# МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
 высшего образования   
**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ПОТОКОВЫХ ДАННЫХ В БЕСПРОВОДНЫХ СЕНСОРНЫХ СЕТЯХ С ПОМОЩЬЮ ДЕРЕВЬЕВ РЕШЕНИЙ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Барулина

(подпись, дата)

Факультет компьютерных технологий и прикладной математики курс 3

Направление 02.03.02 – «Фундаментальная информатика и  
информационные технологии»

Научный руководитель доц.,   
канд.тех. наук, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Приходько

(подпись, дата)

Нормоконтролер преп.,  
канд.тех. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.Е. Полупанова

(подпись, дата)

Краснодар 2017

СОДЕРЖАНИЕ

[Введение 3](#_Toc480398217)

[1. Обработка потоковой информации в WSN 4](#_Toc480398218)

[1.1 Потоковые данные 4](#_Toc480398219)

[1.2 Основные проблемы обработки и анализа потоковых данных 4](#_Toc480398220)

[1.3 Интеллектуальный анализ данных в беспроводных сенсорных сетях 5](#_Toc480398221)

[2 Передовые методы потокового машинного обучения для деревьев решений 8](#_Toc480398222)

[2.1 Сложности потокового машинного обучения 8](#_Toc480398223)

[2.2 Дерево Хеффдинга 9](#_Toc480398224)

[2.3 Задача классификации потоковых данных с помощью дерева Хеффдинга 10](#_Toc480398225)

[3 Алгоритм VFDT для потоковых данных 13](#_Toc480398226)

[3.1 Алгоритм VFDT на основе метода Хеффдинга 13](#_Toc480398227)

[3.2 Дополнительные возможности алгортитма VFDT. 15](#_Toc480398228)

[4. Реализация и тестирование алгоритмов 17](#_Toc480398229)

[4.1. Обоснование выбора языка программирования 17](#_Toc480398230)

[4.2. Тестирование и оценка результатов 18](#_Toc480398231)

[Заключение 29](#_Toc480398232)

[Список используемой литературы 30](#_Toc480398233)

# ВВЕДЕНИЕ

В последнее десятилетие наблюдается большой интерес научного мира к теме обработки больших данных. Такой интерес обусловлен лавинообразным увеличением источников больших данных, которые требуют быстрого анализа. Согласно исследованию IDC Digital Universe, в ближайшие пять лет объем данных на планете вырастет до 40 зеттабайтов, то есть к 2020 году на каждого живущего на Земле человека будет приходиться по 5200 Гб. Источниками данных на сегодня являются люди, промышленность и наука, причем большая часть данных поступает сплошным потоком, который должен быть сразу же обработан, чтобы, во-первых, не занимать хранилища неэффективными дынными, а во-вторых получать новые знания быстро, т к. этого требует современный ритм жизни.

Среди источников потоковых данных можно выделить on-line видеокамеры, новостные сайты, результаты научных исследований в области химии, биомедицины, атомной физики, а также популярные в различных прикладных областях беспроводные сенсорные сети (WSN).

Предполагается, что беспроводные сенсорные сети позволят современным технологиям применяться в настоящем и будущем, начиная от отслеживания, мониторинга и систем наблюдения, до множества других технологий, которые могут улучшить аспекты повседневной жизни. WSNs предлагают недорогой способ сбора данных в распределенной среде, которые могут быть труднополучаемы в природе, из таких мест как области биохимического заражения, сейсмически опасных зон, а также областей с труднодоступными рельефами местности или боевых зон. Датчики WSNs—это миниатюрные встроенные вычислительные устройства, которые продуцируют большие объемы потоковых данных, полученных от своего окружения в течение своего жизненного цикла. В этой прикладной области актуальным является развитие таких средств анализа данных как деревья решений, в том числе изучаются возможные способы их распараллеливания.

# 1. Обработка потоковой информации в WSN

# 1.1 Потоковые данные

Потоковые данные – это данные, формируемые непрерывно большим количеством источников, которые в свою очередь отправляют записи данных одновременно и в небольшом объеме. В состав потоковых данных входят различные виды данных, например:

1. информация о покупках в интернет-магазине;
2. телеметрические данные;
3. температура в океане;
4. информация из социальных сетей;
5. выписки звонков с мобильных телефонов;
6. результаты научных исследований, поступающие в режиме реального времени.

Эти данные очередь должны пройти предварительную обработку, (фильтрацию, очистку, структуризацию) после чего их можно использовать в различных аналитических задачах. Информация, полученная в результате анализа данных, позволяет коммерческим фирмам, производственникам и ученым разобраться во многих аспектах своей деятельности.

# 1.2 Основные проблемы обработки и анализа потоковых данных

При потоковой передаче данных возникают дополнительные условия, усложняющие их обработку:

* непрерывное формирование новых динамических данных;
* наличие пропущенных данных;
* наличие шумов, которые необходимо отсеивать на этапе поступления данных.

