РЕФЕРАТ

Работа состоит из: 93 страниц, 26 рисунков, 68 источников.

СЕТЕВАЯ БЕЗОПАСНОСТЬ, ВРЕДОНОСНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, РАСПОЗНАВАНИЕ ОБРАЗОВ

Настоящая дипломная работа содержит описание результатов идентификации сетевых протоколов прикладного уровня с помощью статистических методов классификации. Также в дипломной работе анализируются методы выделения признаков и кластеризации в применении к наборам сетевого трафика.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

СОА	Система обнаружения атак
СУБД	Система управления базами данных
ОС	Операционная система
ВПО	Вредоносное программное обеспечение
PBNS	Packet Based No State - Метод обработки, основанный на информации о пакете, без учета состояния
PBFS	Packet Based Per Flow State - Метод обработки, основанный на информации о пакете, с учетом состояния потока
MBFS	Message Based Per Flow State - Метод обработки, основанный на информации о сообщении, без учета состояния
MBPS	Message Based Per Protocol State - Метод обработки, основанный на информации о сообщении, с учетом состояния протокола

ВВЕДЕНИЕ

Настоящая работа посвящена методам классификации сетевого трафика на прикладном уровне. В качестве атрибутов классификации должны использоваться свойства пакетов транспортного (TCP-протокол) и сетевого (IP-протокол) уровней.

Классификация трафика применяется в широком диапазоне операций. Обычно она производится в соответствии с некоторым определенным заранее набором характеристик, при этом трафик разделяется на крупные классы (Chat, Interactive, VoIP и т.д.) или хорошо гранулированные классы, согласно протоколу (FTP, HTTP, SSH и т.д.). По завершении процесса классификации весь трафик должен быть соотнесен с соответствующими классами.

Классификация трафика применяется в области защиты информации:

- **Обнаружение атак.** Классификация трафика может быть включена в одну из частей автоматизированных СОА [1, 2, 3].

  В СОА Snort (рисунок 1) классификация применяется для выбора определенного типа трафика в процессе декодирования (часть модуля сбора данных или модуля анализа).

Рисунок 1. Компоненты потока данных СОА Snort

В [4] предложен вариант распределения трафика между несколькими СОА. При использовании только одной СОА задача защиты высокоскоростных сетей со временем становится затруднительной. Эффективным решением проблемы может стать разделение трафика, основанное на классификации (рисунок 2).

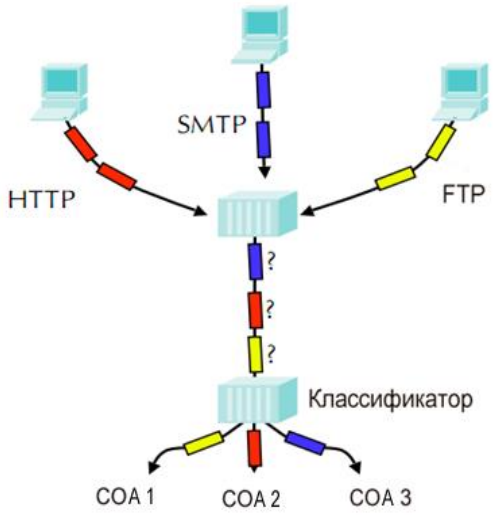

Рисунок 2. Пример распределения нагрузки между несколькими СОА

После классификации каждый класс трафика будет направлен только одной специальной СОА, группе СОА (или Honeypot для получения более подробной информации о целях и методах атаки) для обеспечения балансировки нагрузки. Таким образом, каждая СОА содержит меньший набор правил (упрощается журналирование) и выполняет более детальный анализ протокола, ограничивая себя подмножеством протоколов.

- **Обнаружение трафика, генерируемого вредоносным программным обеспечением.** Статистические методы классификации сетевого трафика предоставляют возможность обнаружения трафика, генерируемого ВПО, использующим стандартные протоколы прикладного уровня. Например, статистические методы позволяют обнаруживать трафик, генерируемый ВПО (троянами-даунлоадерами), которое загружает и устанавливает на компьютер-жертву новые версии вредоносных программ или рекламные системы [5].

Наиболее часто применяемые виды классификации трафика на основе известного номера порта и исследовании полезной нагрузки сетевых пакетов имеют ряд ограничений. Для их преодоления используются статистические методы распознавания шаблонов сетевого трафика. В связи с актуальностью на сегодняшний день проблемы распознавания трафика возникла задача исследования статистических методов классификации.

Сформулируем задачу дипломной работы.

Пусть *TS* является обучающим набором трафика, который представляется в виде входных/выходных пар:

$$TS = \{< x_1, y_1>, < x_2, y_1>, \ldots, < x_N, y_M >\}, \quad (1)$$

где x_i – вектор значений входных параметров пакетов, соответствующих *i*-ому пакету, и y_i – значение выходного класса-протокола. Задача классификации может быть сформулирована следующим образом: на основе обучающего набора трафика *TS* ищется функция *f(x)* от входных параметров, которая при наименьшем количестве ошибочно классифицированных пакетов предсказывает результирующий класс-протокол *y* для любых новых значений *x*. Выходные значения берутся из дискретного множества $\{y_1, y_2, \ldots, y_m\}$, которое включает все предопределенные значения классов-протоколов.

Другими словами,

$$\exists f : X \to Y, \quad (2)$$

где X — набор векторов атрибутов сетевых пакетов, Y — набор наименований классов. Значения целевой зависимости f известны только на объектах конечной обучающей выборки

$$X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}. \quad (3)$$

Требуется построить алгоритм

$$a : X \to Y, \quad (4)$$

способный классифицировать произвольный объект $x \in X$.

Задача дипломной работы заключается в идентификации сетевых протоколов прикладного уровня на основе статистических методов классификации трафика. В качестве атрибутов классификации необходимо использовать свойства пакетов транспортного (TCP-протокол) и сетевого (IP-протокол) уровней.

В ходе выполнения работы необходимо сделать выводы о применимости методов сокращения числа атрибутов и использовании кластеризации для автоматического выделения групп протоколов.

1. СПОСОБЫ КЛАССИФИКАЦИИ IP-ТРАФИКА

1.1. Таксономия способов классификации IP-трафика

Классификация (рисунок 3) IP-трафика основывается на исследовании TCP и UDP номеров портов пакетов (*классификация, основанная на портах*), реконструкции сигнатуры протокола из его полезной нагрузки (*классификация, основанная на полезной нагрузке*), статистических методов анализа характеристик обмена пакетами между хостами и статистических свойств сетевого трафика. Каждый из подходов обладает своими достоинствами и недостатками.

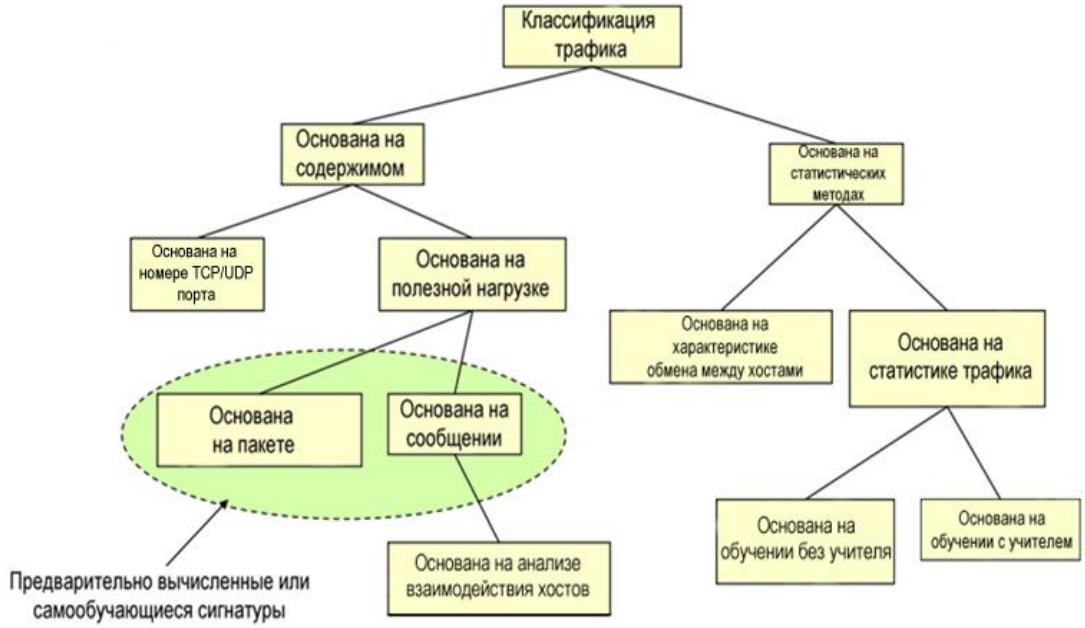

Рисунок 3. Методы классификации трафика

Ниже подробнее рассмотрим методы, приведенные в классификации.

1.1.1. Классификация IP-трафика на основе портов

Исторически многие приложения используют “хорошо известные” порты на своих локальных хостах. В этом случае задача классификатора заключается в поиске TCP SYN-пакетов (первый шаг в трехшаговом TCP-рукопожатии во время установления сеанса), чтобы определить серверную сторону нового клиент-серверного TCP-соединения. Затем, чтобы сделать вывод о приложении, просматривается целевой номер порта пакета в списке зарегистрированных портов IANA [6]. UDP использует порты похожим образом, но без установки соединения.

К несомненным достоинствам метода относятся простота реализации и высокая скорость работы.

Подход имеет ряд недостатков. Во-первых, некоторые приложения могут не иметь своих портов, зарегистрированных в IANA, к примеру, пиринговые приложения, такие как Napster и Kazaa [7]. Приложения могут использовать отличные от хорошо известных портов, чтобы обойти ограничения контроля доступа в ОС, например, непривилегированные пользователи на UNIX-подобных системах могут принудительно запустить HTTP-сервера на портах, отличных от 80. Также в некоторых случаях порты сервера выделяются динамически по мере необходимости. Например, RealVideo стример позволяет динамически согласовать серверный порт, используемый для передачи данных. Этот серверный порт определяется при инициализации TCP-соединения, которое устанавливается с использованием хорошо известного порта управления RealVideo [8].

В [9] отмечается не более чем 70% байтовой точности[1] для классификации, основанной на порте с использованием официального IANA списка. В [10] показали, что при таком анализе не удается определить 30-70% исследуемых потоков Интернет трафика. В [11] сообщили, что на порт по умолчанию приходилось только 30% от общего трафика (в байтах) для P2P протокола Kazaa.

В некоторых случаях шифрование IP уровня может запутать TCP и UDP заголовки, что делает невозможным определение фактического номера порта.

1.1.2. Классификация IP-трафика на основе полезной нагрузки

Чтобы избежать полной зависимости от номеров портов и собрать сведения об используемом протоколе, многие современные промышленные продукты используют восстановление состояния сеанса и прикладную информацию из содержимого каждого пакета.

В [11] показано, что классификация, основанная на полезной нагрузке, для P2P-трафика (путем исследования сигнатур трафика для прикладного уровня) может сократить ошибки первого и второго рода до 5% от общего числа байт большинства изучаемых P2P-протоколов.

В [12] приводится таксономия подходов на основе полезной нагрузки в зависимости от методов обработки и требований к памяти (рисунок 4).

[1] Байтовая точность оперирует количеством правильно классифицируемых байтов, в отличие от потоковой точности, которая при оценке использует потоки.

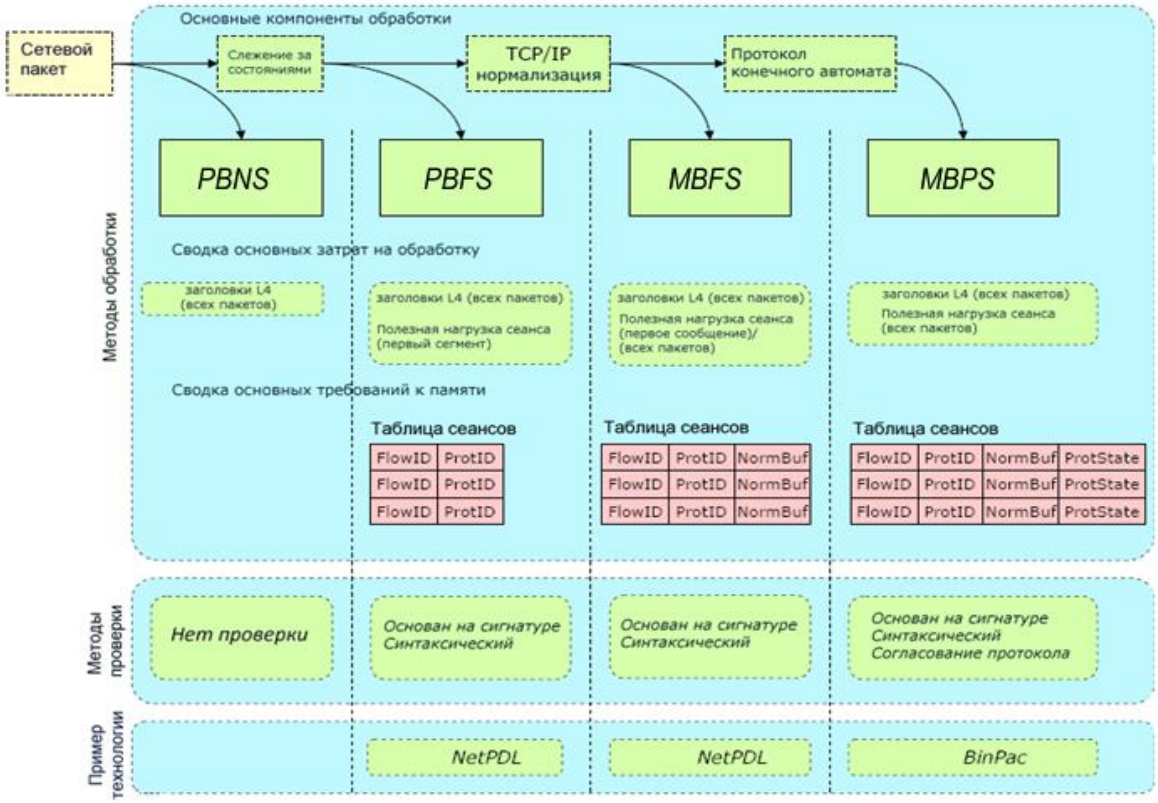

Рисунок 4. Классификация трафика на основе полезной нагрузки

Возрастающую сложность этих методов можно рассматривать через возрастание степени обработки слева направо (например, простейший метод требует только разбора заголовков L2-L4[2], в то время как наиболее сложный, требует обработки полезной нагрузки во всех пакетах), а также в соответствии с требованиями к памяти.

Далее будут рассмотрены проблемы классификации трафика с точки зрения возможных методов проверки, а также перечень методов обработки, которые могут быть использованы в рамках классификации на основе полезной нагрузки.

Можем выделить четыре различных степени проверки. Первая степень проверки основана на *сигнатуре*, ее цель состоит в поиске некоторых сигнатур в рамках полезной нагрузки прикладного уровня. Так, например, HTTP-пакет начинается с команды, следующей за URL и версией протокола, в то время как большинство Edonkey пакетов имеет поля, содержащие размер полезной нагрузки. Метод на основе сигнатур строится на соответствии полезной нагрузки (или ее части) с сигнатурой, определенной для данного протокола. Сигнатуры, как правило, являются регулярными выражениями.

[2] Уровни модели OSI: канальный уровень L2 соответствует протоколу Ethernet, сетевой L3 – протоколу IP, транспортный L4 – протоколам TCP/UDP.

Второй уровень проверки – *синтаксический*. Он может рассматриваться как более точная версия сигнатурной проверки, поскольку направлен на проверку правильности переданных данных с синтаксической точки зрения (к примеру, предполагается, что полезная нагрузка HTTP должна содержать HTTP заголовки). В этом случае необходимо декодировать все поля, содержащиеся в сообщении, и гарантировать, что сообщение является хорошо сформированным.

Третья степень контроля связана с *протоколом соответствия*. Например, она контролирует, что на HTTP GET запрос от клиента следует действительно ответ от сервера. Такая форма контроля является более точной, поскольку она может проверять в соответствии со спецификацией реальное поведение протокола.

Четвертая степень контроля относится к *семантике данных*. К примеру, она способна проверить, является ли объект, передаваемый по протоколу HTTP, изображением или какой-либо другой формой содержания. Такой контроль очень полезен для обнаружения "туннелей", в которых приложение использует другой протокол для транспортировки данных. На данный момент это наиболее неформализованный метод.

Рассмотрим различные методы обработки, указанные на рисунке 4. Простейшим методом является PBNS, который работает, проверяя значения некоторых полей (например, TCP/UDP порты), присутствующих в каждом пакете. Этот метод очень прост с точки зрения вычислений (должны быть обработаны только заголовки пакета до L4), для него не требуется хранить состояния.

Второй метод – PBFS требует реализации таблицы сеансов, в которой каждая запись включает идентификатор сеанса (пять кортежей IP источник/назначения, транспортный протокол, порт источника/назначения), и соответствующий протокол прикладного уровня (ID протокола). Каждая таблица занимает по нескольку десятков байт.

Работа третьего метода MBFS основана на сообщениях. Для этого метода требуется модуль нормализации[3] TCP/IP пакетов. Технологии на основе MBFS могут выполнить те же самые проверки, что и PBFS, но работают на сообщениях, следовательно, их средства управления могут быть расширены на все сообщение целиком взамен первого сегмента данных. В таком случае, требуемые объемы памяти увеличиваются из-за дополнительной информации о состояниях, которая должна быть сохранена для каждого сеанса (например, порядковый номер TCP) и из-за буферов, требуемых TCP/IP нормализатором. Все эти

[3] В случае методов MBFS этот модуль должен обязательно присутствовать и иметь возможность вернуть TCP/IP пакеты таким способом, чтобы следующие модули могли оперировать с целым сообщением прикладного уровня, даже, если сообщение было разбито по разным пакетам.

параметры сильно зависят от природы трафика, то есть от количества фрагментированных пакетов и “ненормальных” (с пропущенными сегментами и т.д.) TCP сеансов. В зависимости от реализации, некоторые продукты могут выполнить синтаксическую проверку для всех сообщениях.

Четвертая категория MBPS точно интерпретирует, что передает и получает каждое приложение. Обработчик MBPS понимает не только семантическую часть сообщения, но и различные этапы обмена сообщениями (например, HTTP GET, должен сопровождаться соответствующим кодом ответа от веб-сервера), так как этот метод полностью понимает конечный автомат протокола. Требуемые объемы памяти становятся еще большими, потому что надо учитывать не только состояние транспортного сеанса, но также и состояние каждого сеанса прикладного уровня. Производительность является самой высокой среди всех методов – все данные прикладного уровня должны быть обработаны, чтобы проверить соответствие протокола. Реализации, основанные на технологии PBFS, обычно с каждым сеансом связывают некоторое дополнительное состояние, чтобы выполнить более точную классификацию. Например, некоторые приложения (Skype, VoIP), могут быть обнаружены, проверяя шаблон из нескольких последовательных пакетов.

Существующие практические разработки (NetPDL [13], NBAR [14], SML [15], BinPac [16]) полностью не укладываются в эту таксономию, поскольку одна и та же технология может принадлежать нескольким категориям. В этом случае используют другой подход, разделяют технологии на основе пакетов (NetPDL, NBAR) и на основе потоков (SML, BinPac). В разных реализациях эти технологии могут вести себя по-разному, например, NetPDL и NBAR могут относиться либо к PBFS, либо к MBFS в зависимости от наличия TCP/IP нормализатора на пути обработки данных.

В [9], используя технологии, сочетающие порты и полезную нагрузку, определяли сетевые приложения. Процедура классификации начиналась с изучения номера порта потока. Если использовался неизвестный порт, поток передавался на следующий этап. На втором этапе проверялся первый пакет на наличие известной сигнатуры. Если она не была найдена, тогда пакет просматривался на наличие известного протокола. Если эта проверка заканчивалась неудачей, просматривался первый килобайт потока на наличие сигнатуры протокола. Потоки, оставшиеся неклассифицированными после этого уровня, потребуют просмотра всей полезной нагрузки потока. Результаты показали, что при наличии информации о порте правильно классифицировать возможно 69% от общего количества байт. Включая информацию, полученную из первого килобайта каждого потока, точность увеличивается почти до 79%. Самая высокая точность (почти 100%) может быть

достигнута только изучением полезной нагрузки оставшихся неклассифицированных потоков.

Хотя исследование полезной нагрузки избегает фиксированных номеров портов, оно накладывает существенные сложности и загружает устройство идентификации трафика. Такое устройство должно обладать обширными знаниями о семантике прикладных протоколов, должно быть достаточно мощным, чтобы выполнять одновременно анализ потенциально большого числа потоков. Этот подход может быть затруднен или неосуществим, когда речь идет о запатентованных протоколах или шифрованном трафике.

1.1.3. Классификация трафика на основе статистических методов

В статистических методах необходимо различать два разных подхода: поведенческие алгоритмы и статистические алгоритмы сетевого и транспортного уровней. Рассмотрим более подробно каждый из этих подходов.

