78.

[12] Kifer Daniel, Ben-David Shai, Gehrke Johannes. Detecting change in data streams // Proceedings of the Thirtieth international conference on Very large data bases-Volume 30 / VLDB Endowment. — 2004. — P. 180–191.

[13] Maintaining stream statistics over sliding windows / Mayur Datar, Aristides Gionis, Piotr Indyk, Rajeev Motwani // SIAM journal on computing. — 2002. — Vol. 31, no. 6. — P. 1794–1813.

[14] Muthukrishnan Shanmugavelayutham et al. Data streams: Algorithms and applications // Foundations and Trends® in Theoretical Computer Science. — 2005. — Vol. 1, no. 2. — P. 117–236.

[15] Scalable real-time classification of data streams with concept drift / Mark Tennant, Frederic Stahl, Omer Rana, João Bártolo Gomes // Future Generation Computer Systems. — 2017. — Vol. 75. — P. 187–199.

[16] Stanley: The robot that won the DARPA Grand Challenge / Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp et al. // Journal of field Robotics. — 2006. — Vol. 23, no. 9. — P. 661–692.

[17] An information-theoretic approach to detecting changes in multi-dimensional data streams / Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, Ke Yi // In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications / Citeseer. — 2006.

[18] A survey on concept drift adaptation / João Gama, Indrė Žliobaitė, Albert Bifet et al. // ACM computing surveys (CSUR). — 2014. — Vol. 46, no. 4. — P. 44.