Обработка потоковых данных требует использования двух уровней: хранилища и обработки. Уровень хранилища должен поддерживать очередность записей и строгую непротиворечивость для обеспечения быстрых, экономичных и воспроизводимых операций записи и чтения больших потоков данных. Уровень обработки в свою очередь отвечает за использование данных, расположенных на уровне хранилища, выполнение вычислений с использованием этих данных и уведомления уровня Интеллектуальный анализ данных в беспроводных сенсорных сетях хранилища о том, какие данные можно удалить. Кроме того, необходимо предусмотреть масштабируемость, надежность данных и отказоустойчивость на любом уровне.

# 1.3 Интеллектуальный анализ данных в беспроводных сенсорных сетях

Интеллектуальный анализ данных WSNs ограничивается определенными характеристиками WSNs. Это в первую очередь зависит от топологии соединения узлов-сенсоров в WSN и целей такого соединения. Обычно рассматриваются три основных топологии: (1) в топологии "звезда", где центральное узел подключен ряд окрестных датчиков; (2) в топологии кластера (иерархическая звезда), где разные звезды объединяются центральным узлом и сеть может быть расширена путем добавления дополнительно таких звезд (как кластер кластеров); (3) ячеистая топология (mesh topology), известная как специальная топология, в которой узлы и датчики произвольно добавляются и смешиваются без какой-либо конкретной модели. В литературе по беспроводным сенсорным сетям, центральный узел, известный как сток (sink) - это сетевой компонент, который собирает показания всех датчиков измерений. «Сток» обычно имеет больше вычислительных ресурсов, чем рядовые сенсорные узлы [1]. На рис. 1(а) представлен простой тип “локальная звезда”, который характеризуется одной функцией приемника. Приемник служит в качестве шлюза, где собираются все данные, агрегируются и над ними выполняется интеллектуальный анализ. Результатом является набор классифицированных данных, основанных на измерениях, собранных со всех непосредственно подключенных датчиков. В этом типе WSNs весь интеллектуальный анализ данных в «стоке».

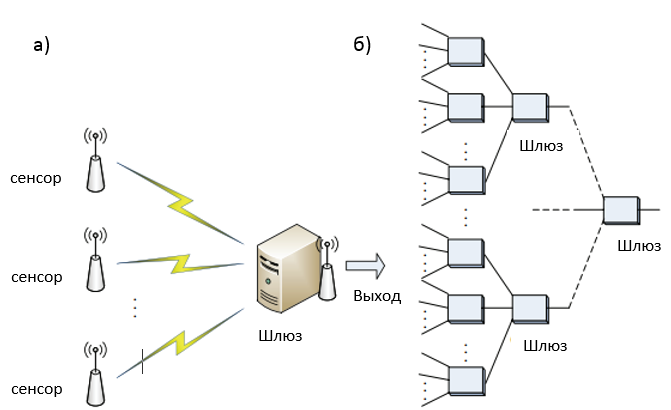


Рисунок 1 - а) Топология "локальной звезды, б) топология "слияния" (фьюжн-типа) в беспроводных сенсорных сетях.

Другой тип «стока» показан на рис. 1(б), напоминает иерархию кластеров и известен как гибридный тип «стока» или топология "слияния" (фьюжн-типа).

Объединение данных часто осуществляется на каждом промежуточном шлюзе; способ голосования используется для выбора наилучшего результата классификации с высоким уровнем точности.

Кроме того, результат анализа данных от каждого промежуточного шлюза служит локальный оптимум или ответом, представляющим собственную ветвь кластеров; результат затем будет подаваться на вход в другой подчиненный кластер. С точки зрения WSNs, интеллектуальный анализ данных проводится в корне сети фьюжн-типа, где каждый вход взят от каждого подключенного кластера, который предлагает репрезентативный выход.

Данная работа сконцентрирована на важной задаче WSN - классификации. Это применимо практически ко всем видам приложений распределенных сенсорных сетей, например, при мониторинге биомедицинских данных пациента – определение того, страдает ли пациент от болезни; при отслеживании стада коров: идет ли оно по обычному маршруту; сбалансирован ли прирост деревьев в тропическом лесу; или при выяснении наличия аномалии любого рода, возникшей в любом другом типе среды. Классификация с помощью деревьев решений делает предсказания или классификации по данным предопределенных классов на основе тестовых наборов путем обхода дерева возможных решений. В WSNs часто приемлем метод дерева решений, потому что деревья, которые представляют взаимосвязи между атрибутами и классами информативны и интуитивно понятны. Каждый путь через дерево решений представляет собой последовательность условий, которые описывают класс. Правила могут быть производными от таких путей решения в дереве, и могут использоваться в WSNs для того чтобы описать результат или явление на основе измерений, полученных от сенсоров. Простота дерева решений дает полезные выводы, базируясь на прозрачной модели процесса обучения. Модель обучается, соблюдая полный набор обучающих выборок. Каждый образец имеет несколько атрибутов, каждый из которых может быть представлен сигналом соответствующего сенсора. Образец записи может иметь вид (х, Y), где X-это вектор (х1, х2,…хn) с N атрибутами, а Y - это класс, где проблема классификации состоит в построении модели, которая задает отображение функция F:{х}⇒{y}. Для сравнения, альтернативные методы классификации, такие как нейронные сети, машина опорных векторов, и регрессионная модель являются трудно интерпретируемыми [2].