Концепция поведенческого алгоритма была разработана в [17], основная цель метода состоит в том, чтобы определить, какие приложения создают определенные потоки трафика. Анализируя, как в рамках сети взаимодействуют хосты, можно определить, какие виды приложений запущены на хосте. Отношения между классом трафика и его поведенческими статистическими свойствами были описаны в [19, 20, 21, 22, 23], где авторы проанализировали и построили эмпирические модели характеристик соединения для ряда специальных TCP-приложений.

Подход статистических методов опирается на статистические характеристики трафика для идентификации приложения. Предположения, лежащие в основании таких методов, заключаются в том, что сетевой трафик обладает статистическими характеристиками, которые являются уникальными для определенных классов приложений и позволяют разделить различные исходные приложения [18].

Статистические алгоритмы (подробнее о современных методах см. приложение 1) в зависимости от подхода к классификации можно разделить на две группы: методы классификации или обучение с учителем (см. п. 2.4) и методы кластеризации или обучение без учителя (см. п.2.3).

Рассмотрим подробнее этапы применения методов машинного обучения с учителем для классификации сетевого трафика.

1.2. Применение методов машинного обучения для классификации IP-трафика

В случае, когда машинное обучение применяется для классификации IP-трафика, ряд понятий изменяют свой смысл. С целью дальнейшего обсуждения определим следующие три термина, относящиеся к потокам:

- *поток* или *однонаправленный поток*: ряд пакетов, разделяющих одинаковый кортеж из пяти элементов: IP-адреса источника и получателя, номера портов источника и получателя, номер протокола;
- *двунаправленный поток*: пара однонаправленных потоков, протекающих в противоположных направлениях, между теми же самыми IP-адресами источника и назначения и портами;
- *полный поток*: двунаправленный поток, захваченный за все его время существования от создания до завершения соединения.

Класс обычно указывает на IP-трафик, сформированный приложением или группой приложений. В качестве признаков обычно выступают числовые атрибуты, вычисленные на основании сетевых пакетов. Не все признаки одинаково влияют на процесс классификации, поэтому на практике классификаторы выбирают наименьшее множество признаков, которое приведет к эффективному разделению.

Рисунки 5, 6 и 7 иллюстрируют шаги, связанные с построением классификатора трафика, использующего алгоритм обучения с учителем (контролируемое машинное обучение).

Рисунок 5 охватил полный процесс обучения и проверки, которые происходят в классификационной модели. Оптимальный подход к алгоритму обучения с учителем должен предусматривать предварительно классифицированные образцы двух типов IP-трафика (более подробно о методе см. п. 2.4.): трафика, соответствующего классу, который хотим позднее идентифицировать в сети, и трафика от других приложений, которые, возможно, встретятся в будущем (часто называемый *вмешивающимся трафиком*).

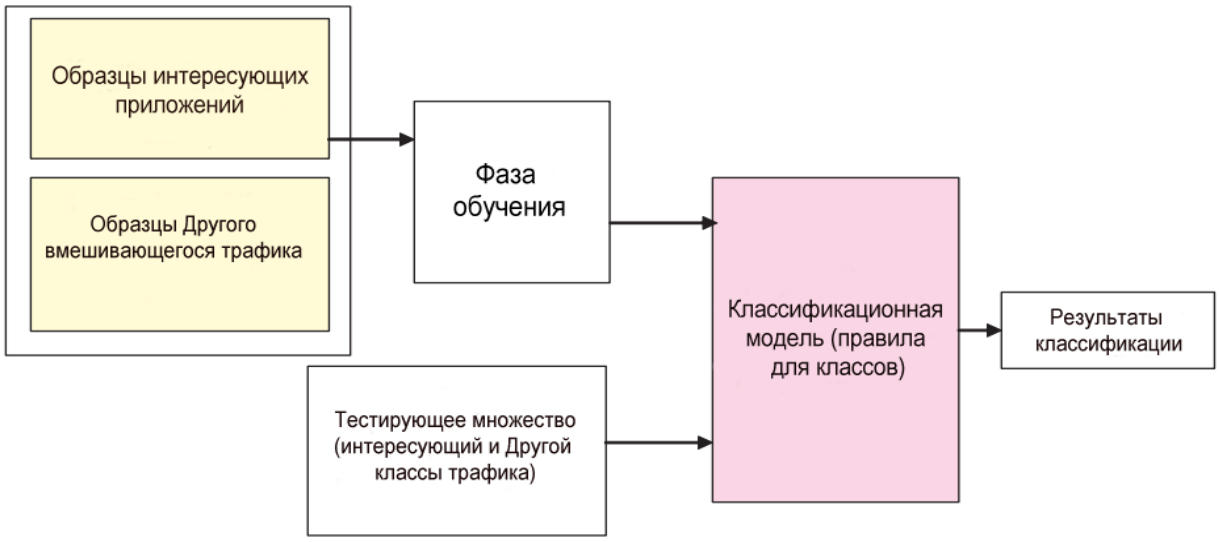

Рисунок 5. Обучение и тестирование для двухклассового классификатора трафика с учителем

Рисунок 6 подробно останавливается на последовательности событий, связанных с обучением классификатора с учителем. Сначала собирается смесь “трасс трафика”, которая включает экземпляры интересующего нас приложения и экземпляры других вмешивающихся приложений (таких как, HTTP, DNS, SSH и/или P2P). Шаг “обработка статистики потока” включает вычисление статистических свойств этих потоков и подводит к началу формирования признаков.

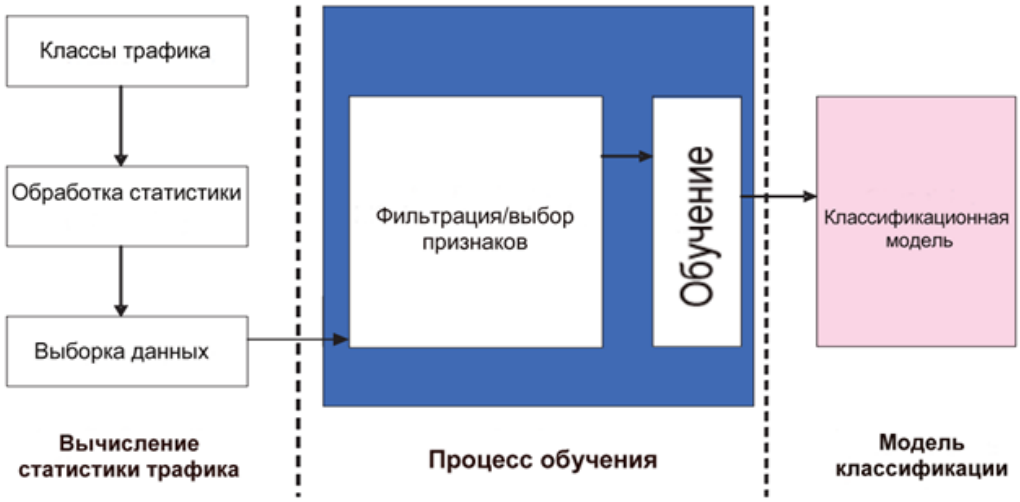

Рисунок 6. Обучение классификатора с учителем

Следующий необязательный шаг – “осуществление выборки данных”, разработанный, чтобы сузить область поиска для обучающего алгоритма, когда он сталкивается с чрезвычайно большими обучающими наборами данных (трассами трафика). Шаг осуществления выборки извлекает статистику из подмножества случаев

различных прикладных классов, и передает их классификатору, который будет использоваться в обучающем процессе.

Шаг фильтрации/выбора признаков желателен, чтобы ограничить число признаков (более подробно о методе см. п. 2.2.), действительно используемых при обучении классификатора, и, таким образом, создавать модель классификации. Выходной сигнал на рисунке 6 – модель классификации.

Перекрестная проверка (или многослойная перекрестная проверка) может использоваться, чтобы произвести результаты оценки точности во время фазы обучения. Однако если исходный набор данных будет состоять из IP-пакетов, собранных в то же самое время и в той же самой измеряемой сетевой точке, то результаты перекрестной проверки, вероятно, слишком высоко оценят точность классификатора. В идеальном случае исходный набор данных должен бы содержать смесь трафика, собранного в разное время и разных точках сети, или использовать полностью независимо собранные обучающие и тестирующие наборы данных.

На рисунке 7 показан поток данных в пределах действующего классификатора трафика, использующего модель, построенную на рисунке 5.

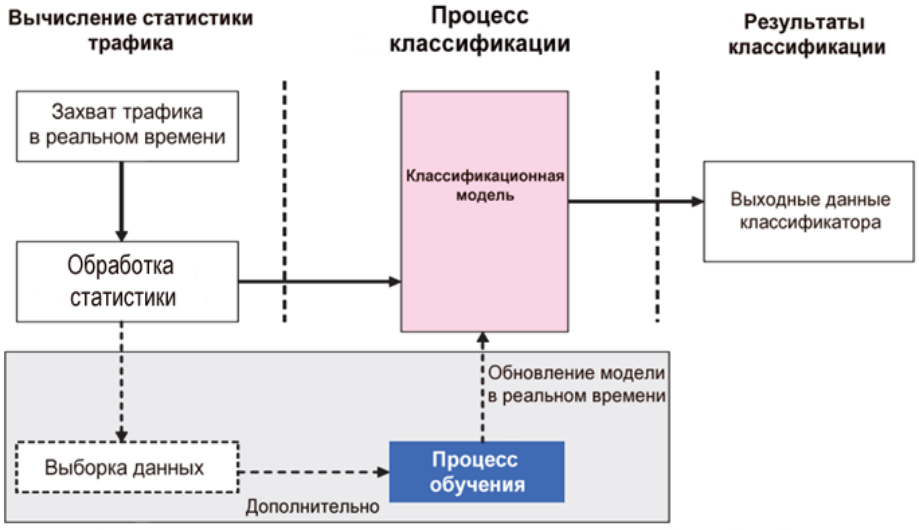

Рисунок 7. Поток трафика в рамках оперативной классификации с учителем

Трафик, собранный в реальном времени, используется, чтобы вычислить статистику потока, от которой определяются признаки, передающиеся затем в модель классификации. Здесь мы предполагаем, что набор признаков, вычисленных от захваченного трафика, ограничен оптимальным набором признаков, определенных во время обучения. На выходе классификатора указывается, какие потоки, предполагается,

являются членами интересующего класса (как определено моделью). Дополнительная реализация может позволить модели обновляться в реальном времени (выполняя выборку данных и процесс обучения, как показано на рисунке 5). Для контроля над тестированием и оценкой точности могут использоваться автономные трассы трафика взамен захвата в реальном времени.

Обучение требует априорной классификации (или маркировки) потоков внутри обучающих наборов данных. Поэтому схему обучения с учителем можно использовать для определения признаков (или групп) интересуемых приложений. Тем не менее, как отмечено ранее, классификатор лучше всего работает в случае обучения его на образцах всех классов, ожидаемых встретиться на практике. Следовательно, его выполнение может быть ухудшено или искажено, если не провести обучение на образце смешенного трафика или в сети появится трафик ранее неизвестных приложений. Поэтому при оценке схем обучения с учителем стоит учитывать, каким образом классификатор будет обеспечиваться адекватными обучающими образцами, когда будет необходимо переобучение, и как пользователь обнаружит новый тип приложений.

1.3. Библиотека алгоритмов машинного обучения Weka

Программная система Weka – коллекция современных машинных обучающих алгоритмов и инструментов предварительной обработки данных. Она включает практически все известные алгоритмы анализа и разработана так, чтобы предоставить возможность гибко испытать существующие методы на новых наборах данных. Программная система Weka включает методы для всех стандартных задач интеллектуального поиска данных: регрессии, классификации, кластеризации, ассоциативных интеллектуальных правил и выбора признака. Кроме того, предоставляется множество услуг по визуализации данных и инструментов предварительной обработки. Все алгоритмы принимают входные параметры в форме таблицы формата ARFF, который может быть прочитан из файла или сгенерирован запросом базы данных. Weka имеет пользовательский интерфейс Исследователь (Explorer), но та же функциональность доступна через компонентный интерфейс Поток Знаний (Knowledge Flow) и из командной строки. Имеется отдельное приложение Экспериментатор (Experimenter) для сравнения предсказательной способности алгоритмов машинного обучения на заданном наборе задач [28].

2. ПРОВЕДЕННЫЕ ИССЛЕДОВАНИЯ

Задачей дипломной работы ставится идентификация сетевых протоколов прикладного уровня на основе статистических методов классификации трафика. В качестве атрибутов классификации использовались свойства пакетов транспортного (TCP-протокол) и сетевого (IP-протокол) уровней. Дополнительно, возникла задача исследовать возможность сокращения числа атрибутов и применения методов кластеризации для предварительного выделения групп протоколов.

Проведенные исследования методов машинного обучения для классификации IP трафика можно разделить на три части:

- *выбор признаков:* включает поиск наилучшего метода сокращения числа параметров для процесса классификации (кластеризации);
- *кластеризация:* сосредотачивается на исследовании возможности автоматического разбиения на классы, описывающие протоколы, и сравнении полученного разбиения с предварительно классифицированным трафиком на основе известного номера порта;
- *классификация:* включает поиск наилучшего метода классификации и классификацию ранее не наблюдавшегося сетевого трафика.

2.1. Подготовка данных

Подготовка данных для анализа статистических методов классификации трафика проходила в несколько этапов:

- захват трафика с помощью программы *Wireshark* [29] и сохранение полученных данных в формате *tcpdump* [30];
- разбор содержимого пакетов с помощью утилиты на языке Perl *pcap2mysql* [31], которая помещает всю информацию о пакетах в базу данных *MySQL* [32];
- обработка данных: выбор необходимых атрибутов для характеристики трафика, классификация трафика на основании известного номера порта с помощью PHP-скрипта;
- конвертирование полученных данных в формат CSV [33] для дальнейшей их передачи в программу Weka.

Схема стенда, на котором производилась подготовка данных, показана на рисунке 8.

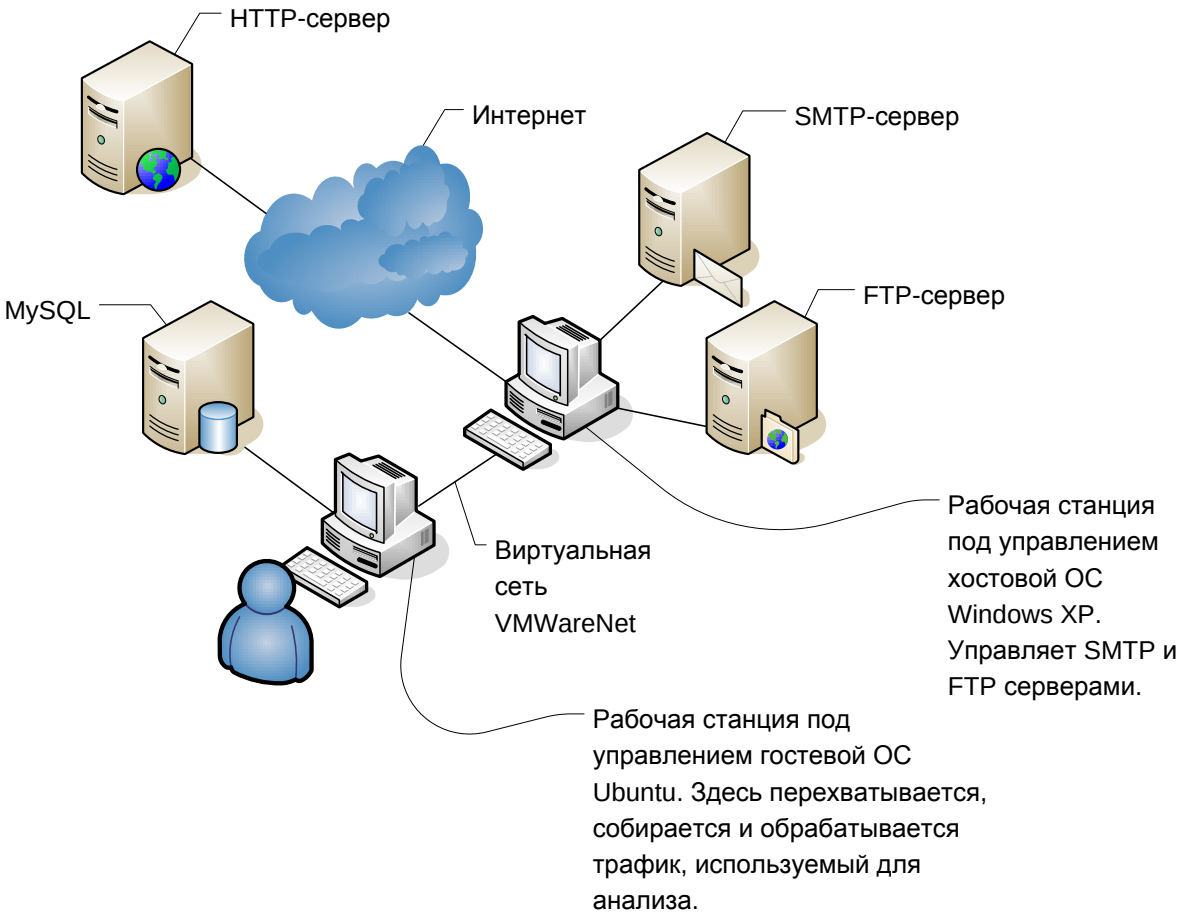

Рисунок 8. Схема стенда, на котором производилась подготовка данных и их анализ.

В качестве данных для исследования был выбран трафик, содержащий сетевые пакеты, которые принадлежат трем протоколам (HTTP, SMTP и FTP-commands), и трафик, сгенерированный вредоносным программным обеспечением, использующим 80 порт (таблица 1).

Таблица 1. Характеристики исследуемых дампов трафика

Название дампа (обучающего или тестирующего)	Количество пакетов, относящихся к протоколу и вредоносному трафику				Общее количество пакетов
	HTTP	FTP-C	SMTP	Trojans	
Training-Set-1	6260	6523	6840	-	19623
Training-Set-2	3924	3076	3996	-	10996
Training-Set-3	6256	6521	6838	2532	22147
Test-Set-1	3517	-	-	-	3517
Test-Set-2	-	3494	-	-	3494
Test-Set-3	-	-	6209	-	6209
Test-Set-4	-	-	-	5666	5666

Для исследуемых наборов пакетов трафика прочерками отмечены полностью отсутствующие в дампе протоколы и вредоносный трафик. На рисунке 9 показана диаграмма соотношения количества сетевых пакетов различных типов трафика, содержащихся в каждом из обучающих дампов (Training-Set).

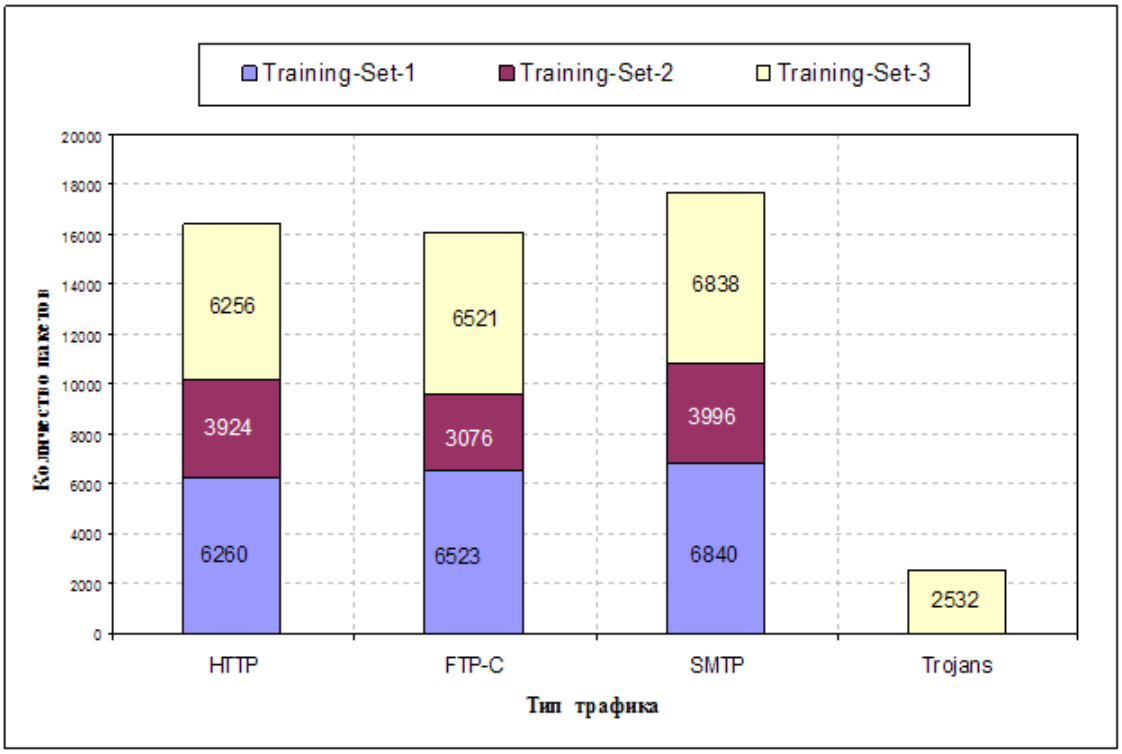

Рисунок 9. Диаграмма соотношения числа пакетов в обучающих множествах

Для получения трафика, необходимого для дальнейшего исследования, с рабочей станции под управлением гостевой ОС Ubuntu (см. рисунок 8) происходило обращение к локальным (FTP и SMTP) и глобальным (HTTP) серверам через виртуальную сеть VMWareNet, в зависимости от типа дампа (обучающий или тестирующий). Локальные сервера (FTP и SMTP) запускались на рабочей станции под управлением хостовой ОС Windows XP.

В обучающее множество Training-Set-3, уже содержащее трафик протоколов HTTP, SMTP и FTP-C, был добавлен дамп, полученный в результате перехвата трафика, сгенерированного ВПО, использующим 80/tcp порт. Для получения большого числа сетевых пакетов на исследуемой рабочей станции последовательно запускались несколько типов вредоносных программ: Trojan-Downloader.Win32.Phifwbypass, Trojan-Downloader.Win32.Squire.b, Trojan-Downloader.Win32.Lalus, Trojan-Downloader.Win32.Nooper.a, Trojan-Downloader.Win32.MlFree. Аналогично, для получения сетевого трафика тестирующего множества Test-Set-4 были запущены следующие зловредные программы: Trojan-Downloader.Win32.Comet, Trojan-

Downloader.Win32.DlxDown, Trojan-Downloader.Win32.Druser.h, Trojan-Downloader.Win32.Delmed.b, Trojan-Downloader.Win32.Brok, Trojan-Downloader.Win32.Dadobra.z.

Так как дампы, предназначенные для тестирования и обучения, были заранее классифицированы на основании известного номера порта (обращение к серверам происходило на стандартные номера портов, HTTP – 80, FTP-C – 21, SMTP - 25), то можно посчитать процент неправильно классифицированных и кластеризованных пакетов.

В качестве атрибутов для обучения были выбраны параметры сетевых пакетов, представленные в таблице 2.

Таблица 2. Атрибуты, выбранные для характеристики трафика

Название атрибута	Описание
ip_flags	Флаги фрагментации.
ip_ttl	Время в секундах, в течении которого пакет может находиться в сети.
tcp_winsize	Размер окна
tcp_data_len	Размер данных. Вычисляется по формуле: ip.len-ip.hlen*4-tcp.hlen*4
tcp_flow_dir	Направление потока (от сервера к клиенту/ от клиента к серверу). Определяется на основании известного номера порта назначения и IP-адреса источника.
app_proto_name	Название протокола. Используется для классификации и оценки кластеризации, определяется на основании известного номера порта.

Из рассмотрения были убраны параметры: tcp_urg (флаг, обеспечивающий экстренную отправку данных, в экспериментальной выборке для всех протоколов принимал значение 0), tcp_seqnum и tcp_acknum (порядковый номер и номер подтверждения соответственно, не зависят от протокола), tcp_flags (флаги управления, значение стандартного отклонения было равно 0).

Рассмотрим более подробно этап подготовки данных – обработка. Для хранения выбранных сведений о пакетах в базе данных под управлением СУБД MySQL была создана таблица *pcap_weka:*

```
CREATE TABLE `pcap_weka` (
  `id` bigint(20) unsigned NOT NULL default '0',
  `ip_s` char(15) NOT NULL default '',
  `ip_d` char(15) NOT NULL default '',
  `ip_flags` tinyint(1) unsigned NOT NULL default '0',
  `ip_ttl` tinyint(1) unsigned NOT NULL default '0',
  `tcp_s_port` int(3) unsigned NOT NULL default '0',
  `tcp_d_port` int(3) unsigned NOT NULL default '0',
  `tcp_winsize` smallint(5) unsigned NOT NULL default '0',
  `tcp_data_len` smallint(5) unsigned NOT NULL default '0',
  `tcp_flow_dir` smallint(5) unsigned NOT NULL default '0',
  `app_proto` smallint(5) unsigned NOT NULL default '0',
  `app_proto_name` varchar(10) NOT NULL default '',
  PRIMARY KEY  (`id`)
) TYPE=MyISAM;
```

Заполнение таблицы происходило с помощью следующего SQL-скрипта:

```
insert into pcap_weka (id, ip_s, ip_d, ip_flags, ip_ttl, tcp_s_port,
tcp_d_port, tcp_flags, tcp_winsize, tcp_data_len)
select
ip.id,ip.s_ip,ip.d_ip,ip.flags,ip.ttl,tcp.s_port,tcp.d_port,tcp.flags,tcp.win
size,ip.len-ip.hlen*4-tcp.hlen*4  from ip, tcp where ip.id=tcp.id;
```

На рисунке 10 показаны таблицы базы данных, которые использовались в процессе подготовки к исследованию (таблицы ip и tcp созданы программой *pcap2mysql* [31]).

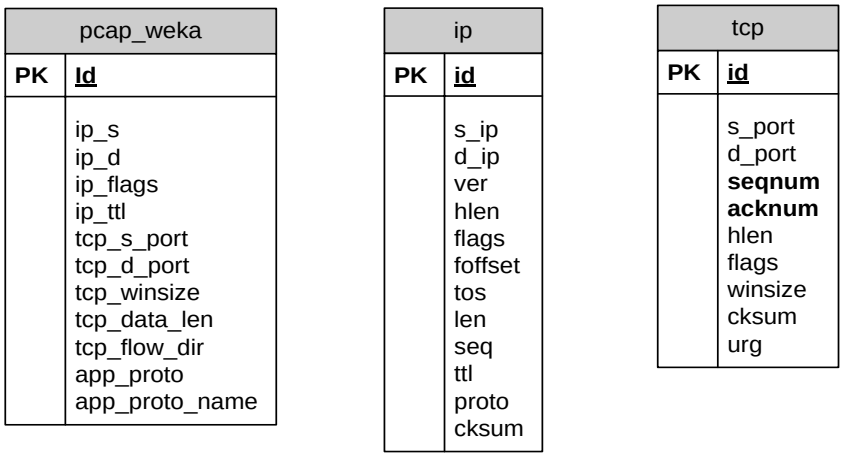

Рисунок 10. Таблицы базы данных, используемые при подготовке сетевого трафика для анализа

Далее, заполнялись недостающие поля таблицы *pcap_weka*: на основании известного номера порта тип протокола (HTTP, SMTP FTP-C), которому принадлежит пакет и направление перемещения пакета (1 – от клиента серверу, 2 – от сервера клиенту). Эти операции выполнялись автоматически с помощью PHP-скрипта. Пример PHP-скрипта, в котором заполняются поля протокола и направления пакета для 80 порта (21 и 25 порты анализируются аналогично):

```
<?php
// Установка соединения с базой данных
$con = @mysql_connect($host,$user,$psw);
if (!$con){ die('Ошибка соединения: ' . mysql_error()); }
// Выбор базы данных
 mysql_select_db("pcap2mysql_data", $con)
      or die("Ошибка выбора базы данных");

// Выполнение запроса
$result_id = mysql_query("select * from pcap_weka")
      or die("Нельзя выполнить запрос");
$flag1=0;

while ($row = mysql_fetch_row($result_id)) {
  for ($i=0; $i < mysql_num_fields($result_id); $i++){
   if ($row[8]==80){ // указываем порт назначения - 80
    if($flag1==0){
      $flag1=1;
// Обновляем поля таблицы, указывая направление и тип протокола
      $query=sprintf("update    pcap_weka    set    pcap_weka.tcp_flow_dir=1,
      pcap_weka.app_proto=1,       pcap_weka.app_proto_name='http'       where
      pcap_weka.id=%d",$row[0]);

      mysql_query($query) or die("Нельзя выполнить запрос, ошибка 1");
    }else{
      $query=sprintf("update    pcap_weka    set    pcap_weka.tcp_flow_dir=1,
      pcap_weka.app_proto=1,       pcap_weka.app_proto_name='http'       where
      pcap_weka.id=%d",$row[0]);

      mysql_query($query) or die("Нельзя выполнить запрос, ошибка 2");
    }
   }
   if ($row[7]==80){
   if($flag1==1){
```

```
      $query=sprintf("update    pcap_weka    set    pcap_weka.tcp_flow_dir=2,
      pcap_weka.app_proto=1,       pcap_weka.app_proto_name='http'       where
      pcap_weka.id=%d",$row[0]);

      mysql_query($query) or die("Нельзя выполнить запрос, ошибка 3");
   }
  }
 }
}
mysql_free_result($result_id);
mysql_close($con);
?>
```

Перейдем к рассмотрению проведенных исследований.

2.2. Алгоритмы выделения признаков

Ключом к построению обучающего классификатора является определение наименьшего необходимого набора признаков, требуемого для достижения наилучших показателей – процесс, известный как *выделение признаков*.

Качество признаков является критическим для выполнения обучающего алгоритма. Использование неподходящих или избыточных признаков часто приводит к негативным последствиям для точности большинства обучающих алгоритмов. Также это может сделать более дорогой систему в вычислительном смысле. Следовательно, желательно выбрать подмножество признаков, которое является небольшим, но все же, сохраняет существенную и полезную информацию об интересующих классах.

Алгоритмы выделения признаков можно в общем случае разделить на фильтрующий метод (рисунок 11) и оберточный метод (рисунок 12).

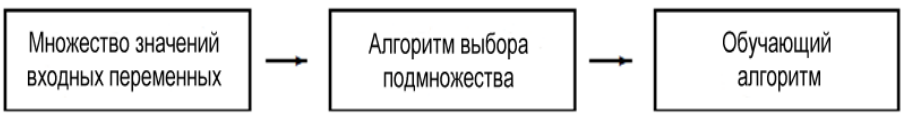

Рисунок 11. Схема фильтрующего метода выделения признаков

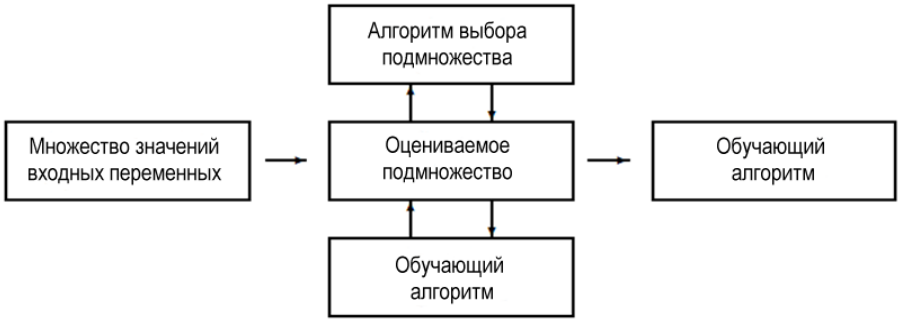

Рисунок 12. Схема оберточного метода выделения признаков

Алгоритмы фильтрующего метода создают независимую оценку, основанную на общих характеристиках данных. Они опираются на определенные метрики, чтобы оценить или выбрать лучшие подмножества до начала обучения. Поэтому на подготовленные результаты не могут оказать влияние детали обучающего алгоритма. С другой стороны, алгоритмы метода обертки оценивают работу различных подмножеств, используя обучающий алгоритм, который будет, в конечном счете, использоваться для обучения. Поэтому на его результаты оказывает влияние используемый обучающий алгоритм. Могут использоваться многочисленные методы поиска подмножеств. Часто у оберточных методов результат лучше (с точки зрения окончательного прогнозирования точности), чем у методов фильтрации, поскольку выбор признаков оптимизирован под особенности обучающего алгоритма. Однако поскольку алгоритм обучения применяется при оценке каждого множества признаков, оберточный метод становится стишком дорогим для запуска и трудоемким для больших баз данных, которые содержат огромное число особенностей. К преимуществам методов фильтрации можно отнести скорость работы, отсутствие требований к повторному выполнению для различных алгоритмов обучения [34].

Другая полезная таксономия может быть получена при делении алгоритмов на те, которые оценивают (ранжируют), отдельные признаки и те, которые ранжируют, подмножества признаков.

Подмножества признаков, которые подвергаются оценке, порождаются с помощью различных методик поиска. Остановимся на них подробнее.

Жадный поиск (*Greedy search*) взвешивает локальные изменения к текущему подмножеству через добавление и удаление признаков. Для данного ‘родительского’ множества жадный поиск исследует все возможные ‘дочерние’ подмножества или через добавление, или через удаление признаков. Дочернее подмножество, которое показывает

наилучшую меру, затем заменяет родительское подмножество и процесс повторяется. Процесс завершается, когда более не может быть произведено улучшений.

Первый лучший (*Best First search*) похож на алгоритм жадного поиска. Однако он обладает способностью возвращаться вдоль подмножества выбранного пути для исследования других возможностей, когда текущий путь больше не показывает улучшений. Предотвращая поиск от возврата через все варианты в пространстве признаков, установлено ограничение на количество не улучшаемых подмножеств. В данном случае используется число 5 [35].

Дополнительные детали могут быть найдены в [36, 37, 38, 39].

Для исследования методов выбора признаков были рассмотрены несколько алгоритмов: PCA, InfoGain, CFS и Wrapper. С использованием программного комплекса Weka применим к обучающему множеству Training-Set-3 каждый из этих алгоритмов и проанализируем полученные результаты.

2.2.1. Метод главных компонент (Principal Components Analysis, PCA)

Центральная концепция метода главных компонент – это понятие главного компонента (ГК) – скрытой переменной, которая не может быть явно объявлена и непосредственно измерена. Скрытая переменная является линейной комбинацией исходных переменных, которая может быть формально определена как собственный вектор ковариационной матрицы данных. Главные компоненты показывают скрытые систематические связи, присущие исходному набору данных (см. рисунок 13).

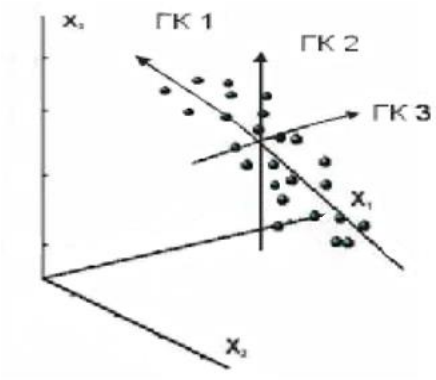

Рисунок 13. Поиск главных компонент модели

При этом новая модель имеет существенно меньшее количество переменных, поэтому такой подход может интерпретироваться как проекционный, когда исходные

данные проецируются на гиперплоскость (см. рисунок 14) меньшей размерности по сравнению с исходным пространством.

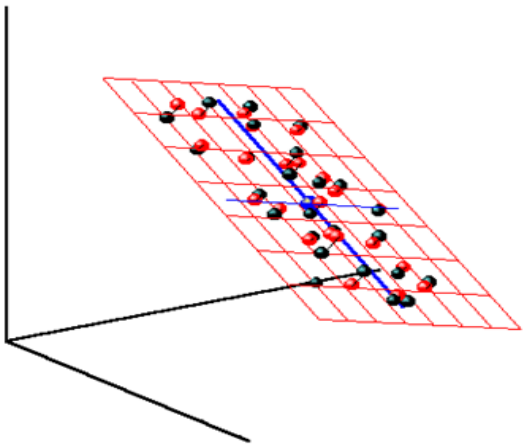

Рисунок 14. Проекция данных на плоскость главных компонент

Достоинства и недостатки метода PCA заключаются одновременно в непараметрическом анализе. Если не используются параметры для настройки и нет коэффициентов, основанных на пользовательском опыте, то результат будет однозначным и независимым от пользователя. Это же можно рассматривать как недостаток метода главных компонент, так как, если заранее известны некоторые особенности структуры системы, то имеет смысл включить эти особенности в алгоритм.

Результат выполнения метода главных компонент для множества Training-Set-3 в Weka представлен ниже:

```
Метод поиска: ранжирование атрибутов
Оценка атрибута (без учителя):
      Преобразовательный метод главных компонент
Корреляционная матрица
  1      -0.95   -0.73   -0.16   -0.82
 -0.95    1       0.76    0.13    0.73
 -0.73    0.76    1       0.27    0.75
 -0.16    0.13    0.27    1       0.42
 -0.82    0.73    0.75    0.42    1

собственное значение
        3.47043
        0.98745
        0.30361

Собственные вектора
 B1             B2           B3
-0.5058        0.2172       -0.4024       ip_flags
```

```
 0.4948      -0.2642       0.288       ip_ttl
 0.4712      -0.0227      -0.8546      tcp_winsize
 0.1916       0.9274       0.146       tcp_data_len
 0.4906       0.1499       0.0585      tcp_flow_dir

Ранжирование атрибутов:
0.3059
1
-0.506 ip_flags + 0.495 ip_ttl + 0.491 tcp_flow_dir + 0.471 tcp_winsize + 0.192
tcp_data_len

0.1084
2
0.927 tcp_data_len - 0.264 ip_ttl + 0.217 ip_flags + 0.15 tcp_flow_dir - 0.023
tcp_winsize

0.0477
3
- 0.855 tcp_winsize - 0.402 ip_flags + 0.288 ip_ttl + 0.146 tcp_data_len + 0.059
tcp_flow_dir

Выбранные атрибуты: 1,2,3 : 3
```

Результатом работы метода PCA являются 3 выбранных признака, взамен 5 исходных (см. таблицу 2). Выбранные признаки строятся как линейная комбинация элементов исходных признаков с коэффициентами в виде собственных векторов:

```
-0.506 ip_flags + 0.495 ip_ttl + 0.491 tcp_flow_dir + 0.471 tcp_winsize +
0.192 tcp_data_len

0.927 tcp_data_len - 0.264 ip_ttl + 0.217 ip_flags + 0.15 tcp_flow_dir -
0.023 tcp_winsize

- 0.855 tcp_winsize - 0.402 ip_flags + 0.288 ip_ttl + 0.146 tcp_data_len +
0.059 tcp_flow_dir.
```

Также метод PCA проводит ранжирование на основании индивидуальной оценки признаков: 0.3059, 0.1084, 0.0477.

2.2.2. Метод ранжирования атрибутов на основе информационного выигрыша (Information Gain Attribute Ranking, InfoGain)

Это один из самых простых (и самый быстрый) методов ранжирования признаков. Если A – признак, и C – класс, уравнения (5) и (6) вычисляют энтропию класса до и после наблюдения признака.

$$H(C) = -\sum_{c \in C} p(c) \log_2 p(c) \tag{5}$$

$$H(C \mid A) = -\sum_{a \in A} p(a) \sum_{c \in C} p(c \mid a) \log_2 p(c \mid a) \tag{6}$$

Величина, на которую энтропии класса уменьшается, отражает дополнительную информацию о классе, обеспеченную атрибутом и называется информационным выигрышем [40]. Каждому атрибут A_i назначается оценка, основанная на информационном выигрыше между самим атрибутом и классом:

$$\begin{aligned} IG_i &= H(C) - H(C \mid A_i) \\ IG_i &= H(A_i) - H(A_i \mid C) \\ IG_i &= H(A_i) + H(C) - H(A_i, C) \end{aligned} \tag{7}$$

Результат выбора атрибутов в Weka с использованием 10-тислойной перекрестной проверки (подробнее о методе проверки см. п. 2.4.5):

```
среднее значение качества    среднее значение ранжирования        атрибут
    1.379 +- 0.002                    1   +- 0              4 tcp_data_len
    0.815 +- 0.003                    2   +- 0              3 tcp_winsize
    0.457 +- 0.001                    3   +- 0              2 ip_ttl
    0.198 +- 0.001                    4   +- 0              1 ip_flags
    0.085 +- 0.001                    5   +- 0              5 tcp_flow_dir
```

Результатом работы метода InfoGain стало ранжирование признаков по их значимости. Наилучшее значение ($1{,}379 \pm 0{,}002$) при оценке качества (*merit*) показал четвертый признак: tcp_data_len.

2.2.3. Метод выбора признаков, основанный на корреляции (A Correlation-based Feature Selector, CFS)

Этот метод представляет собой простой алгоритм фильтрации, который ранжирует подмножества признаков. Основой алгоритма CFS является эвристика для оценки ценности или качества подмножества признаков. Эта эвристика принимает во внимание полезность отдельных признаков для предсказания метки класса в соответствии с уровнем корреляции внутри них. Предположение, на котором основывается эвристика, звучит следующим образом: хорошие подмножества признаков содержат признаки сильно коррелирующие с классом, однако некоррелирующие друг с другом.
Уравнение (8) формализует эвристику:

$$Merit_s = \frac{k\overline{r}_{cf}}{\sqrt{k + k(k-1)\overline{r}_{ff}}}, \qquad (8)$$

где $Merit_s$ – эвристическое “качество” подмножества признаков S, включающих k признаков, $\overline{r}_{cf}$ - среднее значение корреляции признак-класс, $\overline{r}_{ff}$ - среднее значение внутренней корреляции признак-признак. Уравнение (8) – это фактически корреляция Пирсона, где все переменные нормализованы [34].