# 2 Передовые методы потокового машинного обучения для деревьев решений

# 2.1 Сложности потокового машинного обучения

Потоковое машинное обучение может быть интерпретировано как выполнение машинного обучения в условиях потоковой передачи со следующими характеристиками:

* высокий объем и скорость передачи данных, например, журналы транзакций в банкоматах и ​​операциях с кредитными картами, журнал вызовов в телекоммуникационной компании и данные в социальных сетях, поток информации с узлов беспроводных сенсорных сетей;
* неограниченный объем данных - данные все время поступают в нашу систему, и мы не можем разместить их в памяти или на диске для дальнейшего анализа. Это подразумевает, что мы ограничены однократным анализом данных и имеем мало шансов вернуться к данным для повторной проверки.

Учитывая эти характеристики, обычные алгоритмы машинного обучения (требующие, чтобы все данные были доступны в памяти) не подходят для этого. Таким образом, требования к потоковому обучению для алгоритмов классификации должны придерживаться следующих четырех требований:

1. однократная обработка данных в момент их поступления;
2. использование ограниченного количество памяти;
3. ограниченное время для анализа;
4. готовность давать предсказания в любой момент.

Другим аспектом обучения в потоковой машине является обнаружение изменений, то есть, поскольку объем наших входных данных не ограничен, нам нужно иметь механизм для обработки и реагирования на изменения в характеристиках входящих данных. В классификации этот аспект дает такие вопросы, как

* Следует ли изменить наш классификатор для подстройки к изменениям в характеристиках данных?
* Какие модификации мы должны выполнить?

# 2.2 Дерево Хеффдинга

Индукционное потоковое дерево решений называется Деревом Хёффинга. Название происходит от границы Хёффдинга (Hoeffding bound), которая используется в индукции дерева. Основная идея заключается в том, что Hoeffding bound дает определенный уровень доверия лучшему атрибуту для разделения дерева, поэтому мы можем построить модель на основе определенного количества экземпляров, которые мы видели.

Преодоление недостатков существующих методов и решение задачи высокоскоростной непрерывной обработки потока данных позволит получить больше информации и улучшить анализ в непрерывном потоке данных. «Улучшенный анализ» в этом контексте относится к качеству, а также к эффективности анализа (т.е. к быстрому анализу, который способен справиться с входящей скоростью передачи данных)

Основным вкладом в это решение является дерево Хеффдинга, которое является новым методом потокового машинного обучения для деревьев решений, который решает следующие задачи:

1. неопределенность времени обучения. Время обучения в дереве Хеффинга постоянно для каждого примера (экземпляра), и это означает, что дерево Хеффдинга подходит для потоковой передачи данных;
2. результирующие деревья почти идентичны деревьям, построенным обычными пакетными методами.

# 2.3 Задача классификации потоковых данных с помощью дерева Хеффдинга

Классические алгоритмы машинного обучения для деревьев решения могут одновременно сохранить все обучающие данные основную память. Если данные хранятся в постоянной памяти ЭВМ, то алгоритмы машинного обучения могут неоднократно считывать тренировочнык данные с диска последовательно, что непомерно дорого при обучении сложных деревьев.

Цель дерева Хеффдинга: разработка такого обучающего дерева решений, в котором для обучения каждый пример читается только однажды, и используются малые постоянные времени для обработки этих примеров.

Для того, чтобы найти лучший атрибут в узле, возможно будет достаточно рассматривать только небольше подмножество обучающих примеров, которые проходят через этот узел. В данном потоке примеров можно использовать первый в качестве корневого атрибута.

После того, как выбран корневой атрибут, последовательные примеры передаются в соответствующие листья, и используются для выбора следующих атрибутов, и так далее рекурсивно.

Границы Хеффдинга помогают решить вопрос: сколько примеров достаточно для обучения в каждом узле. Или иначе, сколько экземпляров необходимо для достижения определенного уровня доверия.

Дано:

* r -> вещественная случайная величина, с диапазоном R ;
* n -> число независимых наблюдений ;
* \bar{r} -> среднее значение, вычисленное по формуле из n независимых наблюдений.

Связанная с Хеффдингом теория утверждает, что с вероятностью 1-δ, истинное среднее значение переменной не меньше \bar{r}-\epsilon. А также \epsilon определяется по следующей формуле:

, (1)

Следует заметить способность алгоритма давать одни и те же результаты независимо от распределения вероятностей, порождающего наблюдения. Однако число наблюдений, необходимых для достижения определенных значений ***δ*** а также ***∈*** различаются по вероятностным распределениям.