Результат выбора атрибутов с помощью метода CFS в Weka:

```
Метод поиска:
	Первый лучший.
	Стартовое множество: нет атрибутов
	Направление поиска: вперед
	Выходить из поиска после 5 затруднений
	Общее число оцененных подмножеств: 20
	Качество лучшего найденного подмножества:  0.511

Оценка подмножества атрибутов (обучение с учителем, класс (номинальный)): 6
app_proto_name):
	CFS оценка подмножества
	Включая локальное предсказание атрибутов

Выбранные атрибуты: 3,4 : 2
                     tcp_winsize
                     tcp_data_len
```

Метод CFS выбрал в качестве признаков tcp_winsize и tcp_data_len, то есть число исходных признаков сократилось с 5 до 2.

2.2.4. Оберточный метод выбора признаков (Wrapper)

В оберточном методе алгоритм выделения подмножества признаков существует как обертка вокруг индукционного алгоритма. Алгоритм выделения подмножества признаков сопровождается поиском хорошего подмножества, используя этот же индукционный алгоритм как часть функции, оценивающей подмножества признаков. Идея оберточного метода, показанная схематично на рисунке 15, раскрывается далее.

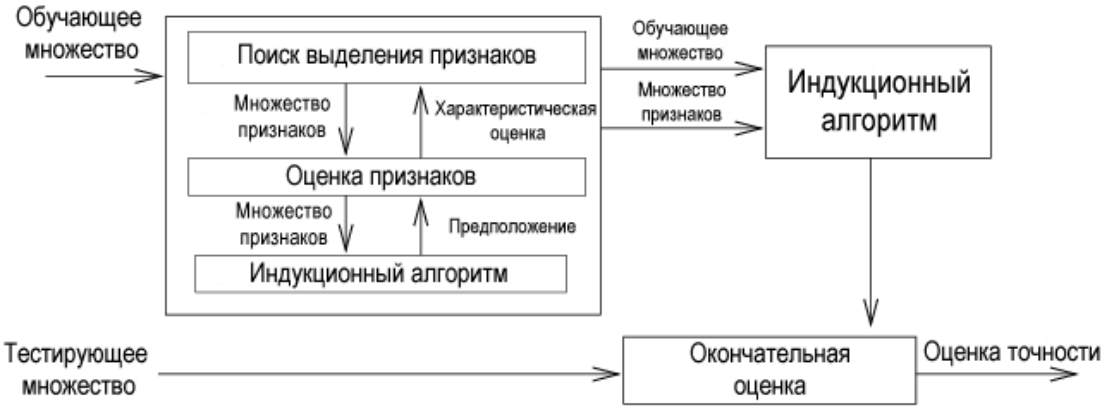

Рисунок 15. Схема оберточного метода выделения признаков

Индукционный алгоритм рассматривается как черный ящик, он запускается на множестве данных, разделяемых обычно на внутреннее обучающее и тестирующее множества (о методах разделения множества см. п. 2.4.5) с различным набором признаков, перемещенных из данных. Конечное множество признаков с наибольшей оценкой выбирается в качестве заключительного множества, на котором запускается индукционный алгоритм. Результирующий классификатор затем оценивается на независимом тестовом множестве, которое не использовалось во время поиска [41].

Результат выбора атрибутов методом Wrapper для индукционного алгоритма “наивный” Байес в Weka:

```
Метод поиска:
	Первый лучший.
	Стартовое множество: нет атрибутов
	Направление поиска: вперед
	Выходить из поиска после 5 затруднений
	Общее число оцененных подмножеств: 17
	Качество лучшего найденного подмножества: 0.335

Оценка подмножества атрибутов (обучение с учителем, класс (номинал): 6
app_proto_name):
```

```
        Оберточная оценка подмножества
        Обучающая схема: weka.classifiers.bayes.NaiveBayesSimple
        Оценка точности: ошибка классификации
        Число слоев для оценки точности: 5
Выбранные атрибуты : 1,3,4,5 : 4
                      ip_flags
                      tcp_winsize
                      tcp_data_len
                      tcp_flow_dir
```

Метод Wrapper оставил 4 признака из 5 исходных (см. таблицу 2): ip_flags, tcp_winsize, tcp_data_len, tcp_flow_dir.

В таблице 3 представлены сводные данные, полученные в результате выполнения методов PCA, InfoGain, CFS и Wrapper.

Таблица 3. Сводная таблица результатов выполнения методов выбора атрибутов

Название метода	Выбранные атрибуты	Комментарии
PCA	-0.506 ip_flags + 0.495 ip_ttl + 0.491 tcp_flow_dir + 0.471 tcp_winsize + 0.192 tcp_data_len 0.927 tcp_data_len - 0.264 ip_ttl + 0.217 ip_flags + 0.15 tcp_flow_dir - 0.023 tcp_winsize - 0.855 tcp_winsize - 0.402 ip_flags + 0.288 ip_ttl + 0.146 tcp_data_len + 0.059 tcp_flow_dir.	Выбранные признаки строятся как линейная комбинация элементов исходных признаков с коэффициентами в виде собственных векторов.
InfoGain	1. tcp_data_len 2. tcp_winsize 3. ip_ttl 4. ip_flags 5. tcp_flow_dir	Результатом работы метода является список признаков, ранжированных по их значимости.
CFS	tcp_winsize tcp_data_len	
Wrapper	ip_flags tcp_winsize tcp_data_len tcp_flow_dir	Результаты метода Wrapper в отличие от остальных рассмотренных методов, зависят от индукционного алгоритма.

2.2.5. Сравнение методов выделения признаков

Для сравнения методов выбора признаков воспользуемся интерфейсом Weka Исследователь. В качестве входных значений для этого интерфейса укажем три обучающих множества, алгоритм для сравнения – “наивный” Байес (более подробно о методе см. п. 2.4.1.) и методы выбора признаков рассмотренные в п. 2.2.1-2.2.4. Задача эксперимента – сравнить точность выполнения алгоритма классификации до сокращения числа признаков и после сокращения, для каждого из методов выбора. Фактически, происходит многократное выполнение перекрестной проверки для каждого из методов и на основе полученных результатов формирует выходной набор данных.

Количество проходов перекрестной проверки устанавливаем равным 10 (наиболее часто применяется на практике). Сравнение методов производится по полю “Percent Correct”, то есть по количеству правильно классифицированных данных в процентах.

Weka предоставляет возможность выбрать один из вариантов отображения результатов: сравнение с выбранным методом (выбирается один из методов, с которым сравниваются все остальные), Summary (сравнение каждый с каждым) и Ranking (сравнение “побед” и “поражений” каждого из алгоритмов).

Под названием метода выбора признаков далее подразумевается метод “наивного” Байеса, к которому был применен соответствующий метод сокращения признаков.

Воспользуемся методом сравнения с одним из выбранных алгоритмов, в данном случае – “наивным” Байесом с исходным набором признаков. В результате проведения вычислений формируется таблица, в строках которой указываются методы выбора признаков, а в рядах – входные наборы данных:

Дамп (1)	NB	(2) PCA	(3) InfoGain	(4) CFS	(5) Wrapper
TS1 (10)	68.10(1.21)	66.16(0.93)*	68.10(1.21)	59.28(0.39)*	68.20(1.26)
TS2 (10)	68.10(3.16)	66.53(0.57)v	56.45(3.16)	58.09(0.73)	58.94(0.76)
TS3 (10)	66.13(1.00)	70.37(0.78)v	66.13(1.00)	59.05(1.06)*	66.53(0.97)v
	(v/ /*)	(2/0/1)	(0/3/0)	(0/1/2)	(1/2/0)

В последней строке под чертой указывается, что алгоритм статистически лучше/нейтрален/хуже других в данном исследовании. Символом “v” обозначено, что точность метода выше по сравнению с “наивным” Байесом без выбора атрибутов (столбец для сравнения выбирается вручную). В нашем случае, к примеру, для первого обучающего

набора процент верно классифицированных экземпляров данных выглядит следующим образом (в скобках указано стандартное отклонение):

- NaiveBayes (без сокращения числа атрибутов) – 68.10 (1.21)
- NaiveBayes (предварительно применен PCA) – 66.16 (0.93)
- NaiveBayes (предварительно применен InfoGain) – 68.10 (1.21)
- NaiveBayes (предварительно применен CFS) – 59.28 (0.39)
- NaiveBayes (предварительно применен Wrapper) – 68.20 (1.26)

На рисунке 16 показана диаграмма зависимости точности классификации от результатов выполнения метода “наивного” Байеса с исходным набором признаков и методами сокращения признаков для Training-Set-1, 2, 3.

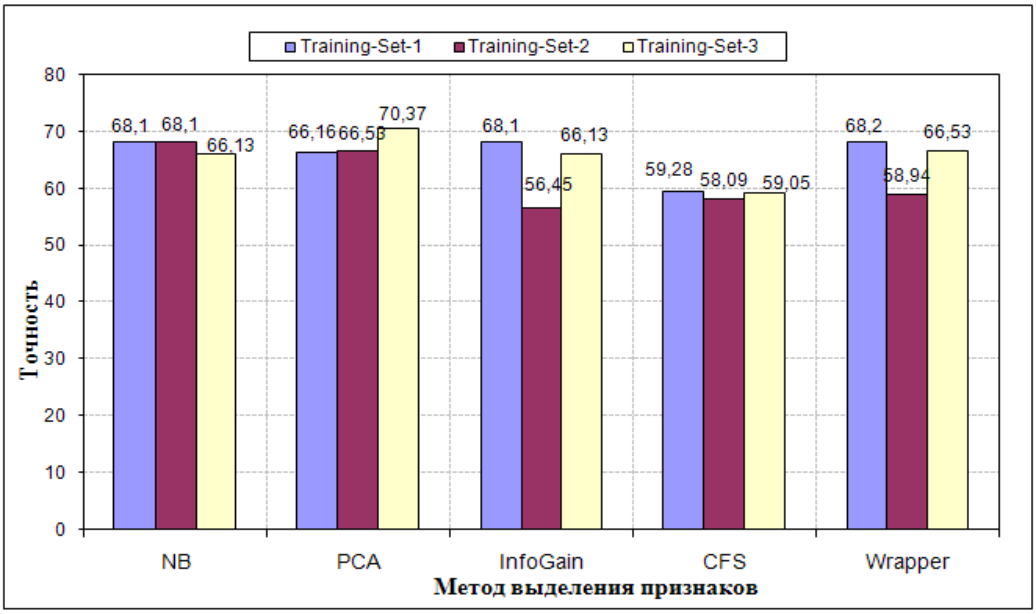

Рисунок 16. Диаграмма зависимости точности классификации от результатов выполнения метода “наивного” Байеса с исходным набором признаков и методами сокращения признаков для Training-Set-1, 2, 3

Из рисунка 16 видно, что наилучшие результаты среди методов выделения признаков для алгоритма “наивного” Байеса показал PCA.

Для наглядности все рассматриваемые методы выделения признаков показаны на одной диаграмме, при этом необходимо учитывать особые свойства (и улучшенные результаты) метода Wrapper, так как это единственный метод, который зависит от индукционного

алгоритма (в данном случае “наивного” Байеса). В дальнейшем это замечание относится ко всем диаграммам сравнения методов выделения признаков.

Рассмотрим результаты при выборе Ranking (сравнение “побед” и “поражений” каждого из алгоритмов). Эта таблица выводит количество наборов данных, в котором “победил” (колонка >) или “проиграл” (колонка <) определенный алгоритм в сравнении с остальными. Колонка > - < показывает разницу “победы” – “поражения” для данного алгоритма. Необходимо отметить, что в данном случае “победа” алгоритма означает, что у него самый высокий процент верно классифицированных экземпляров, что равносильно “победе”, то есть лучшему результату применимости.

Результат вывода:

```
>-<   >   < Результирующее множество
  6   9   3 PCA
  4   6   2 Wrapper
  0   3   3 InfoGain
  0   3   3 NaiveBayes
-10   0  10 Cfs
```

Из таблицы видно, что хуже всего себя показал алгоритм CFS. PCA оказался наиболее применим к входным обучающим наборам данных. Метод Wrapper (с “наивным” Байесом в качестве индукционного алгоритма) занял вторую позицию и показал близкие к PCA результаты.

Повторим проведенный эксперимент на множествах Training-Set-1, 2, 3 для методов классификации OneR и SVM (см. рисунок 17, 18). Более подробно о методах классификации OneR и SVM будет рассказано в п. 2.4.4 и 2.4.3 соответственно. Для метода J48 результаты эксперимента будут не показательны, так как этот метод сам по себе обладает высокой точностью (98-99%).

На рисунках 17, 18 под названием метода классификации (OneR и SVM) подразумевается выполнение этого метода на исходном наборе атрибутов, под названием метода выбора атрибутов (PCA, InfoGain, CFS, Wrapper) подразумевается выполнение соответствующего метода классификации с предварительным сокращением атрибутов одним из методов выбора.

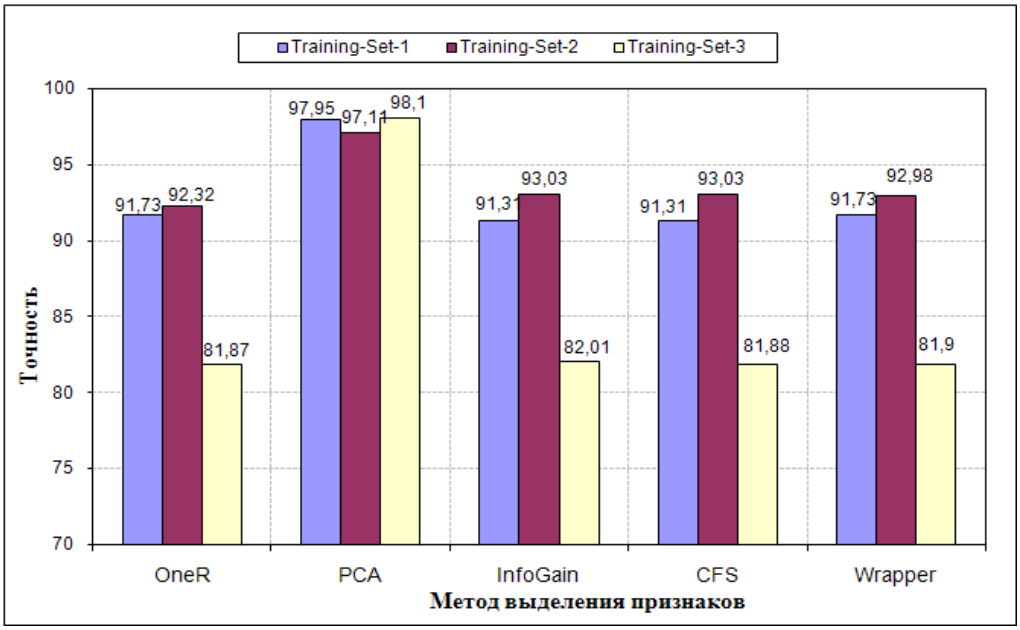

Рисунок 17. Диаграмма зависимости точности классификации методом 1R от метода выделения признаков для Training-Set-1, 2, 3

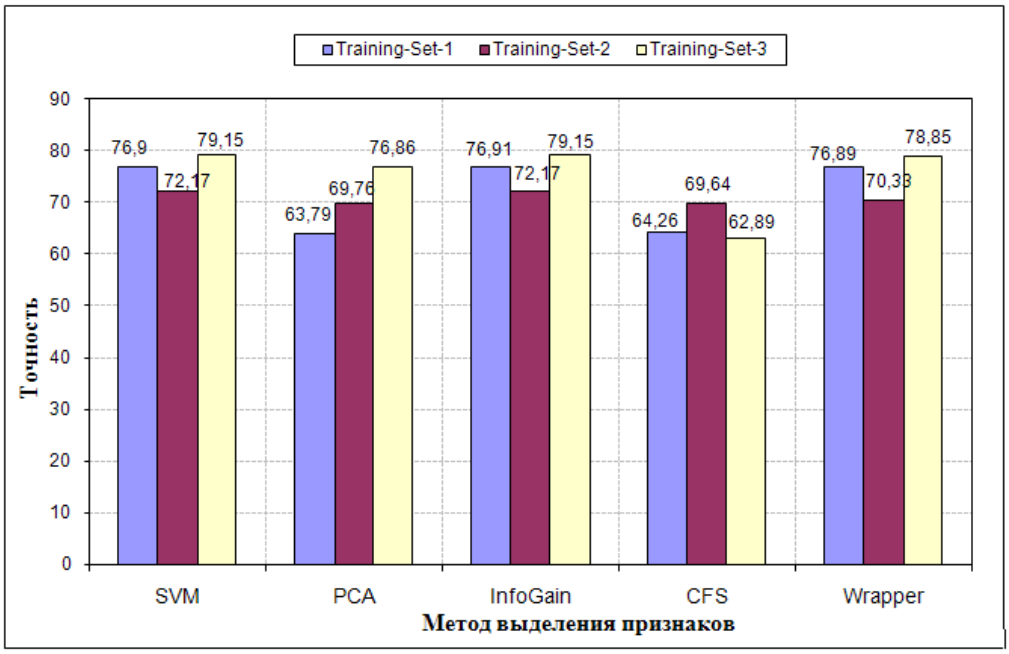

Рисунок 18. Диаграмма зависимости точности классификации методом SVM от метода выделения признаков для Training-Set-1, 2, 3

В результате исследования методов сокращения признаков для алгоритмов OneR и SVM получили, что для алгоритма OneR очень хорошо проявил себя метод PCA, для SVM – на результаты классификации методы выбора признаков оказали слабое влияние: InfoGain показал нейтральный результат (значения полностью совпали с классификацией без применения методов выбора признаков), остальные методы ухудшили показатели.

2.3. Кластеризация

Технологии классификации используют предопределенные классы обучающих экземпляров. В отличие от этого, методы кластеризации не предусматривают таких сведений; взамен, они обнаруживают естественные кластеры (группы) в данных, используя эвристики [42].

Кластеризация сосредоточена на поиске шаблонов во входных данных. Она объединяет экземпляры с похожими свойствами (определяются конкретным подходом к измерению расстояния, например, Евклидовым расстоянием) в группы. Определенные таким образом группы могут быть исключительными, когда любой экземпляр принадлежит только одной группе; или могут быть перекрывающимися, когда один экземпляр может относиться к нескольким группам; они также могут быть вероятностными, когда экземпляр принадлежит группе с определенной вероятностью. Группы также могут быть иерархическими, в этом случае, происходит разделение экземпляров по группам, начиная с верхнего уровня, и далее каждая из этих групп улучшается, вплоть до уровня отдельных экземпляров [43].

Существует три основных метода кластеризации: классический алгоритм k-средних, инкрементная кластеризация и кластеризация, основанный на вероятности. Классический алгоритм k-средних формирует кластеры в числовые области, разбивая экземпляры на отдельные кластеры, в то время как, инкрементная кластеризация производит иерархическую группировку экземпляров. Методы, основанные на вероятности, назначают отдельным классам вероятности [43].

Для исследования методов кластеризации были выбраны алгоритмы EM и k-средних.

2.3.1. Метод максимизации ожидания (Expectation Maximization, EM)

В данном методе предполагается, что кроме известных нам из наших данных величин существуют еще и неизвестные нам, относящиеся к распределению по кластерам. То есть фактически эти неизвестные “создают” кластер, а мы наблюдаем только результат их деятельности. И именно эти неизвестные мы и стараемся максимально точно оценить.

Алгоритм использует широко известный метод максимизации ожиданий (Expectation Maximization). В наиболее простом случае предполагается, что кластер – это результаты наблюдения, распределенные нормально. Тогда для их характеристики можно

применять многомерную функцию Гаусса (многомерное распределение Гаусса). И тогда основная задача – это определить, к какому из распределений принадлежит каждая конкретная точка, оценив параметры этих распределений исходя из реального распределения точек.

Как правило, EM-алгоритм применяется при решении задач двух типов. К первому типу можно отнести статистические задачи, связанные с анализом действительно неполных данных, когда некоторые статистические данные отсутствуют в силу каких-либо причин. К другому типу задач можно отнести статистические задачи, в которых функция правдоподобия имеет вид, не допускающий удобных аналитических методов исследования, но допускающий серьезные упрощения, если в задачу ввести дополнительные “наблюдаемые” (“отсутствующие”, скрытые) величины. Примерами прикладных задач второго типа являются задачи распознавания образов, реконструкции изображений. Математическую суть этих прикладных задач составляют задачи кластерного анализа, классификации и разделения смесей вероятностных распределений.

На основании перекрестной проверки EM-алгоритм может принимать решение о числе кластеров, алгоритм принятия решения выглядит следующим образом:

1. число кластеров устанавливается в 1;
2. обучающее множество случайным образом разбивается на 10 слоев;
3. EM-алгоритм выполняется 10 раз, используя 10-тислойную перекрестную проверку;
4. логарифмическое правдоподобие усредняется для всех 10-ти результатов;
5. если логарифмическое правдоподобие увеличивается, число кластеров увеличивается на 1 и выполнение продолжается с пункта 2 [28].