Вот конкретный пример, который поможет нам понять концепцию: предположим, что разница в информации (индекс Gain) между двумя атрибутами A и B (коэффициент усиления информации A больше, чем B) равен 0,3 и \epsilon=0.1. Это означает, что в будущем минимальная разница между коэффициентом усиления A и B будет, по меньшей мере, 0.3-0.1=0.2.

Другими словами, один признак превосходит другие c вероятностью ***1-δ*** , при условии, что наблюдаемая разница в получении информации больше,   
чем e.

Входные параметры алгоритма:

* S -> последовательность примеров;
* X -> множетво дискретных атрибутов;
* G(.) -> функция расщепления;
* δ -> единица минус желаемая вероятность выбора правильного атрибута в любом узле.

Выходные параметры алгортма:

***HT (Hoeffding Tree)*** -> дерево решений.

Пусть G(*Xi*) − эвристическаямера для выбора тестовых атрибутов (т.е. индекс расщепления - Information Gain, Gini Index). *Xa* : атрибут с наиболее высоким значенем значением индекса после перебора ***n*** примеров. *Xb* : атрибут со вторым по счету индексом ращепления после перебора ***n*** примеров.

Дано желаемое значение ***δ***, if  после перебора ***n*** примеров в узле,

Границы Хеффдинга гарантируют правдоподобный результат с вероятностью *1-****δ*** , при условии, что

Этот узел может быть расщеплен с помощью атрибута *Xa*, и последующие примеры будут направлены в новые листья.

Алгоритм выполняется в следующей последовательности:

* Вычисление разницы в информации (индекс Gain) для атрибутов и определяем лучшие два атрибута;
* Считаем пустым (“null”) атрибут, не приводящий к расщеплению узла (Pre-pruning – предварительная обрезка дерева);
* Для каждого узла проверяется условие: ;
* Если условие удовлетворяется, создаем дочрние узлы, как результаты условной вершины;
* Если не удовлетворяется – из потока извлекаются еще примеры и выполняются вычисления пока условие не будет выполнено.

# 3 Алгоритм VFDT для потоковых данных

Очень быстрое дерево решений (VFDT- Very Fast Decision Tree implementation) является пионером индукции потокового дерева решений, и выполняет все необходимые требования для эффективного управления потоками данных. Предыдущие алгоритмы дерева решений потоковой передачи, введенные до VFDT, не имеют этой возможности.

# 3.1 Алгоритм VFDT на основе метода Хеффдинга

Псевдокод индукционного алгоритма VFDT на основе статистического метода Хеффдинга и, так называемых, границ Хеффдинга, которые помогают ограничить количество примеров достаточных для обучения в каждом узле:

1. пусть HT – дерево с одним листом (корень);
2. for all training examples do (для всех тренировочных примеров делать);
3. sort example into leaf *l* using HT (сортировать пример в листе *l*, используя HT);
4. update sufficient statistics in *l* (обновить достаточную статистику   
   в листе *l)*;
5. increment ni, the number of examples seen at *l* (увеличить число примеров (случаев ni), выдимих в листе *l)*;
6. if ni mod nmin=0 and examples seen at *l* not all of same class then (если ni mod nmin=0 и не все случаи, видимые из *l*, принадлежат одинаковому классу, тогда):
7. compute Gl(Xi) for each attribute (для каждого атрибута   
   вычисляем *l*(Xi));
8. let *Xa* be attribute with highest Gl (найти *Xа*, свойство с наибольшим *l)*;
9. let *Xb* be attribute with second-highest Gl (найти *Xb*, свойство со вторым по счету наибольшим *l);*
10. compute Hoeffding bound (вычислить границы Хеффдинга);
11. if Xа != XØ  and (Gl(Xa) - Gl(Xb) > ϵ or < T) then (Если Xа != XØ и (*l*( Xа) - *l*( Xb)) > ε or ε < τ) тогда):
12. replace ***l*** with an internal node that splits on Xa (заменить лист ***l*** разделенным узлом на основе Xа);
13. for all branches of the split do **(**для всех ветвей разделения сделать):
14. add a new leaf with initialized sufficient statistics (добавить новый лист с полученной достаточной статистикой от разделенного узла);
15. end for;
16. end if;
17. end if;
18. end for.

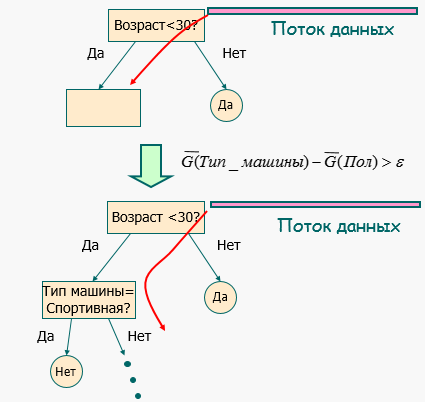


Рисунок 2 - Иллюстарция алгортима VFDT с использованием границ Хеффдинга.