Результат выполнения EM-алгоритма для обучающего набора Training-Set-3:

```
Число кластеров: 4
                  Кластер
Атрибут              0          1          2          3
                   (0.09)     (0.33)     (0.22)     (0.36)
============================================================
ip_flags
  среднее           1.9724          2          0          0
  станд.откл.       0.2332     0.9869     0.9869     0.9869
ip_ttl
  среднее            57.57    67.3889   127.0193        128
  станд.откл.       1.2237     14.332     8.3456    32.2443
tcp_winsize
  среднее       44795.9004 19924.5688 63470.8279 64239.9999
  станд.откл.   27253.7501 23449.6862  6549.3814     0.0112
```

```
tcp_data_len
  среднее        1281.8472    125.368  1431.6439     77.534
  станд.откл.     400.2275   233.8358    102.387   116.0503
tcp_flow_dir
  среднее                2          1          2          2
  станд.откл.       0.0015      0.469      0.469      0.469

Экземпляров кластеризовано
0       2054 (  9%)
1       7295 ( 33%)
2       4786 ( 22%)
3       8012 ( 36%)

Логарифмическое правдоподобие: -17.59832

Атрибут класса для оценки: app_proto_name
Классы и кластеры:
    0    1    2    3
    0  814 4753  689 | http
    0 2962    0 3559 | ftp-c
    0 3075    0 3763 | smtp
 2054  444   33    1 | trojans

Кластер 0 <-- trojans
Кластер 1 <-- ftp-c
Кластер 2 <-- http
Кластер 3 <-- smtp

Неправильно кластеризованных экземпляров :      8615.0       38.8992 %
```

Проанализируем полученные результаты. С помощью перекрестной проверки было определено количество кластеров – 4. Этим значением мы воспользуемся в дальнейшем при задании входных параметров (числа кластеров) для алгоритма k-средних.

Для каждого из классов EM-алгоритм указал среднее значение и стандартное отклонение для всех атрибутов.

Помимо автоматического выделения групп (кластеров) в наборе сетевого трафика, Weka также позволяет оценить точность выделения. Для оценки кластеризации использовался набор заранее классифицированного трафика Training-Set-3 (app_proto_name – поле, в котором указан тип сетевого пакета). На основании предварительной классификации и кластеризации, проведенной EM-алгоритмом, была сформирована сравнительная таблица 4.

Таблица 4. Отношение количества кластеризованных сетевых пакетов с помощью алгоритма EM и предварительно классифицированных пакетов для обучающего набора Training-Set-3

Название заранее классифицированных типов трафика	Группы, выделенные EM-алгоритмом, в скобках – типы трафика, присвоенные EM-алгоритмом			
	Кластер 0 (Trojans)	Кластер 1 (FTP-C)	Кластер 2 (HTTP)	Кластер 3 (SMTP)
HTTP	0	814	4753	689
FTP-C	0	2962	0	3559
SMTP	0	3075	0	3763
Trojans	2054	444	33	1

Из таблицы 4 можно заметить, что к кластеру 2 было отнесено большинство сетевых пакетов (4753), принадлежащих HTTP-протоколу, при этом часть пакетов (1503) были кластеризованы ошибочно. Большинство сетевых пакетов FTP-C протокола (3559) ошибочно было отнесено к кластеру 3. Аналогично, трафик SMTP-протокола был ошибочно отнесен к кластеру 1: EM-алгоритм по причине небольшого числа признаков не смог хорошо разделить свойства кластеров 1 и 3. Наилучший результат показал трафик от ВПО (Trojans): 477 ошибочно кластеризованных сетевых пакетов из 2531.

На рисунке 19 с помощью средств визуализации Weka показано более наглядное по сравнению с таблицей графическое соотношение между кластерами и заранее классифицированными группами. В скобках указаны типы трафика, присвоенные EM-алгоритмом. Прямоугольниками отмечены места, где сетевые пакеты были кластеризованы правильно.

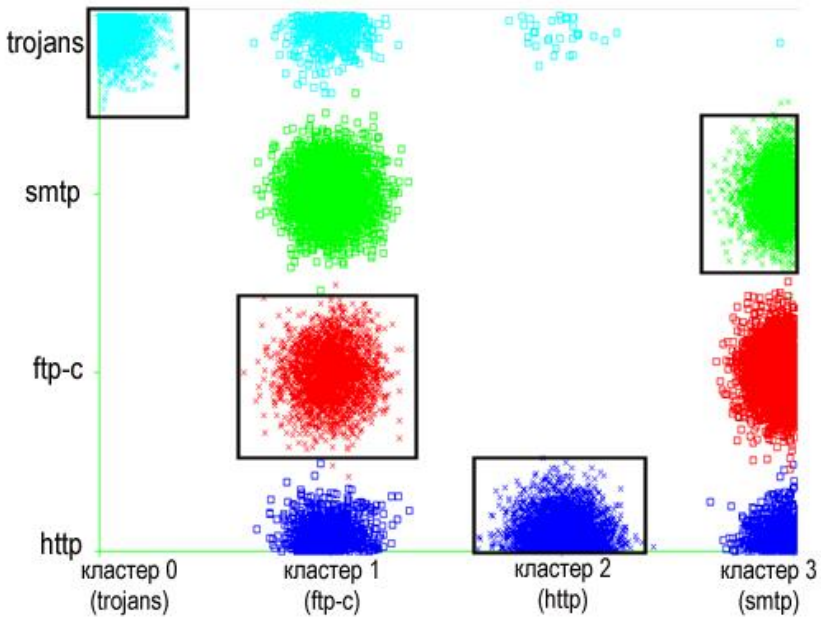

Рисунок 19. График зависимости кластерного разбиения с помощью EM-алгоритма от предварительно классифицированных сетевых пакетов (цветами обозначены типы трафика: HTTP – синий, FTP-C – красный, SMTP – зеленый, Trojans – голубой)

Из-за небольшого количества выбранных исходных признаков (см. таблицу 2) алгоритм показал достаточно высокий процент ошибочно кластеризованных пакетов – 38.9%.

2.3.2. Метод k-средних

Алгоритм k-средних, — метод жесткой кластеризации. Это значит, что точка данных может принадлежать только одному кластеру и для принадлежности каждой точки данных этому кластеру. Кластеризация методом k-средних — хорошо известный метод определения принадлежности элементов кластерам с помощью минимизации разницы между элементами кластера и максимизации расстояния между кластерами. Слово “средние” в названии метода относится к центроидам кластеров. Центроид — точка данных, которая выбирается произвольно, а затем итеративно уточняется, пока не начинает представлять собой истинное среднее всех точек данных кластера. “k” означает произвольное количество точек, используемых для формирования начальных значений процесса кластеризации. Алгоритм k-средних вычисляет квадраты евклидовых расстояний между записями данных в кластере и вектор, представляющий собой среднее данного кластера. Метод сходится, выдавая окончательный набор из k кластеров, когда упомянутая сумма минимизирована. Алгоритм k-средних назначает каждой точке данных ровно один кластер и не допускает неопределенности в принадлежности точки кластеру.

Принадлежность к классу выражается расстоянием от центроида [44]. Метод k-средних хорошо работает, если данные по своей естественной природе делятся на компактные, примерно сферические группы.

Для метода k-средних необходимо предварительно указать число кластеров, на которое будет разбиваться исследуемое множество. Воспользуемся оценкой числа кластеров, проведенной методом кластеризации EM (см. п. 2.3.1).

Далее приведен результат применения метода k-средних (предварительно заданное число кластеров равно четырем) к обучающему множеству Training-Set-3:

```
Число итераций: 6
Внутри кластерная сумма ошибки в квадрате: 3806.812698553883
Центройды кластера:
                        Кластер#
Атрибут   Всего данных       0             1             2             3
            (22147)       (12789)        (5638)        (2118)        (1602)
==============================================================================
ip_flags    0.8387             0             2         1.933             2
            +/-0.9869        +/-0          +/-0     +/-0.3601          +/-0
ip_ttl      101.3666      127.8799       63.9931        57.449       79.3009
           +/-32.2443      +/-2.77     +/-0.2032     +/-1.1501    +/-27.3058
tcp_winsize 47769.7495   64130.4033     7679.2595    44207.9674    62961.7821
        +/-25781.1796  +/-2527.6053  +/-4813.7173 +/-27410.6984  +/-4697.7173
tcp_data_len 503.4492     585.6226      117.7946     1299.6468      152.0506
          +/-630.9101   +/-665.3253   +/-226.2395    +/-373.307   +/-256.7422
tcp_flow_dir  1.6734         2        1.0012            2             1
            +/-0.469       +/-0     +/-0.0352          +/-0          +/-0
Кластеризовано экземпляров:
0      12789 ( 58%)
1       5638 ( 25%)
2       2118 ( 10%)
3       1602 (  7%)

Атрибут класса для оценки: app_proto_name
Классы и кластеры:
    0    1    2    3
 5443  749    0   64 | http
 3559 1807    0 1155 | ftp-c
 3763 3075    0    0 | smtp
   24    7 2118  383 | trojans

Кластер 0 <-- http
Кластер 1 <-- smtp
Кластер 2 <-- trojans
Кластер 3 <-- ftp-c
```

```
Неправильно кластеризованных экземпляров :    10356.0       46.7603 %
```

Для оценки кластеризации использовался набор заранее классифицированного трафика Training-Set-3 (app_proto_name – поле, в котором указан тип сетевого пакета). На основании предварительной классификации и кластеризации, проведенной алгоритмом k-средних, была сформирована сравнительная таблица 5.

Таблица 5. Отношение количества кластеризованных сетевых пакетов с помощью алгоритма k-средних и предварительно классифицированных пакетов для обучающего набора Training-Set-3

Название заранее классифицированных типов трафика	Группы, выделенные алгоритмом k-средних, в скобках – типы трафика, присвоенные алгоритмом k-средних			
	Кластер 0 (HTTP)	Кластер 1 (SMTP)	Кластер 2 (Trojans)	Кластер 3 (FTP-C)
HTTP	5443	749	0	64
FTP-C	3559	1807	0	1155
SMTP	3763	3075	0	0
Trojans	24	7	2118	383

Из таблицы 5 можно заметить, что к кластеру 1 было отнесено большинство сетевых пакетов (5443), принадлежащих HTTP-протоколу, при этом часть пакетов (813) были кластеризованы ошибочно. Большинство сетевых пакетов FTP-C протокола (5366) ошибочно было отнесено к кластерам 0 и 1. Трафик SMTP-протокола был ошибочно отнесен к кластеру 0 (3763 ошибочно кластеризованных сетевых пакетов). Алгоритм k-средних показал худшие результаты по сравнению с EM-алгоритмом, так как не смог выделить свойства кластеров 0, 1 и 3. Наилучший результат (как и в методе EM) показал трафик от ВПО (Trojans): 414 ошибочно кластеризованных сетевых пакетов из 2531, то есть хорошие показатели кластеризации сетевых пакетов, сгенерированных ВПО, зависят в большей степени от ярко выраженных отличительных признаков самого трафика, а не от выбранного метода кластеризации.

На рисунке 20 с помощью средств визуализации Weka показано более наглядное по сравнению с таблицей графическое соотношение между кластерами и заранее классифицированными группами. В скобках указаны типы трафика, присвоенные алгоритмом k-средних. Прямоугольниками отмечены места, где сетевые пакеты были кластеризованы правильно.

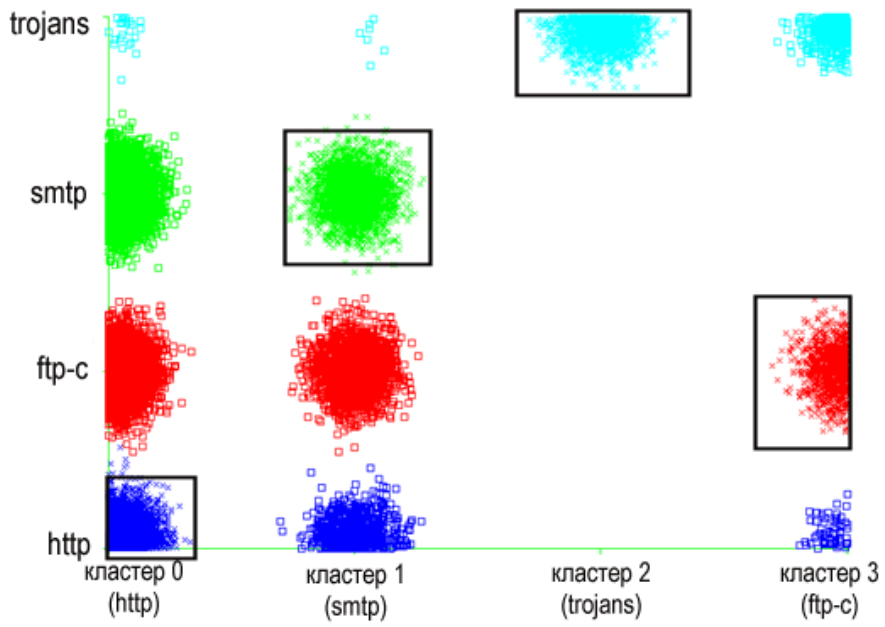

Рисунок 20. График зависимости кластерного разбиения с помощью алгоритма k-средних от предварительно классифицированных сетевых пакетов (цветами обозначены типы трафика: HTTP – синий, FTP-C – красный, SMTP – зеленый, Trojans – голубой)

Алгоритм k-средних из-за небольшого числа входных признаков показал очень плохой процент ошибочно классифицированных пакетов – 46.8%.

2.3.3. Оценка обучающих алгоритмов без учителя

С учетом набора данных, алгоритм кластеризации всегда производит разделение в зависимости от его собственной найденной структуры в пределах данных. Разные методы могут приводить к различным наборам кластеров и различные параметры одного и того же алгоритма могут повлиять на конечные результаты [45, 46].

Таким образом, важно иметь эффективные оценочные стандарты и критерии предоставления пользователям уверенности в результатах, произведенных отдельным алгоритмом или компонентами различных алгоритмов [47]. Критерий может помочь ответить на вопросы о том, сколько кластеров скрыто в данных, какое оптимальное количество кластеров [46], выразительны ли результирующие кластеры или они только артефакты алгоритмов [47], как один из алгоритмов сравнить с другими: на сколько просто они используются, сколько быстродействия должно быть задействовано [45], каково качество внутрикластерного разделения, как хорошо провести разделение на

кластеры, каковы затраты на маркировку кластеров и, каковы требования с точки зрения компьютерного вычисления и хранения.

В [46] выделяют три подхода для исследования обоснованности кластеров: внешний критерий, внутренний критерий и относительный критерий. Первые два подхода основаны на проверке статистических предположений. Внешний критерий основан на некоторых предопределенных структурах, также известных как априорная информация для данных, и используемая как стандарт для сравнения и подтверждения результатов кластеризации [47]. Внутренний критерий оценивает результат кластеризации алгоритма, основанного на рассмотрении внутренней структуры, унаследованной от набора данных. Относительный критерий обращает особое внимание на поиск наилучших схем кластеризации таких, что алгоритм кластеризации можно определить по известным предположениям и параметрам. Основная идея состоит в том, чтобы оценить структуру кластеризации, сравнивая ее с некоторым значением, используя один и тот же алгоритм с различными значениями параметров [48]. Больше деталей можно найти в [46, 47, 45, 43].

2.4. Классификация

При обучении с учителем создаются структуры, которые позволяют решить задачу классификации новых экземпляров в заранее классифицированные классы [49]. Машинное обучение обеспечивает сбор простых предварительно классифицированных экземпляров. Результатом такого процесса обучения является классификационная модель, которая строится на основе изучения и обобщения предоставленных экземпляров.

Обучение с учителем уделяет основное внимание моделированию входных/выходных отношений. Его цель состоит в выявлении отображения входных особенностей на выходной класс. Обученные знания могут быть представлены в виде блок-схемы, дерева решений, классификационных правил, и т.д., которые могут быть использованы в дальнейшем для классификации новых экземпляров.

Существует две основные фазы (шаги) в обучении с учителем:

- *обучение*: фаза, на которой исследуются подготовленные данные (называемые обучающим множеством данных) и создается (строится) классификационная модель;
- *тестирование* (или классифицирование): модель, которая была построена на этапе обучения, используется для классификации новых экземпляров.

Модель, созданная во время обучения, улучшается, если мы попутно предусматриваем примеры экземпляров, принадлежащих интересующему нас классу, и

экземпляров, которые не принадлежат этому классу. В дальнейшем это улучшит способность модели определять экземпляры, принадлежащие интересующему нас классу [42, 43].

Для исследования методов классификации были выбраны алгоритмы Naïve Bayes, J4.8, SVM (SMO в Weka) и OneR.

На рисунке 21 показано главное окно Weka Explorer с открытым набором данных.

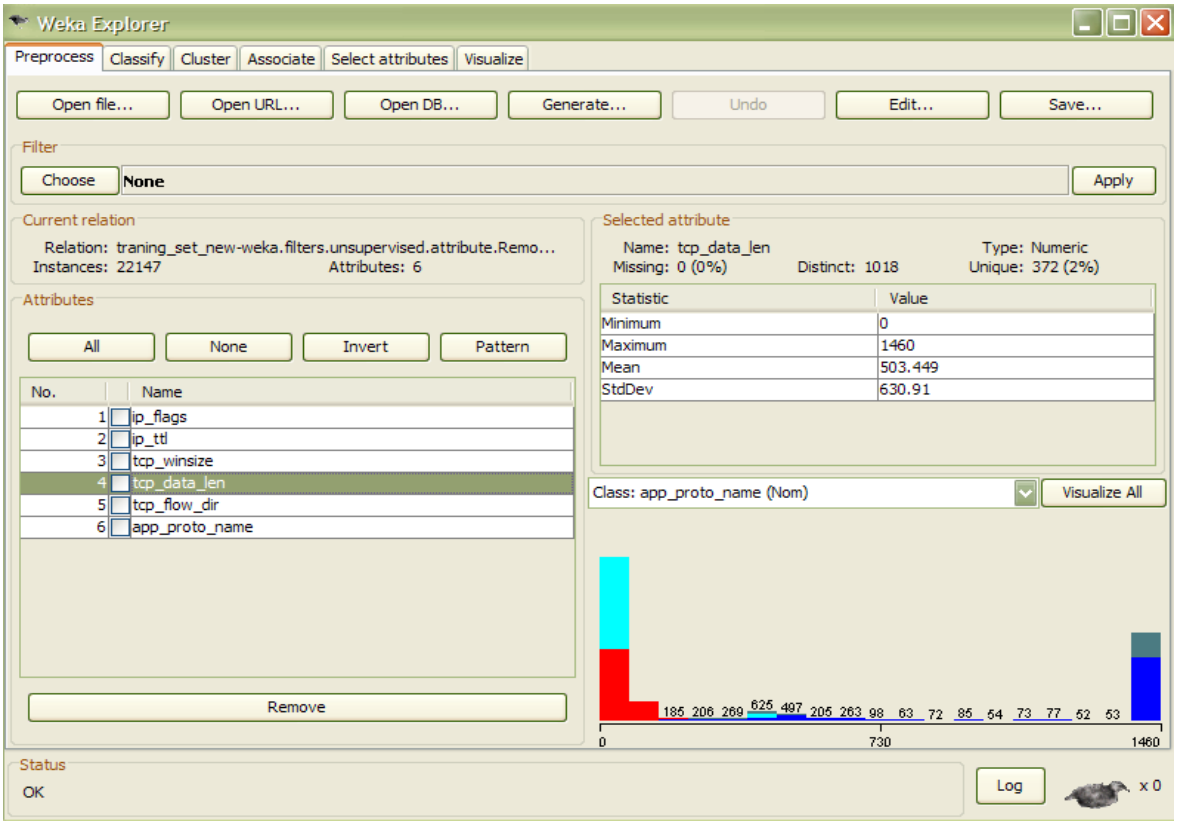

Рисунок 21. Главное окно Weka Explorer

Было выбрано несколько методов проверки точности классификации. В качестве первого метода проверки – метод перекрестной проверки (см. п. 2.4.5). В качестве второго метода проверки использовались подготовленные тестирующие множества (Test-Set) из таблицы 1.

2.4.1. Метод Naïve Bayes

Нередко для классификации необходимо рассмотреть несколько независимых переменных. Такую классификацию позволяет выполнять алгоритм “наивного” Байеса, использующий формулу Байеса для расчета вероятности. Название “наивный” происходит от наивного предположения, что все рассматриваемые переменные независимы друг от друга. В действительности это не всегда так, но на практике данный алгоритм находит применение.

Вероятность того, что некоторый объект i_j относится к классу c_r (т. е. $y = c_r$), обозначим как $P(y = c_r)$. Событие, соответствующее равенству независимых переменных определенным значениям, обозначим как E, а вероятность его наступления $P(E)$. Идея алгоритма заключается в расчете условной вероятности принадлежности объекта к c_r при равенстве его независимых переменных определенным значениям.