Рассматирваемый алгоритм очень быстрого дерева решений VFDT не имеет условия завершения, так как это потоковый алгоритм. Дерево может расти бесконечно, что противоречит одному из требований к алгоритму потоковой настройки (требует ограниченного объема памяти). Чтобы удовлетворить требованиям ограниченного использования памяти, в алгоритме VFDT вводят метод ограничения узлов. Этот метод вычисляет "предположение" для каждого оучающегося листа ***l***. "Предположение" для активного обучающегося листа в VFDT определяется, как взвешенная верхняя граница уменьшения ошибок, достигаемая за счет сохранения активности листа. Основанный на "предложении" алгоритм может деактивировать листья с низким значением ошибки когда дерево достигнет предела доступной памяти. Однако, VFDT все еще контролирует "предложения" для каждого неактивного листа и может активировать неактивные листья, когда их "предложения" выше, чем "предложения", вычисленные для активных в настоящий момент листьев.

# 3.2 Дополнительные возможности алгортитма VFDT.

Помимо метода ограничения узлов, алгоритм VFDT вводит так же метод устранения слабых атрибутов для уменьшения использования памяти VFDT. Алгоритм удаляет атрибуты, которые окажут очень слабое влияние на результат разбиения на классы, поэтому и статистика, связанная с удаленным атрибутом, может быть удалена из памяти.

Таким образом, алгоритм VFDT в противоположность другим алгоритмам не требует постоянного хранения исходных данных и полного их считывания в процессе обучения, его преимуществом также является мнимизация влияния неполных и "зашумленных" данных на результат. Для этого используется вспомогательный механизм распознавания (auxiliary reconciliation control - ARC). Схема работы VFDT с механизмом ARC показана на рис.3.

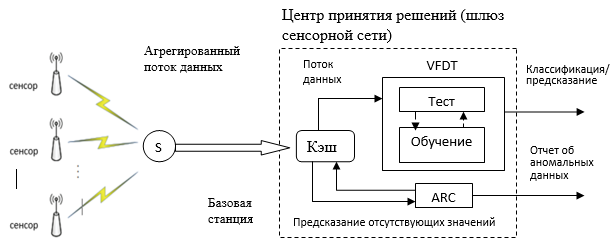


Рисунок 3 - Схема работы VFDT с механизмом ARC

Вспомогательный механизм распознавания (ARC) может быть представлен отдельным программным модулем и работать параллельно и синхронно с обучающим алгоритмом VFDT. Синхронизация обеспечивается использованием "скользящего окна", который позволяет только одному сегменту данных появляться в одном и том же временном интервале. При отсутствии данных оба алгоритма простаивают. Вычислительная скорость "скользящего окна" должна быть не ниже вычислительной скорости VFDT, но выше скорости передачи данных сенсорами WSN. После того, как данные обработаны механизмом ARC, т.е. восстановлены зашумленные и недостающие данные, они передаются далее алгоритму VFDT для обучения и тестирования.

# 4. Реализация и тестирование алгоритмов

# 4.1. Обоснование выбора языка программирования

Для моделирование деревьев решений был выбран язык Java, в котором использовалась специальная билеотека Weka.

Weka (Waikato Environment for Knowledge Analysis) – библеотека алгоритмов машинного обучения для решения задач интеллектуального анализа (data mining). Система позволяет непосредственно применять алгоритмы к выборкам данных, а также вызывать алгоритмы из программ на языке Java. Цель проекта – создать современную среду для разработки методов машинного обучения и применения их к реальным данным. Weka – предоставляет прямой доступ к библиотеке реализованных в ней алгоритмов.

Weka – позволяет выполнять задачи анализа данных таках как:

* подготовка данных(предворительная обработка);
* отбор признаков;
* класстеризация;
* классификация;
* поиск ассоциативных правил;
* регрессивный анализ;
* визуализация результатов.

Weka – набор средств визуализации и бтблиотека алгоритмов машинного обучения для решения задач интеллектуального анализа данных и прогнозирования, с графической пользовательской оболочкой для доступа к ним. Это открытый программный продукт, развиваемый мировым научным сообществом, свободно распространяемый под лицензией GNU GPL. Программное обеспечение написано на Java.

Предполагается, что исходные данные представлены в виде матрицы признаковых описаний объектов. Weka предоставляет доступ к SQL-базам через Java Database Connectivity (JDBC) и в качестве исходных данных может принимать результат SQL-запроса. Возможность обработки множества связанных таблиц не поддерживается, но существуют утилиты для преобразования таких данных в одну таблицу, которую можно загрузить в Weka.

# 4.2. Тестирование и оценка результатов

Предположим, что нас интересует, какую зарплату в среднем получает человек в определенном возрасте, проживающий в определенном городе. Для этого необходимо узнать несколько параметров, таких как:

-город проживания;

- возраст;

-пол;

-образование;

-накопления за год;

-траты в месяц;

-семейное положения;

-сколько дней длится отпуск;

Зная, эти факторы попробуем составит дерево принятия решений. Подготовим данные, стандартом является формат ARFF, фактически это CSV с некоторыми методанными. Далее добавляем поля методанных в начало файла. На отдельных строчках добавляется следующая информация:

- название зависимости @relation имя;

- описание атрибутов @attribute: имя, тип;

- @data перед началом самих данных.

Различают следующие типы данных:

- численные (numeric,real,integer),

-перечислимые(nominal) ,

-строковые (string),

-дата (date[date format]),

-составной тип (relation).

Полученный файл необходимо поместить в систему с помощью Open File вкладки Preprocess. Теперь мы можем редактировать файл уже в системе, изменения могут осуществляться как вручную, так и наложением на данные фильтра для их очищения или трансформации.

Фильтры делятся на два типа:

1. применение, которых может вызвать отклонение (фактически эти фильтры требуют уже наличия каких-то знаний, полученных от примененного какого-то алгоритма обучения);
2. которые можно применять к еще необработанными данными.

Далее в ходе работы будем использовать фильтры unsupervised.

RemoveType, Remove – для удаления определенных атрибутов, в том числе и по типу – необходимо, так как не все типы могут быть использованы в различных алгоритмах;

Disctretize – для перевода числового атрибута в перечислимый;

RemoveUseless – для удаления атрибута, значение которого варьируется слишком сильно;

ReplaceMissing Values – для замещения отсутствующих значений средними по атрибуту;

На данной вкладке можно выбрать зависимую переменную и увидеть в графическом виде ее зависимость от текущего выбранного атрибута. При помощи Visualize All представить зависимость этой переменной от всех атрибутов в графическом виде.

Для лучшего понимания работы программы рассмотрим небольшой пример.

1. Описание данных, создание файла с расширением arff:

@relation 'Data'

@attribute city {Moscow,Belgorod,Krasnodar,Ivanovo,Lipetsk,Kaluga,Kazan,Vladimir,Rostov-on-Don,Voronezh,Bryansk}

@attribute age numeric

@attribute sex {Male,Female}

@attribute education {higher,average,heigher-not-full,school,school-9}

@attribute accumulation numeric

@attribute expenditure-month numeric

@attribute marital-status {Married-civ-spouse,Divorced,Never-married,Separated,Widowed,Married-spouse-absent,Married-AF-spouse}

@attribute holiday numeric

@attribute worked {yes,no}

@attribute income{<=25,25<50,>50}

@data

Belgorod,18,Male,school,12,5,Never-married,30,?,<=25

Moscow,25,Female,higher,50,23,Never-married,15,yes,>50

Krasnodar,45,Male,higher,126,45,Married-AF-spouse,14,no,>50

Voronezh,43,Female,average,100,20,Married-AF-spouse,43,yes,<=25

Bryansk,20,Male,average,10,20,Never-married,14,no,<=25

Rostov-on-Don,36,Male,higher,50,36,Married-AF-spouse,20,yes,25<50

Ivanovo,32,Female,higher,150,20,Widowed,14,no,25<50

1. Загрузка файла в программу Weka:

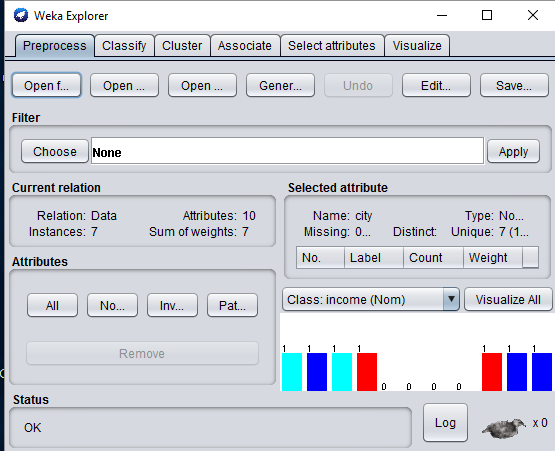


Рисунок 4 - Пример использование программы Weka

1. Для выполнения классификации переходим на вкладку Classify. Там мы увидим окно Test Options. Выбираем **Use Training Set,** чтобы использовать модель, которую WEKA создаст на основе наших данных из **\*.arff** – файла. Далее необходимо выбрать метод классификации:

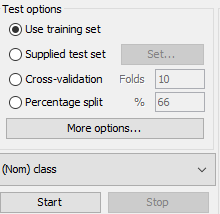
[](http://floppyy.ru/wp-content/uploads/2016/02/test_options.png)

Рисунок 5 - Вкладка Тest opinion

1. Жмем кнопку старт и получаем результат:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: Data

Instances: 7

Attributes: 10

city

age

sex

education

accumulation

expenditure-month

marital-status

holiday

worked

income

Test mode: evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree

------------------

education = higher: 25<50 (4.0/2.0)

education = average: <=25 (2.0)

education = heigher-not-full: <=25 (0.0)

education = school: <=25 (1.0)

education = school-9: <=25 (0.0)