После проведения обучения для множества Training-Set-3 методом “наивного” Байеса с использованием перекрестной проверки были получены следующие результаты:

```
Класс http: P(C) = 0.28247032
Атрибут ip_flags Среднее: 0.25991049          Стандартное отклонение: 0.67256213
Атрибут ip_ttl Среднее: 119.68286445          Стандартное отклонение: 21.5219881
Атрибут tcp_winsize Среднее: 57342.87100384    Стандартное отклонение: 18637.06919303
Атрибут tcp_data_len Среднее: 1217.49728261    Стандартное отклонение: 413.50686617
Атрибут tcp_flow_dir Среднее: 1.87004476       Стандартное отклонение: 0.33628106

Класс ftp-c: P(C) = 0.29443366
Атрибут ip_flags Среднее: 0.90844962           Стандартное отклонение: 0.99587681
Атрибут ip_ttl Среднее: 98.92961202            Стандартное отклонение: 31.86805787
Атрибут tcp_winsize Среднее: 49173.55006901    Стандартное отклонение: 23961.33622285
Атрибут tcp_data_len Среднее: 42.25486889      Стандартное отклонение: 46.1461409
Атрибут tcp_flow_dir Среднее: 1.54577519       Стандартное отклонение: 0.4979384

Класс smtp: P(C) = 0.30874453
Атрибут ip_flags Среднее: 0.89938579           Стандартное отклонение: 0.99499827
Атрибут ip_ttl Среднее: 99.21965487            Стандартное отклонение: 31.83994472
Атрибут tcp_winsize Среднее: 37977.93506873    Стандартное отклонение: 29053.94955486
Атрибут tcp_data_len Среднее: 44.92161451      Стандартное отклонение: 86.2376941
Атрибут tcp_flow_dir Среднее: 1.55030711       Стандартное отклонение: 0.49749914

Класс trojans: P(C) = 0.1143515
Атрибут ip_flags Среднее: 1.92496051           Стандартное отклонение: 0.38013831
Атрибут ip_ttl Среднее: 68.18562401            Стандартное отклонение: 25.28647993
Атрибут tcp_winsize Среднее: 46945.41943128    Стандартное отклонение: 26585.37544681
Атрибут tcp_data_len Среднее: 1165.28791469    Стандартное отклонение: 469.28797967
```

```
Атрибут tcp_flow_dir Среднее: 1.84873618       Стандартное отклонение: 0.35837662

Время, затраченное на построение модели: 0.11 секунд

=== Многослойная перекрестная проверка ===
=== Итог ===
Корректно классифицированных экземпляров       14645               66.1263 %
Некорректно классифицированных экземпляров      7502               33.8737 %
Средняя абсолютная ошибка                      0.187
Корень из среднеквадратической ошибки         0.3395
Относительная абсолютная ошибка              51.5649 %
Общее количество экземпляров                   22147

=== Детальная точность классов ===

TP Оценка   FP Оценка   Точность     Класс
  0.855       0.006       0.982       http
  0.659       0.246       0.528       ftp-c
  0.399       0.158       0.531       smtp
  0.897        0.059      0.663       trojans

=== Смешанная матрица ===
    a    b    c    d
 5349   57   42  808 |    a = http
   10 4295 2216    0 |    b = ftp-c
    0 3763 2730  345 |    c = smtp
   89   17  155 2271 |    d = trojans
```

Этот метод основывается на анализе данных по всем исходным переменным. Формулируются правила, в условных частях которых сравниваются все независимые переменные с соответствующими возможными значениями. Оценив ошибки, выбирается переменная, для которой ошибка набора минимальна. В результате перекрестной проверки получаем достаточно средний процент верной классификации (66%).

К примеру, к классу HTTP принадлежат пакеты, имеющие следующие средние значения для атрибутов (в скобках указано среднее отклонение):

```
ip_flags = 0.25991049 (0.67256213)
ip_ttl = 119.68286445 (21.5219881)
tcp_winsize = 57342.87100384 (18637.06919303)
tcp_data_len = 1217.49728261 (413.50686617)
tcp_flow_dir = 1.87004476 (0.33628106)
```

Последовательно протестируем метод “наивного” Байеса на подготовленных для тестирования множествах (Test-Set 1-4). Результаты проверки представлены в таблице 6.

Таблица 6. Отношение количества классифицированных с помощью алгоритма “наивного” Байеса и предварительно классифицированных пакетов для обучающего набора Training-Set-3

Название дампа	HTTP		FTP-C		SMTP		Trojans		Ошибки, %
	После	До	После	До	После	До	После	До	
Test-Set-1	2767	3517	38	-	32	-	680	-	21.3
Test-Set-2	40	-	2508	3494	946	-	0	-	28.2
Test-Set-3	0	-	3540	-	2233	6209	436	-	64
Test-Set-4	311	-	123	-	251	-	4981	5666	12

В столбце “После” указано число пакетов, которое было отнесено к классу, соответствующему сетевому протоколу, после классификации, в столбце “До” – число пакетов, предварительно классифицированных. Прочерками отмечены полностью отсутствующие протоколы и типы трафика. В столбце “Ошибки” указано процентное соотношение ошибочно классифицированных пакетов к общему числу пакетов.

На основании таблицы 6 наибольшее число ошибочно классифицированных сетевых пакетов было продемонстрировано на тестовом наборе, содержащем SMTP-трафик и FTP-C трафик (3540 пакетов SMTP трафика ошибочно отнесено к FTP-C классу, 946 пакетов FTP-C трафика ошибочно отнесено к SMTP классу). Схожее совпадение свойств классов, описывающих протоколы SMTP и FTP-C, наблюдалось при выполнении проверки на заранее классифицированном множестве для алгоритма кластеризации EM (см. п. 2.3.1).

2.4.2. Метод J4.8 (C4.5)

Деревья решений, к которым относится метод J4.8 (реализация алгоритма C4.5 восьмой редакции, последней перед выходом коммерческой версии C5.0), представляют собой последовательные иерархические структуры, состоящие из узлов, которые содержат правила, то есть логические конструкции вида “если, то”. Конечными узлами дерева являются “листья”, соответствующие найденным решениям и объединяющие некоторое количество объектов классифицируемой выборки. Аналогично тому, как положение листа

на дереве можно задать, указав ведущую к нему последовательность ветвей, начиная от корня и кончая самой последней веточкой, на которой лист растет.

Есть целый ряд причин, делающих деревья классификации более гибким средством, чем традиционные методы анализа:

- схема *одномерного ветвления*, которая позволяет изучать эффект влияния отдельных переменных и проводить последовательный анализ их вклада;
- отсутствие предварительных предположений о законах распределения данных.

После проведения обучения для множества Training-Set-3 методом J4.8 с использованием перекрестной проверки (подробнее о методе проверки см. п.2.4.5) были получены следующие результаты:

```
J48 отсеченное дерево
------------------
tcp_data_len <= 197
|   tcp_winsize <= 5840
|   |   tcp_data_len <= 27: smtp (2420.0/35.0)
|   |   tcp_data_len > 27
|   |   |   tcp_data_len <= 47: smtp (345.0)
|   |   |   tcp_data_len > 47
|   |   |   |   ip_flags <= 0: trojans (8.0)
|   |   |   |   ip_flags > 0: http (2.0/1.0)
|   tcp_winsize > 5840
|   |   ip_flags <= 0
|   |   |   tcp_data_len <= 46
|   |   |   |   tcp_data_len <= 33
|   |   |   |   |   tcp_data_len <= 30
|   |   |   |   |   |   tcp_data_len <= 27
|   |   |   |   |   |   |   tcp_data_len <= 14
|   |   |   |   |   |   |   |   tcp_data_len <= 7: http (9.0)
|   |   |   |   |   |   |   |   tcp_data_len > 7: smtp (691.0/1.0)
|   |   |   |   |   |   |   tcp_data_len > 14: ftp-c (955.0/6.0)
|   |   |   |   |   |   tcp_data_len > 27: smtp (1350.0)
|   |   |   |   |   tcp_data_len > 30: ftp-c (463.0/1.0)
|   |   |   |   tcp_data_len > 33: smtp (1779.0/56.0)
|   |   |   tcp_data_len > 46: ftp-c (2153.0/63.0)
|   |   ip_flags > 0
|   |   |   ip_ttl <= 60: trojans (26.0)
|   |   |   ip_ttl > 60: ftp-c (2942.0/1.0)
tcp_data_len > 197
|   ip_ttl <= 60: trojans (2092.0)
|   ip_ttl > 60
|   |   tcp_winsize <= 64240
```

```
|   |   |   tcp_data_len <= 482
|   |   |   |   tcp_winsize <= 5840
|   |   |   |   |   tcp_data_len <= 423
|   |   |   |   |   |   tcp_data_len <= 391
|   |   |   |   |   |   |   ip_flags <= 0: trojans (3.0)
|   |   |   |   |   |   |   ip_flags > 0
|   |   |   |   |   |   |   |   tcp_data_len <= 379
|   |   |   |   |   |   |   |   |   tcp_data_len <= 370: http (3.0)
|   |   |   |   |   |   |   |   |   tcp_data_len > 370: smtp (9.0)
|   |   |   |   |   |   |   |   tcp_data_len > 379: http (4.0)
|   |   |   |   |   |   tcp_data_len > 391
|   |   |   |   |   |   |   tcp_data_len <= 412: smtp (217.0/7.0)
|   |   |   |   |   |   |   tcp_data_len > 412
|   |   |   |   |   |   |   |   tcp_data_len <= 415: http (8.0/1.0)
|   |   |   |   |   |   |   |   tcp_data_len > 415: smtp (56.0/3.0)
|   |   |   |   |   tcp_data_len > 423: http (173.0/66.0)
|   |   |   |   tcp_winsize > 5840
|   |   |   |   |   tcp_data_len <= 251
|   |   |   |   |   |   tcp_data_len <= 203: ftp-c (6.0/1.0)
|   |   |   |   |   |   tcp_data_len > 203: http (52.0/9.0)
|   |   |   |   |   tcp_data_len > 251
|   |   |   |   |   |   ip_flags <= 0: http (288.0)
|   |   |   |   |   |   ip_flags > 0
|   |   |   |   |   |   |   ip_ttl <= 64: http (39.0)
|   |   |   |   |   |   |   ip_ttl > 64: trojans (3.0)
|   |   |   tcp_data_len > 482
|   |   |   |   tcp_data_len <= 552
|   |   |   |   |   ip_ttl <= 64: http (250.0/10.0)
|   |   |   |   |   ip_ttl > 64
|   |   |   |   |   |   ip_flags <= 0: http (102.0)
|   |   |   |   |   |   ip_flags > 0: trojans (4.0)
|   |   |   |   tcp_data_len > 552: http (5316.0/3.0)
|   |   tcp_winsize > 64240: trojans (379.0/4.0)

Число листьев : 32
Размер дерева : 63

Время, затраченное на построение модели: 1.02 секунд

=== Многослойная перекрестная проверка ===
=== Итог ===
Корректно классифицированных экземпляров       21855               98.6815 %
Некорректно классифицированных экземпляров       292                1.3185 %
Средняя абсолютная ошибка                      0.0113
Корень из среднеквадратической ошибки          0.0773
Относительная абсолютная ошибка                3.1084 %
Общее количество экземпляров                   22147
```

```
=== Детальная точность по классам ===
TP Оценка   FP Оценка   Точность      Класс
  0.982       0.006       0.984       http
  0.988       0.005       0.988       ftp-c
  0.99        0.007       0.984       smtp
  0.988       0           0.998       trojans

=== Смешанная матрица ===
    a    b    c    d
 6144   71   36    5 |    a = http
   13 6443   65    0 |    b = ftp-c
   71    0 6767    0 |    c = smtp
   14    6   11 2501 |    d = trojans
```

После построения дерева методом J4.8 происходит усечение (prunning) его ветвей, то есть группировка нескольких узлов в один лист либо замещение узла дерева нижележащим поддеревом. Эта операция важна для поддержания обобщающей способности построенной модели и позволяет избежать проблемы переобучения (overfitting). По мере того как записи распределяются по узлам, их количество в каждом узле становятся меньше, и в них преобладают экземпляры с близкими значениями атрибутов. С уменьшением числа экземпляров в узлах падает ценность связанных с ними правил. И если на некотором уровне решений разбиение дало узлы, содержащие 1-2 экземпляра, то такие разбиения не имеют смысла, поскольку обобщающая способность такой модели оказывается очень малой. Перед операцией над деревом вычисляется ошибка правила классификации, содержащегося в рассматриваемом узле. Если после замещения (или группировки) ошибка не возрастает, значит, замену можно произвести без ущерба для построенной модели (см. рисунок 22).

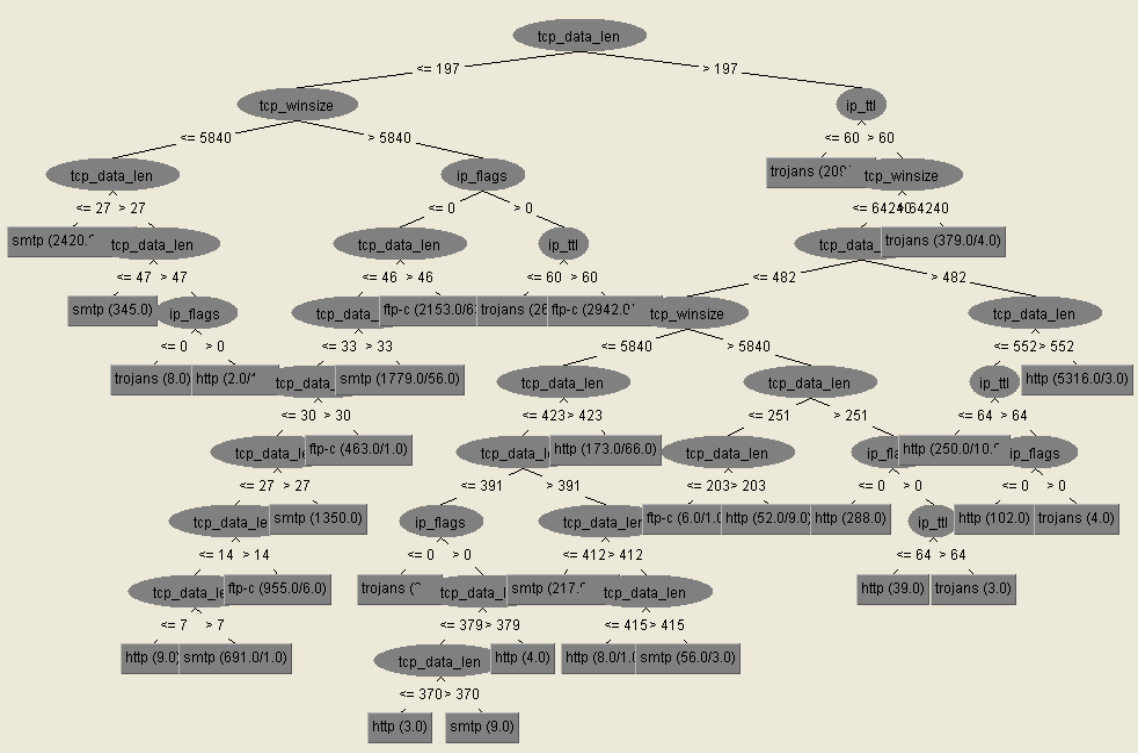

Рисунок 22. Дерево, построенное с помощью алгоритма J4.8

Значения в скобках, например (2.0/1.0), означают, что из двух случаев, достигших листа, один был классифицирован неправильно.

Последовательно протестируем метод J4.8 на подготовленных для тестирования множествах (Test-Set). Результаты проверки представлены в таблице 7.

Таблица 7. Отношение количества классифицированных с помощью алгоритма J4.8 и предварительно классифицированных пакетов для обучающего набора Training-Set-3

Название дампа	HTTP		FTP-C		SMTP		Trojans		Ошибки, %
	После	До	После	До	После	До	После	До	
Test-Set-1	3408	3517	58	-	50	-	1	-	3
Test-Set-2	40	-	3412	3494	42	-	0	-	2
Test-Set-3	63	-	1	-	6145	6209	0	-	1
Test-Set-4	252	-	113	-	70	-	5231	5666	7.7

В результате выполнения метода J4.8 достигается высокий процент правильно классифицированных сетевых пакетов (98%), несмотря на небольшое число исходных признаков.

2.4.3. Метод опорных векторов (Support Vector Machines, SVM или Sequential Minimal Optimization, SMO в Weka)

Идея метода основывается на предположении о том, что наилучшим способом разделения точек в *m*-мерном пространстве является *m-1* плоскость (заданная функцией *f*(*x*)), равноудаленная от точек, принадлежащих разным классам. Для двумерного пространства эту идею можно представить в виде, изображенном на рисунке 23.

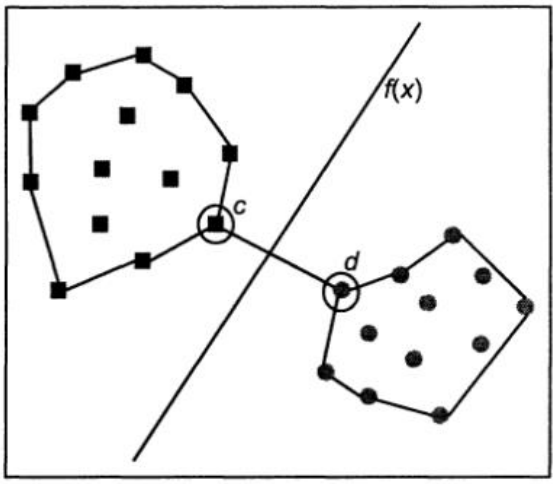

Рисунок 23. Графическая интерпретация идеи метода SVM

Как можно заметить, для решения этой задачи достаточно провести плоскость, равноудаленную от ближайших друг к другу точек, относящихся к разным классам. На рисунке такими точками являются точки *c* и *d*. Данный метод интерпретирует объекты (и соответствующие им в пространстве точки) как векторы размера *m*. Другими словами, независимые переменные, характеризующие объекты, являются координатами векторов. Ближайшие друг к другу векторы, относящиеся к разным классам, называются опорными векторами (*support vectors*) [50].

Данный метод является алгоритмом классификации с использованием математических функций. Основная идея метода опорных векторов – перевод исходных векторов в пространство более высокой размерности и поиск максимальной разделяющей гиперплоскости в этом пространстве. Однако следует помнить, что чем выше размерность пространства, тем сложнее с ним работать. Один из вариантов работы с данными высокой размерности – это предварительное применение какого-либо метода понижения размерности данных для выявления наиболее существенных компонент, а затем использование метода опорных векторов.

Метод SVM имеет свои сильные и слабые стороны, которые следует учитывать при выборе данного метода. Недостаток метода состоит в том, что для классификации используется не все множество образцов, а лишь их небольшая часть, которая находится на границах. Достоинство метода состоит в том, что для классификации методом опорных векторов, в отличие от большинства других методов, достаточно небольшого набора данных.

После проведения обучения для множества Training-Set-3 методом SVM (в Weka называется SMO) с использованием перекрестной проверки (подробнее о методе проверки см. п.2.4.5) были получены следующие результаты:

```
Используется ядро:
  Линейное ядро: K(x,y) = <x,y>

Классификатор для классов: http, ftp-c
Бинарный SMO
Линейный механизм: показаны веса атрибутов, нет опорных векторов.
         0.0693 * (нормированный) ip_flags
 +      -0.06   * (нормированный) ip_ttl
 +       0.7473 * (нормированный) tcp_winsize
 +     -14.9685 * (нормированный) tcp_data_len
 +      -0.0693 * (нормированный) tcp_flow_dir
 +       1.9152
Количество оцененных ядер: 969165 (61.713% кэшировано)

Классификатор для классов: http, smtp
Бинарный SMO
Линейный механизм: показаны веса атрибутов, нет опорных векторов.
         0.0472 * (нормированный) ip_flags
 +      -0.0408 * (нормированный) ip_ttl
 +      -4.092  * (нормированный) tcp_winsize
 +     -17.8012 * (нормированный) tcp_data_len
 +      -0.0472 * (нормированный) tcp_flow_dir
 +       5.6396
Количество оцененных ядер: 1602607 (76.798% кэшировано)

Классификатор для классов: http, trojans
Бинарный SMO
Линейный механизм: показаны веса атрибутов, нет опорных векторов.
         4.3095 * (нормированный) ip_flags
 +       2.3122 * (нормированный) ip_ttl
 +      -0.0014 * (нормированный) tcp_winsize
 +      -0.0004 * (нормированный) tcp_data_len
 +       2.3109 * (нормированный) tcp_flow_dir
 -       5.6209
```

```
Количество оцененных ядер: 10438272 (50.382% кэшировано)

Классификатор для классов: ftp-c, smtp
Бинарный SMO
Линейный механизм: показаны веса атрибутов, нет опорных векторов.
       -11.1514 * (нормированный) ip_flags
 +       9.6445 * (нормированный) ip_ttl
 +     -31.3241 * (нормированный) tcp_winsize
 +      -0.0014 * (нормированный) tcp_data_len
 +      11.1514 * (нормированный) tcp_flow_dir
 +      10.8484
Количество оцененных ядер: 17384332 (45.383% кэшировано)

Классификатор для классов: ftp-c, trojans
Бинарный SMO
Линейный механизм: показаны веса атрибутов, нет опорных векторов.
         3.7651 * (нормированный) ip_flags
 +       1.1986 * (нормированный) ip_ttl
 +      -0.4006 * (нормированный) tcp_winsize
 +       6.7578 * (нормированный) tcp_data_len
 +       2.4787 * (нормированный) tcp_flow_dir
 -       5.1223
Количество оцененных ядер: 1749354 (67.138% кэшировано)

Классификатор для классов: smtp, trojans
Бинарный SMO
Линейный механизм: показаны веса атрибутов, нет опорных векторов.
         3.7858 * (нормированный) ip_flags
 +      -0.857  * (нормированный) ip_ttl
 +       3.0227 * (нормированный) tcp_winsize
 +       4.305  * (нормированный) tcp_data_len
 +       2.7165 * (нормированный) tcp_flow_dir
 -       5.9503

Количество оцененных ядер: 2037242 (76.801% кэшировано)

Время, затраченное на построение модели: 36.89 секунд

=== Многослойная перекрестная проверка ===
=== Итог ===
Корректно классифицированных экземпляров       17529               79.1484 %
Некорректно классифицированных экземпляров      4618               20.8516 %
Средняя абсолютная ошибка                      0.2682
Корень из среднеквадратической ошибки          0.3381
Относительная абсолютная ошибка               73.9703 %
Общее количество экземпляров                   22147
```

```
=== Детальная точность по классам ===
TP Оценка    FP Оценка    Точность      Класс
  0.974       0.039        0.907         http
  0.336       0            1             ftp-c
  0.996       0.261        0.63          smtp
  0.962       0            1             trojans

=== Смешанная матрица ===
    a    b    c    d
 6091    0  165    0 |    a = http
  535 2192 3794    0 |    b = ftp-c
   28    0 6810    0 |    c = smtp
   63    1   32 2436 |    d = trojans
```

На выводе алгоритма показываются веса (коэффициенты задающие плоскость, разделяющую исходные данные на классы) для всех возможных атрибутов, при этом заметна задержка его вывода из-за проведения расчетов (самый медленный алгоритм из всех рассмотренных). Процент верной классификации оказывается достаточно высоким – 79,1%, а средняя ошибка классификатора наоборот, оказывается минимальной среди всех рассмотренных методов.