Number of Leaves : 5

Size of the tree : 6

Time taken to build model: 0.03 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0.01 seconds

=== Summary ===

Correctly Classified Instances 5 71.4286 %

Incorrectly Classified Instances 2 28.5714 %

Kappa statistic 0.5625

Mean absolute error 0.1905

Root mean squared error 0.3086

Relative absolute error 43.4783 %

Root relative squared error 66.0819 %

Total Number of Instances 7

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

1,000 0,000 1,000 1,000 1,000 1,000 1,000 1,000 <=25

1,000 0,400 0,500 1,000 0,667 0,548 0,800 0,500 25<50

0,000 0,000 0,000 0,000 0,000 0,000 0,800 0,500 >50

Weighted Avg. 0,714 0,114 0,571 0,714 0,619 0,585 0,886 0,714

=== Confusion Matrix ===

a b c <-- classified as

3 0 0 | a = <=25

0 2 0 | b = 25<50

0 2 0 | c = >50

Из полученного результата можно увидеть, что:

* в блоке Classifier model строится дерево с указанием количества его уровня;
* в блоке Confusion Matrix строится матрица ответов на поставленный вопрос;
* в блоке Summary описывается время исполнения программы, вероятность неточного ответа.

Реализуем алгоритм С4.5 на выборке из 100,200,300,400,500 опрошенных людях, на основе этого составим сравнительную таблицу работы алгоритма.

Таблица 1 - Характеристика работы алгоритма C4.5 на данных о количестве заработной платы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Количество опрошенных | Точность классификации,% | Средняя ошибка метода,% | Средняя абсолютная ошибка | Средне-  квадратичная ошибка | Относительная абсолютная  Ошибка, % | Относительная квадратичная ошибка, % | Время выполнения моднли, сек |
| 100 | 60 | 40 | 0.3101 | 0.4519 | 71.4321 | 96.9976 | 0.02 |
| 200 | 81.5 | 18.5 | 0.1451 | 0.3308 | 33.4364 | 71.0219 | <0.01  (0.009) |
| 300 | 93.7 | 6.3 | 0.0649 | 0.2066 | 15.081 | 44.5302 | 0.02 |
| 400 | 93.25 | 6.75 | 0.0672 | 0.2088 | 15.6573 | 45.0861 | 0.022 |
| 500 | 93.014 | 6.986 | 0.0642 | 0.2093 | 14.8704 | 45.0503 | 0.02 |

Реализуем алгоритм VFDT на выборке из 100,200,300,400,500 опрошенных людях, на основе этого составим сравнительную таблицу работы алгоритма.

Таблица 2 - Характеристика работы алгоритма VFDT на данных о количестве заработной платы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Количество опрошенных | Точность классификации,% | Средняя ошибка метода,% | Средняя абсолютная ошибка | Средне-  квадратичная ошибка | Относительная абсолютная  Ошибка, % | Относительная квадратичная ошибка, % | Время выполнения моднли, сек |
| 100 | 80 | 20 | 0.2383 | 0.4089 | 52.5804 | 84.3023 | <0.01  (0.003) |
| 200 | 82 | 18 | 0.2307 | 0.3743 | 52.714 | 79.9834 | <0.01  (0.004) |
| 300 | 95.6667 | 4.333 | 0.044 | 0.1706 | 10.2212 | 36.7743 | <0.01  (0.0033) |
| 400 | 96 | 4 | 0.0382 | 0.1627 | 8.9065 | 35.1221 | <0.01  (0.0037) |
| 500 | 96.32 | 3.68 | 0.0341 | 0.1568 | 7.902 | 33.7487 | <0.01  (0.005) |

Для того, чтобы проанализировать и сравнить результаты работ алгоритмов составим графики.

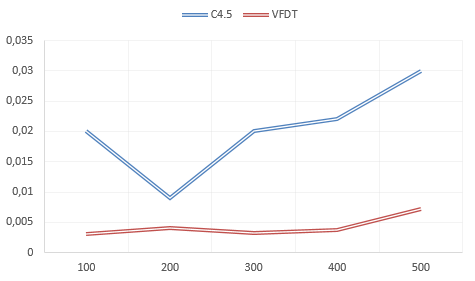


Рисунок 6 - График времени работы алгоритмов C4.5 и VFDT



Рисунок 7 - График точности классификации алгоритмов C4.5 и VFDT

Из рисунков 5 и 6 можно сделать вывод, что коэффициент точности у алгоритмов растет с увеличением получаемой информации, причем коэффициенты близки к равным. А вот время работы алгоритмов значительно различается с приростом информации, алгоритм С4.5 значительно поднимается в верх, а вот VFDT изменяется незначительно.

Для более точного анализа, рассмотрим еще одну статистику, для частоты эксперимента возьмем такую же выборку из 100,200,300,400,500 опрошенных людей.