Последовательно протестируем метод SVM на подготовленных для тестирования множествах (Test-Set). Результаты проверки представлены в таблице 8.

Таблица 8. Отношение количества классифицированных с помощью алгоритма SVM и предварительно классифицированных пакетов для обучающего набора Training-Set-3

Название дампа	HTTP		FTP-C		SMTP		Trojans		Ошибки, %
	После	До	После	До	После	До	После	До	
Test-Set-1	3303	3517	5	-	209	-	0	-	6
Test-Set-2	466	-	1350	3494	1678	-	0	-	61.4
Test-Set-3	0	-	0	-	6209	6209	0	-	0
Test-Set-4	27	-	100	-	24	-	5515	5666	2.7

Из таблицы 8 видно, что наибольшее количество ошибочно классифицированных пакетов наблюдается для FTP-C протокола (61.4%), хотя для SMTP-протокола абсолютно все пакеты были классифицированы правильно.

2.4.4. Метод OneR

Метод строит правила по значению одной независимой переменной, поэтому в литературе его часто называют “1-правило” (1-rule) или кратко 1R-алгоритм (OneR).

Идея алгоритма очень проста. Для любого возможного значения каждой независимой переменной формируется правило, которое классифицирует объекты из обучающей выборки. При этом в заключительной части правила указывается значение зависимой переменной, которое наиболее часто встречается у объектов с выбранным значением независимой переменной. В этом случае ошибкой правила является количество объектов, имеющих то же значение рассматриваемой переменной, но не относящихся к выбранному классу.

Таким образом, для каждой переменной будет получен набор правил (для каждого значения). Оценив степень ошибки каждого набора, выбирается переменная, для которой построены правила с наименьшей ошибкой и на основании выбранной переменной производится классификация.

Метод классификации OneR – один из самых простых методов классификации. Применяется как к числовым данным, которые разбиваются на промежутки, так и к данным типа nominal.

После проведения обучения для множества Training-Set-3 методом OneR с использованием перекрестной проверки (подробнее о методе проверки см. п.2.4.5) были получены следующие результаты:

```
tcp_data_len:
       < 0.5  -> trojans
       < 6.5  -> ftp-c
       < 8.5  -> smtp
       < 13.5 -> ftp-c
       < 14.5 -> smtp
       < 17.5 -> ftp-c
       < 18.5 -> smtp
       < 22.5 -> ftp-c
       < 23.5 -> smtp
       < 26.5 -> ftp-c
       < 29.5 -> smtp
       < 34.5 -> ftp-c
       < 35.5 -> smtp
       < 40.5 -> ftp-c
       < 42.5 -> smtp
       < 44.5 -> http
       < 45.5 -> smtp
```

```
< 130.5      -> ftp-c
< 131.5      -> http
< 194.0      -> ftp-c
< 202.0      -> trojans
< 262.5      -> http
< 276.0      -> trojans
< 283.0      -> http
< 286.5      -> trojans
< 329.5      -> http
< 347.5      -> trojans
< 350.5      -> http
< 374.0      -> trojans
< 375.5      -> smtp
< 384.5      -> http
< 389.0      -> trojans
< 398.5      -> smtp
< 409.5      -> trojans
< 410.5      -> smtp
< 414.5      -> trojans
< 416.5      -> smtp
< 421.5      -> trojans
< 422.5      -> smtp
< 426.5      -> http
< 428.5      -> smtp
< 430.5      -> trojans
< 431.5      -> http
< 434.5      -> smtp
< 443.5      -> http
< 446.5      -> smtp
< 455.5      -> http
< 459.5      -> trojans
< 463.0      -> http
< 464.5      -> smtp
< 469.5      -> http
< 470.5      -> smtp
< 475.5      -> http
< 476.5      -> smtp
< 512.5      -> http
< 515.5      -> trojans
< 549.5      -> http
< 552.5      -> trojans
< 702.0      -> http
< 709.5      -> trojans
< 756.5      -> http
< 771.5      -> trojans
< 876.0      -> http
< 906.5      -> trojans
```

```
       < 930.5      -> http
       < 951.0      -> trojans
       < 993.0      -> http
       < 1003.5     -> trojans
       < 1039.5     -> http
       < 1061.5     -> trojans
       < 1261.5     -> http
       < 1288.5     -> trojans
       >= 1288.5    -> http
(18192/22147 правильных экземпляров)

Время, затраченное на построение модели: 0.17 секунд

=== Многослойная перекрестная проверка ===
=== Итог ===
Корректно классифицированных экземпляров       18132               81.8711 %
Некорректно классифицированных экземпляров      4015               18.1289 %
Средняя абсолютная ошибка                     0.0906
Корень из среднеквадратической ошибки         0.3011
Относительная абсолютная ошибка              25.0013 %
Общее количество экземпляров                  22147

=== Детальная точность по классам ===
TP Оценка   FP Оценка   Точность   Класс
  0.954       0.137       0.732      http
  0.948       0.08        0.832      ftp-c
  0.827       0.028       0.93       smtp
  0.131       0.008       0.671      trojans

=== Смешанная матрица ===
    a    b    c    d
 5967   80   67  142 |    a = http
   22 6180  317    2 |    b = ftp-c
   37 1129 5654   18 |    c = smtp
 2122   40   39  331 |    d = trojans
```

Метод выбирает переменные, принимающие наибольшее возможное количество значений, для таких переменных ошибка будет наименьшей. К примеру, для переменной, по которой у каждого ключа свое уникальное значение ошибка будет равна нулю. В нашем случае была выбрана переменная *tcp_data_len* (см. рисунок 24).

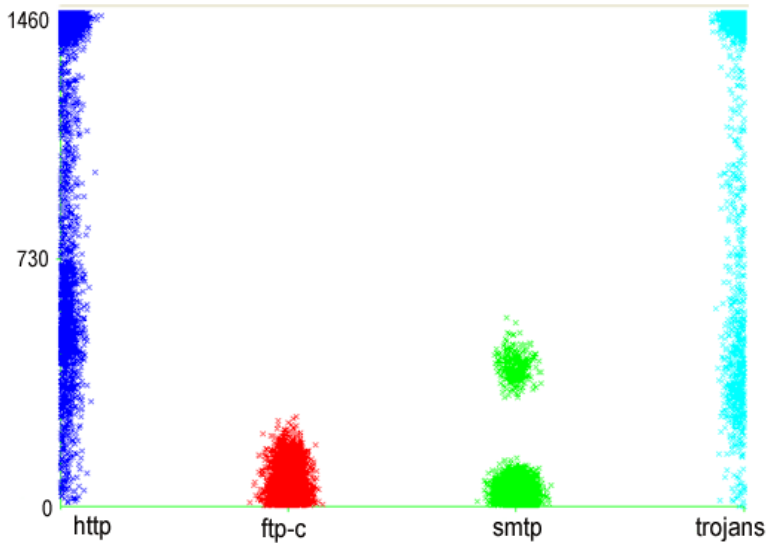

Рисунок 24. График зависимости типа трафика от размера пакета (цветами обозначены типы трафика: HTTP – синий, FTP-C – красный, SMTP – зеленый, Trojans - голубой)

Последовательно протестируем метод OneR на подготовленных для тестирования множествах (Test-Set). Результаты проверки представлены в таблице 9.

Таблица 9. Отношение количества классифицированных с помощью алгоритма SVM и предварительно классифицированных пакетов для обучающего набора Training-Set-3

Название дампа	HTTP		FTP-C		SMTP		Trojans		Ошибки, %
	После	До	После	До	После	До	После	До	
Test-Set-1	3082	3517	56	-	128	-	251	-	12.4
Test-Set-2	43	-	3309	3494	123	-	19	-	5.3
Test-Set-3	58	-	1309	-	4797	6209	45	-	22.7
Test-Set-4	4729	-	293	-	109	-	535	5666	90.6

Из таблицы 9 видно, что наихудшие результаты метод классификации OneR показал для трафика, сгенерированного ВПО (4729 сетевых пакетов были ошибочно отнесены к классу HTTP). Такие результаты объясняются схожими значениями выбранного в качестве параметра классификации признака *tcp_data_len* для HTTP и Trojans классов. Несмотря на наличие ошибки при классификации сетевых пакетов, относящихся к классу Trojans, метод OneR показал высокую точность классификации (82%).

2.4.5. Оценка обучающих алгоритмов с учителем

Задача оценки обучающих алгоритмов с учителем усложняется в случае, когда фазы обучения и тестирования должны выполняться с использованием наборов предварительно классифицированных (отмеченных)[4] данных.

В идеале можно иметь большое обучающее множество (для оптимального обучения и создания моделей) и большой, уже независимый, набор тестовых данных, правильно оценивающий характеристики алгоритма. Проверка на обучающем наборе данных часто вводит в заблуждение. Она, обычно, показывает, что построенная модель хороша для распознанных экземпляров, из которых она была построена.

В реальном мире исследователи часто сталкиваются с количественными ограничениями для предварительно классифицированных наборов данных. Простая процедура (еще известная как контроль на отложенных данных [43]) предполагает выделение определенной части (например, двух третьих) от предварительно классифицированного множества для обучения и остаток (например, одну треть) – для тестирования.

На практике, когда доступен лишь ограниченный набор данных, вариант контроля на отложенных данных, называемый *N-кратной перекрестной проверкой* (N-fold cross-validation), является наиболее используемым. Набор данных, во-первых, разбивается на N аппроксимированных равных сегментов (или слоев). Каждый сегмент ($1/N$) по очереди используется для тестирования, в то время как оставшаяся часть ((N - 1) / N) используется для обучения. Процедура повторяется N раз для того, чтобы в конце проверки все экземпляры были использованы в точности один раз.

Простое разбиение полного набора данных N способами не гарантирует, что будет использовано одинаковое соотношение для любого данного класса в рамках набора данных. Обычно применяется следующий шаг, известный как *стратификация* – случайная выборка данных таким способом, при котором каждый класс должным образом представлен и в обучающем, и в тестирующем наборах данных.

---

[4] В данном контексте процесс ‘маркировки’ является частью классификации, используя ручное (человеческое) вмешательство или неоспоримый автоматический процесс.

2.4.6. Сравнение методов классификации для тестовых множеств

Обобщив результаты, представленные в таблицах 6-9, составим сводную таблицу, в которой будет указано среднее значение процента ошибочно классифицированных сетевых пакетов (таблица 10).

Таблица 10. Сводная таблица соотношения метода классификации, тестируемого набора и процента ошибочно классифицированных пакетов

Название дампа	Naïve Bayes	J4.8	SVM	OneR	Итог
Test-Set-1	21.3	3	6	12.4	J4.8 (HTTP)
Test-Set-2	28.2	2	61.4	5.3	J4.8 (FTP-C)
Test-Set-3	64	1	0	22.7	J4.8, SVM (SMTP)
Test-Set-4	12	7.7	2.7	90.6	SVM (Trojans)

В столбце “Итог” перечислены методы классификации, соотнесенные с каждым из тестирующих множеств на основании минимального процента ошибок, допущенных в процессе классификации. В скобках указан тип трафика (протокол), преобладающий в тестовом дампе: для классификации HTTP и FTP-C трафика предпочтительно использовать метод J4.8, наименьшее количество ошибок для Trojans продемонстрировал SVM метод, для SMTP хорошие результаты показали методы J4.8 и SVM.

2.4.7. Сравнение методов классификации для обучающих множеств

Интерфейс Исследователя Weka позволил получить сведения о количестве ошибочно классифицированных пакетов для отдельных типов трафика, при этом использовался метод перекрестной проверки и подстановки отдельных тестирующих множеств. В продолжение исследований воспользуемся интерфейсом Экспериментатор (см. рисунок 25), который предназначен для сравнения на основе статистических механизмов применимости построения классификаторов к конкретным данным. Фактически, он многократно выполняет перекрестную проверку для каждого из методов и на основе полученных результатов формирует выходной набор данных.

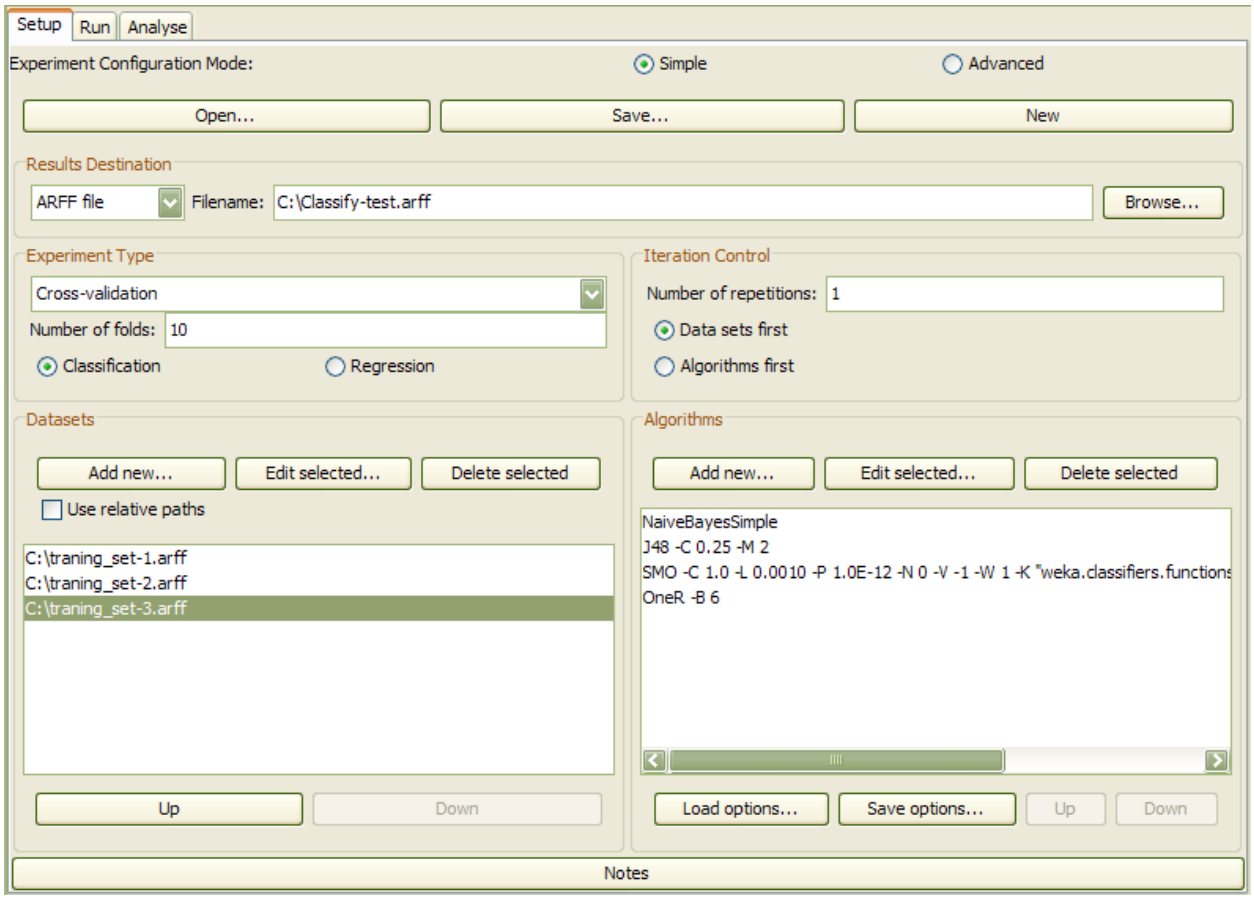

Рисунок 25. Окно загрузки данных и выбора методов, которые будут сравниваться в дальнейшем ходе исследования

Количество проходов перекрестной проверки устанавливаем равным 10 (наиболее часто применяется на практике), а количество проходов каждого алгоритма устанавливается равным 1. Сравнение методов производится по полю “Percent Correct”, то есть по количеству правильно классифицированных данных в процентах.

Воспользуемся методом сравнения с одним из выбранных алгоритмов (“наивным” Байесом). В результате проведения вычислений формируется таблица (подробнее об описании результатов сравнения см. п. 2.2.5), в строках которой указываются методы классификации, а в рядах – входные наборы данных (см. рисунок 26):

```
Набор данных  (1)        NB        |  (2) trees.J48     (3) SVM      (4) rules.OneR
   ------------------------------------------------------------------------------
traning_set_1 (10)    68.10(1.21) |    98.71(0.26) v   76.90(0.50) v   91.73(0.86) v
traning_set_2 (10)    56.45(3.16) |    98.08(0.44) v   72.17(0.25) v   92.32(0.77) v
traning_set_3 (10)    66.13(1.00) |    98.68(0.36) v   79.15(0.44) v   81.87(0.63) v
   ------------------------------------------------------------------------------
                          (v/ /*) |      (3/0/0)         (3/0/0)         (3/0/0)
```

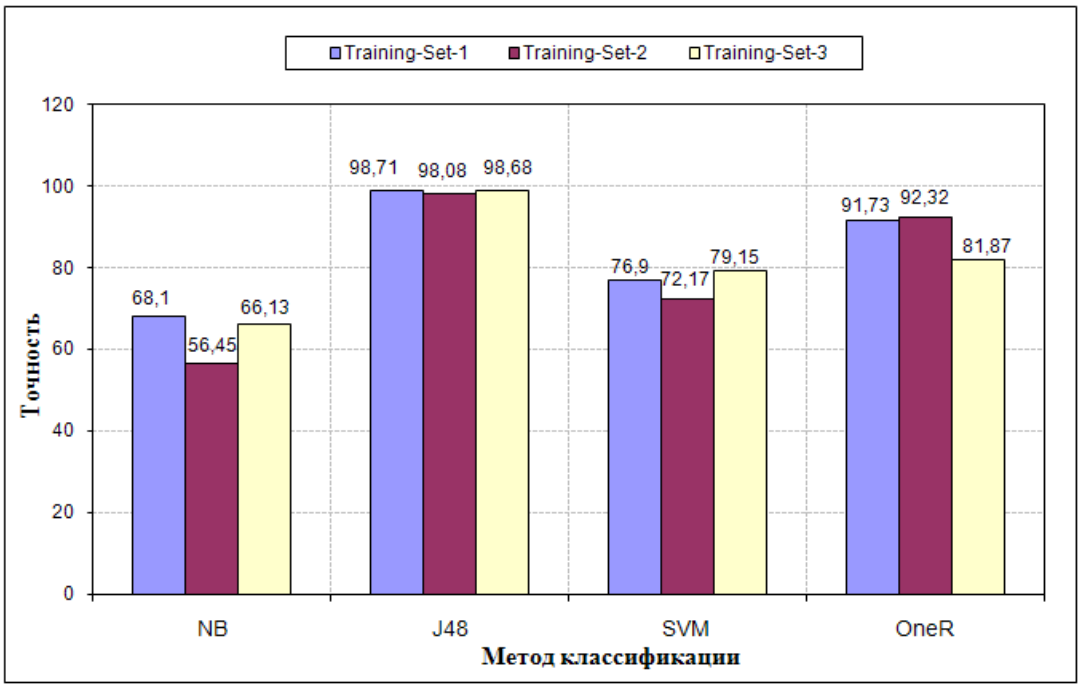

Рисунок 26. Диаграмма зависимости метода классификации от точности классификации для трех обучающих множеств Training-Set-1, 2, 3

В нашем случае, к примеру, для первого обучающего набора процент верно классифицированных экземпляров данных выглядит следующим образом (в скобках указано стандартное отклонение):

- J48 – 98.71 (0.26)
- OneR – 91.73 (0.86)
- SVM – 76.90 (0.50)
- NaiveBayes – 68.10 (1.21)

Наилучшие результаты для всех трех обучающих множеств показал метод J4.8 (98.71%), несмотря на свою простоту (и в связи с отсутствием в обучающих наборах Training-Set-1 и Training-Set-2 трафика, сгенерированного ВПО, см. п. 2.4.4), алгоритм OneR также показал хорошие результаты (91.73%), далее следует метод SVM (76.90%) и Naïve Bayes (68.10%).