Узнаем, какова вероятность одобрения заявки на кредит, для этого необходимо уточнить несколько параметров, таких как:

-город проживания;

- возраст;

-пол;

-кредитная история;

-стабилен ли доход;

-траты в месяц;

-семейное положения;

-история работы;

После обоработки информации алгоритмами, получим данные о работе программы:

Таблица 3 - Характеристика работы алгоритма C4.5 на данных о кредитовании

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Количество опрошенных | Точность классификации,% | Средняя ошибка метода,% | Средняя абсолютная ошибка | Средне-  квадратичная ошибка | Относит-я абсолютн. ошибка, % | Относит. квадратичная ошибка,% | Время выполн-я модели, сек |
| 100 | 64 | 36 | 0.3037 | 0.4538 | 68.5996 | 96.4417 | 0.01 |
| 200 | 85 | 15 | 0.1515 | 0.2753 | 34.4275 | 58.6792 | 0.01 |
| 300 | 86.6667 | 13.33 | 0.1356 | 0.2604 | 30.8819 | 55.5748 | 0.02 |
| 400 | 86 | 14 | 0.135 | 0.2598 | 30.727 | 55.4344 | 0.027 |
| 500 | 85.4 | 14.6 | 0.1444 | 0.2687 | 32.8844 | 57.347 | 0.03 |

Таблица 4 - Характеристика работы алгоритма VFDT на данных о кредитовании

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Количество опрошенных | Точность классификации,% | Средняя ошибка метода,% | Средняя абсолютная ошибка | Средне-  квадратичная ошибка | Относит-я абсолютн. ошибка, % | Относит. квадратичная ошибка,% | Время выполн-я модели, сек |
| 100 | 65 | 35 | 0.3218 | 0.3461 | 65.9655 | 90.646 | <0.01  (0.003) |
| 200 | 76 | 24 | 0.2724 | 0.3562 | 61.5379 | 75.723 | <0.01  (0.0035) |
| 300 | 94 | 6 | 0.0649 | 0.2066 | 15.081 | 44.5302 | <0.01  (0.005) |
| 400 | 97.2 | 2.8 | 0.0341 | 0.1568 | 7.902 | 33.7487 | <0.01  (0.0053) |
| 500 | 97.7 | 2.3 | 0.034 | 0.156 | 7.9 | 33.7480 | <0.01  (0.0072) |

Для более наглядного анализа построим графики времени работы алгоритмов и их точности.



Рисунок 8 - График времени работы алгоритмов C4.5 и VFDT для данных о кредитовании

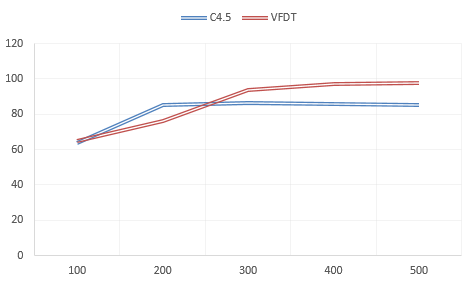


Рисунок 9 - График точности классификации алгоритмов C4.5 и VFDT для данных о кредитовании

Из рисунков 8 и 9 получили идентичные значения, что и в предыдущем примере.

# ЗАКЛЮЧЕНИЕ

По результатам работы можно сделать следующие выводы о алгоритме VFDT :

* время работы алгоритма увеличивается по степенно с увеличением потоковых данных;
* чем больше данных, тем выше точность алгоритма;
* справляется лучше других алгоритмов с увеличением потоковых данных.

Следующим шагом моей работы планируется исследование возможности реализации алгоритмов обработки потоковых данных в распределенном и параллельном режиме, и, в частности алгоритма VFDT.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

[1] M. Bahrepour, N. Meratnia, Z. Taghikhaki, and P. Havinga, Sensor fusion-based activity recognition for Parkinson patients // Sensor Fusion—Foundation and Applications, pp. 171–191, InTech

[2] K. P. Lam, M. Hoynck, B. Dong et al., Occupancy detection through an extensive environmental sensor network in an open-plan office building //Proceedings of the 11th International IBPSA Conference, pp. 1452–1459, Glasgow, Scotland, July 2009.

[3] D. Janakiram, V. A.Mallikarjuna Reddy, and A. V. U. P. Kumar, Outlier detection in wireless sensor networks using Bayesian belief networks // Proceedings of the 1st International Conference on Communication System Software and Middleware, pp. 1–6, 2006.

[4] Y.Hang and S. Fong, Stream mining over fluctuating network traffic at variable data rates // Proceedings of the 6th International Conference on Advanced Information Management and Service (IMS ’10), pp. 436–441, Seoul, Korea, November 2010.

[5] The Second International Knowledge Discovery and Data Mining Tools Competition, Sponsored by the American Association for Artificial Intelligence (AAAI) Epsilon Data Mining Laboratory Paralyzed Veterans of America (PVA), [Электронный ресурс], Статья,

<URL:http://www.kdnuggets.com/meetings/kdd98/kdd-cup-98.html>. (дата обращения: 5 апреля 2017 года)