3. ВРЕДНЫЕ ПРОИЗВОДСТВЕННЫЕ ФАКТОРЫ ПРИ РАБОТЕ С ПК, ЭЛЕКТРОННО-ЛУЧЕВЫМИ И ЖИДКОКРИСТАЛЛИЧЕСКИМИ МОНИТОРАМИ

В настоящей работе был рассмотрен метод идентификации сетевого трафика прикладного уровня. Результатами исследования пользуется человек, разрабатывающий программные средства анализа сетевого трафика. В связи с этим необходимо рассмотреть вопросы безопасности при работе с персональным компьютером и в частности с ЭЛТ и ЖК мониторами.

Проанализируем особенности труда при работе с видеотерминалами [51], выделив специфические вредные и опасные факторы как самого компьютера, так и производственной среды. Основное внимание уделим нормативным требованиям по ограничению воздействия этих факторов, эргономическим требованиям к параметрам оборудования, и более детально становимся на новейшем мировом стандарте безопасности для мониторов TCO’03.

3.1. Основные вредные и опасные факторы при работе с персональным компьютером

3.1.1. Повышенное зрительное напряжение

Повышенная нагрузка на зрение способствует возникновению близорукости, приводит к переутомлению глаз, к мигрени и головной боли, повышает раздражительность, нервное напряжение, может вызвать стресс.

Технические характеристики мониторов ПК: разрешающая способность, яркость, контрастность, частота мелькания сильно влияют на зрительную работу и могут крайне негативно сказаться на зрении, если их не учитывать при выборе и установке устройства.
В комплексе причин, отрицательно влияющих на зрение оператора ПК, в первую очередь следует выделить недостаточную контрастность изображения на экране, вызванную пространственной и временной нестабильностью, излишнюю яркость, блики и отраженный свет на поверхности монитора. Кроме того, зрение страдает от частого переноса взгляда с ярко освещенного экрана на клавиатуру и документацию, что в совокупности с другими факторами может привести к астенопатии.

В термин “астенопатия” специалисты вкладывают проявление зрительных симптомов (пелена перед глазами, неясные очертания предметов, изменение их цвета и др.) и глазных симптомов (ощущение усталости глаз, повышение их температуры, дискомфорт, боли в глазах). За этим понятием стоят признаки нарушения функций всех

звеньев зрительного анализатора, включая как перенапряжение мышечного аппарата глаза, так и изменение биохимических реакций в элементах сетчатки.

Объективные исследования подтверждают ухудшение основных функций зрения, а также существенное снижение работоспособности и ухудшение внимания.

3.1.2. Костно-мышечные напряжения

Причиной болезней пальцев и кистей рук является специфика работы на клавиатуре: пользователи с высокой скоростью повторяют одни и те же движения. Поскольку каждое нажатие на клавишу сопровождается сокращением мышц, сухожилия непрерывно скользят вдоль костей и соприкасаются с тканями, в результате развиваются воспалительные процессы. Подобные болезни развивают также в плечевом суставе и в руке, когда приходится долго манипулировать “мышью”.

Причиной заболеваний, возникающих при длительном сидячем положении работающего с ПК, многие исследователи считают несоответствие параметров мебели антропометрическим характеристикам человека. Кроме того, многие виды заболеваний вызваны неудачной конструкцией клавиатуры и манипулятора “мышь”.

3.1.3. Нервное напряжение

У людей, занятых работой на ПК, по сравнению с другими профессиональными группами выявлено значительно более выраженное нервно-сенсорное напряжение по показателям критической частоты слияния мельканий и электрической чувствительности глаза, которые свидетельствуют о повышенной возбудимости и нестабильности нервной системы.

Необходимость активного внимания в процессе работы, высокая ответственность за ее результаты, особенно при управлении сложными техническими системами, при решении серьезных научных задач или выполнении финансовых операций вызывают у операторов ПК реакцию в виде психического напряжения, чаще называемую стрессом.

3.1.4. Электромагнитные поля

Особое внимание при анализе безопасности в процессе работы на ПК следует уделять потенциальному воздействию электромагнитных полей, возникающих в видеодисплейных терминалах во время эксплуатации. Видеотерминалы являются источником широкого спектра электромагнитных излучений: рентгеновского, ультрафиолетового, видимого спектра, инфракрасного, радиочастот, очень низких частот, включая промышленную.

Наиболее сильно действие электромагнитных полей проявляется на расстоянии до 30см от экрана. При этом современные требования к дисплейным устройствам таковы, что действие рентгеновского излучения незначительно уже на расстоянии 5 см. Но вредное излучение не меньшей интенсивности имеют боковые и задняя поверхность монитора. Это обстоятельство также нужно учитывать при организации рабочего места. Как установлено, воздействие электромагнитного поля способствует развитию катаракты и глаукомы, нежелательных явлений в период беременности, разрушению зубных пломб на основе амальгамы и др. Кроме того, низкочастотные поля могут явиться причиной раковых заболеваний, некоторых заболеваний кожи, нарушения протекания беременности, вероятностью нарушения репродуктивной функции. Электростатическое поле также оказывает негативное воздействие на пользователей ПК.

3.1.5. Другие факторы

Помимо перечисленных выше негативных факторов, связанных прежде всего с визуальными и эмиссионными параметрами работы компьютеров и особенностями работы с ПК, на пользователя также могут оказывать неблагоприятное влияние также шум от работы самого ПК и оборудования в помещении (иногда до 80 дБА, что существенно выше нормы), тепловыделения и выделение вредных веществ (фенола, формальдегида и стирола) в воздух рабочей зоны при эксплуатации ПК. Кроме того, всегда имеется потенциальная опасность поражения электрическим током при пользовании устройством, питаемым электроэнергией, при несоблюдении правил техники безопасности. При неправильной эксплуатации и подключении нескольких электроприборов к источнику питания существует опасность возгорания вследствие перегрузки.

3.2. TCO'03 - Стандарт на эргономику, экологию и безопасность дисплеев

TCO (профсоюз Швеции) занимается регламентацией ИТ-оборудования, в частности видеодисплеев, с конца 80-х годов 20-го века. В 1998 г. этой работой занялась группа TCO Development, дочерняя фирма TCO. Используя коллективные знания и опыт более миллиона офисных служащих, сотрудничающих с профсоюзом TCO, эта группа разработала требования и методики тестирования офисного компьютерного оборудования. Данные требования, определяющие качество и экологическую безопасность, проложили путь быстрому развитию всемирно признанного стандарта.

Первый вариант системы TCO был запущен в 1992 г., сменяясь затем новыми версиями — TCO'95, TCO'99, каждая из которых вносила всё больший охват требований и всё большую их строгость в соответствии с научно-техническим прогрессом. Данный

стандарт, принятый в ноябре 2002 г., знаменует четвёртое поколение TCO. Его основные разделы преимущественно совпадают с ранними версиями, но по количеству и характеру требований имеются существенные отличия, особенно в части визуальной эргономики, как наиболее проблемной и быстро развивающейся в последние годы.

3.2.1. Визуальная эргономика

Многие проблемы визуальной эргономики дисплея могут быть хорошо видны даже невооружённым глазом. Однако отдельные характеристики могут быть неоднозначными при восприятии и измерении. В большинстве случаев реальный мир гораздо сложнее, чем любое его научное описание. Тем не менее, это не повод не пытаться разрабатывать методики тестирования и требования, предъявляемые к оборудованию.

3.2.1.1. Требования к размеру пикселя

Применимость: только ЖК-дисплеи.

Качество изображения может быть заметно ухудшено в следствие низкого коэффициента заполнения, заметной ступенчатости, плохой передачей деталей. Все эти параметры связаны с задействованным массивом пикселей — их количеством и, главное, угловым размером. Разумеется, расстояние от пользователя до экрана вносит поправки на требуемый линейный размер пикселя. Пиксель (pixel) — наименьший адресуемый элемент экрана, способный воспроизводить полный диапазон яркости и цвета.

Требование состоит в соблюдении плотности пикселей ≥30 на градус на расстоянии 50 см или по таблице:

Таблица 11. Минимальное разрешение в зависимости от диагонали монитора

Диагональ, дюймы	Минимальное разрешение
15, 16	1024 x 768
17, 18, 19	1280 x 1024
21	1600 x 1200

Для широких экранов и других специализированных дисплеев требования могут быть пересчитаны. Горизонтальное и вертикальное разрешения должны соотноситься с размерами экрана — шириной и высотой — настолько близко, насколько возможно.

3.2.1.2. Геометрические характеристики изображения

Линейность

Применимость: только ЭЛТ-дисплеи.

Линейностью (linearity) называют адекватность отображения горизонтальных и вертикальных линий: они должны быть прямыми и непрерывными. Искажение вдоль любой из четырёх сторон экрана не должно превышать 1 %. В простейшем случае, без поправок на наклон растра, поворот и другие его параметры, искажение есть частное максимального отклонения линии от прямой и длины прилегающей стороны. Например, искажение вдоль верхней стороны будет равно максимальному отклонению линии, проходящей вдоль этой самой стороны, делённому на высоту экрана.

Точность представляемого результата: до десятой доли процента.

Допуск: ±0,2 % от измеренных размеров.

Ортогональность

Применимость: только ЭЛТ-дисплеи.

Ортогональность (orthogonality) означает соблюдение перпендикулярности горизонтальных и вертикальных линий: прямоугольник должен быть похожим на прямоугольник, а не на трапецию или параллелограмм.

Трапецевидность должна быть в пределах 2 %, ортогональность — в пределах 3 %.

Коэффициент трапецевидности есть удвоенное отношение разностей размеров к их сумме. Например, чтобы посчитать горизонтальную трапецевидность, надо из ширины изображения, замеренной по верхней стороне экрана, вычесть ширину, замеренную у нижней стороны; а затем абсолютное значение полученного результата поделить на сумму этих же размеров и умножить на два. Коэффициент ортогональности вычисляется точно так же, только вместо ширины или высоты используются диагонали.

Точность представляемого результата: до десятой доли процента.

Допуск: ±0,2 % от измеренных размеров.

3.2.1.3. Яркость изображения

Уровень яркости

Яркость (luminance) в самом общем случае — световая величина, равная отношению светового потока к геометрическому фактору. Для дисплеев якрость удобнее определять как отношение силы света элемента поверхности к площади его проекции, перпендикулярной рассматриваемому направлению. Яркость измеряется в канделах на метр квадратный, кд⁄м².

Для ЭЛТ яркость не должна быть менее 120 кд⁄м².

Для ЖКД яркость не должна быть менее 150 кд⁄м².

Яркость и контрастность на дисплее устанавливаются на максимум. Если изображение выглядит неудовлетворительно, сначала пытаются снизить яркость, затем контрастность.

Для измерений используют картинку “80 %” (все компоненты RGB равны 204). В центр экрана помещается белый квадрат (RGB=255) со стороной 3 см. Замер производится по центру этого квадрата.

Допуск: ±10 % от измеренного значения яркости.

Равномерность яркости

Равномерность яркости (luminance uniformity) — это способность дисплея обеспечивать одинаковый уровень яркости по всей активной площади экрана. Определяется как отношение максимальной и минимальной яркостей.

Разброс яркости в пределах активной области экрана, L_{max} : L_{min} не должен превышать 1,5 в условиях тестирования.

Измерение ведётся по всей активной поверхности экрана. Используется картинка “100 %” (RGB=255). Выбираются участки с максимальной и минимальной яркостью. Если таковые явно не выделяются, замер производится в центре и по четырём углам. Яркость измеряется для участка размером 1°, то есть круга диаметром 8,7 мм при удалении 50 см.

Точность измерений: до сотых долей.

Допуск: ±10 % от измеренного значения яркости.

Независимость яркости от нагрузки

Применимость: только ЭЛТ-дисплеи.

Зачастую при сильной нагрузке, когда на дисплее отображаются большие области белого цвета или просто светлые участки, уровень яркости экрана может снижаться. Запас нагрузки (image loading capacity) — способность дисплея в определённой степени сохранять уровень яркости изображения вне зависимости от яркости элементов этого изображения.

Яркость в центре экрана должна быть не ниже указанной в таблице:

Таблица 12. Зависимость яркости от нагрузки

Нагрузка, %	80	100
Яркость, кд/м^2	100	80

Загрузка 80 % примерно соответствует равенству всех каналов RGB=204. Загрузка 100 % является максимальной: RGB=255. Белый квадрат со стороной 4 см окружается

картинкой “80 %”. Яркость в этом квадрате настраивается равной 100 кд⁄м². Затем весь экран заливается белым цветом и берётся ещё один замер яркости в том же квадрате. Отношение двух полученных величин и является мерой запаса нагрузки.

Точность представляемого отношения яркости: до сотых долей процента.

Допуск: ±10 % от измеренного значения яркости.

Независимость яркости от угла обзора

Применимость: только ЖК-дисплеи.

В отличие от ЭЛТ-дисплеев, яркость ЖКД часто является зависимой от угла обзора. Небольшое движение головы при рассматривании различных частей экрана может вызвать заметное изменение воспринимаемого свечения, подобного неравномерному распределению яркости. Независимостью яркости от угла обзора (luminance uniformity – angular dependence) называют способность дисплея обеспечивать яркость в достаточных пределах при заданном диапазоне углов обзора. В идеале дисплей должен обеспечивать одинаковую яркость под любым углом.

Для дисплеев в ландшафтной ориентации (большая сторона параллельна горизонту) в горизонтальном направлении в диапазоне углов ±30° среднее значение отклонения L_{max} :L_{min} не должно превышать 1,7.

В вертикальном направлении допускаемое отклонение то же, но иной диапазон углов поворота дисплея — от строго вертикального до 15° вверх. Для дисплеев в портретной ориентации (меньшая сторона параллельна горизонту) диапазон углов ±15° применяется и для горизонтального направления.

Измерительное оборудование располагают на расстоянии, в полтора раза превышающем размер экрана по диагонали, то есть не ближе 50 см. Контрольными областями служат квадраты со стороной 4 см, расположенные в центре и у краёв экрана (на расстоянии от края, равном десятой части ширины активной области) по середине каждой из сторон.

Сначала яркомер устанавливается напротив центра экрана. Затем дисплей вращают в горизонтальной плоскости на 30° влево и вправо. Потом — в вертикальной плоскости на 15° вверх. Расстояние между дисплеем и датчиком остаётся неизменным.

Отклонение для каждого угла поворота дисплея равно L_{max} :L_{min}. Для горизонтальной плоскости оно усредняется: значение, полученное при повороте влево и значение при повороте вправо, суммируются и делятся на два. Для вертикальной плоскости значение только одно — при измерении на +15°.

Точность представляемого результата: до сотых долей.

Допуск: ±10 % от измеренного значения яркости, ±1° угла поворота.

3.2.1.4. Контрастность изображения

Контрастность

Применимость: только ЭЛТ-дисплеи.

Контрастность (luminance contrast) — это отношение между уровнями яркости некоторого элемента и окружающей его области изображения. Контраст является залогом чёткости изображения и узнаваемости символов. Мерой контрастности является коэффициент модуляции — отношение разности максимальной и минимальной яркостей элемента (в месте с фоном) и их суммы. Коэффициент модуляции должен быть не ниже 0,52 для тестового уровня яркости: 100 кд⁄м² для ЭЛТ и 125 кд⁄м² для ЖК (см. требование к уровню яркости).

Испытания проводятся только для области экрана, отстоящей на 0,05 D (D равно диагональному размеру экрана) по диагонали от края активной области, то есть края не учитываются. В качестве тестовых символов используются латинские “m” и “e" как наиболее представительные: “m” содержит достаточное количество вертикальных линий, а “e” — достаточное количество горизонтальных. Эти символы отображаются на экране 12-м кеглем и гарнитурой Arial.

Область интегрирования яркости по направлению сканирования должна соответствовать анатомическим особенностям человека: 1 угловая минута или примерно 0,145 мм на удалении 50 см. При этом точность сканера должна быть достаточной, чтобы обеспечить шаг сканирования до 1⁄8 пикселя, но не более 0,05 мм. Сканер перемещается по символу в заданном направлении (например, для вертикальных линий — в горизонтальном). “Лишние” детали изображения, не являющиеся тестовыми вертикальными (горизонтальными) линиями, не должны присутствовать на пути сканирования.

Значения, полученные со сканера, усредняются до 1 угловой минуты и 4 минут. Оба ряда усреднённых значений будут находиться довольно близко друг к другу, но значительно отличаться от измеренных значений, которые включают пики в местах прохождения сканера через активные части пикселей и спады в местах между пикселями.

L_{min} принимается равным наименьшему значению при разрешении 1′ (соответствует разрешающей способности человеческого глаза). L_{max} принимается равным наибольшему значению при разрешении 4′ (соответствует наиболее удобному восприятию, когда контрастная чувствительность глаза близка к максимуму).

Точность представляемого коэффициента модуляции: до сотых долей.

Допуск: ±10 % от измеренного значения яркости.

Независимость контрастности от угла обзора

Применимость: только ЖК-дисплеи.

Независимостью контрастности от угла обзора (luminance contrast - angular dependence) - называют способность дисплея обеспечивать контрастность в достаточных пределах при заданном диапазоне углов обзора. В идеале дисплей должен обеспечивать одинаковую контрастность под любым углом. Контрастность нормируется только для горизонтального направления: 30° влево и вправо. При этом коэффициент модуляции яркости должен быть не ниже 0,8.

Измерительное оборудование располагают на расстоянии, в полтора раза превышающем размер экрана по диагонали, то есть не ближе 50 см. Контрольными областями служат пары квадратов со стороной 4 см белого и чёрного цвета, расположенные в центре, у левого и правого краёв (по середине стороны, на расстоянии, равном одной десятой ширины активной области, от края).

Сначала яркость всех шести квадратов замеряется в положении, когда яркомер находится на нормали к центральной точке экрана. Затем дисплей поворачивают на 30° влево и вправо и повторяют измерения, не передвигая датчик.

Коэффициент модуляции для каждой пары квадратов вычисляется как разность уровней яркости белого и чёрного, отнесённая к их же сумме. В качестве результата выбирают наименьшее значение. Для сравнения обычно приводят номинальную контрастность - для центральной пары квадратов при нормальном расположении яркомера.

Точность представляемого коэффициента модуляции: до сотых долей.

Допуск: ±10 % от измеренного значения яркости.

Равномерность контрастности в деталях

Применимость: только ЖК-дисплеи.

Контрастность отдельных линий на ЖК-дисплее иногда бывает недостаточно высокой, и такие линии могут возникать в любом месте экрана, тем самым снижая чёткость. В некоторых случаях регулировкой параметров дисплея можно добиться улучшения, но далеко не всегда. Равномерностью контрастности в деталях (luminance contrast - characters) называют способность дисплея обеспечивать в заданных пределах временные характеристики формирования изображения в любом месте активной области без образования участков пониженной контрастности.

Контрастность в деталях не должна быть ниже 0,7.

Метод измерения во многом напоминает измерение самой контрастности, с той разницей, что в качестве тестового символа выступает латинская “H”, заполняющая всю активную область экрана (это аналогично картинке “80 %”). Сканирование ведётся только для вертикальных линий.

Точность представляемого коэффициента модуляции: до сотых долей.

Допуск: ±10 % от измеренного значения яркости.

3.2.2. Излучения

Несмотря на множественные исследования, эксперты не могут прийти к единому мнению о наличии и масштабах вреда, причиняемого пользователям электромагнитными полями. Однако многие операторы компьютерных дисплеев жалуются на симптомы, которые трудно отнести на счёт прочих рабочих факторов. Поэтому, до появления новых знаний в данной области, решено ограничивать уровни излучений настолько, насколько это осуществимо. При этом также нельзя обделять вниманием вопросы защиты оборудования от взаимного облучения.

Излучение традиционно делится на ионизирующее и неионизирующее. Первое существует в форме рентгеновского излучения внутри электронно-лучевой трубки, создаваясь при столкновении электрона со стеклом экранной поверхности. Для эффективного поглощения рентгеновского излучения стеклянное покрытие содержит большое количество свинца. Рекомендация TCO непосредственно не определяет содержание свинца в экране, поскольку данное требование раскрыто в части, посвящённой электрической безопасности.

ЖК-дисплеи формируют изображение методом, принципиально отличающимся от ЭЛТ. Поэтому проблем рентгеновского излучения и статического заряда на поверхности экрана у них просто не существует.

Неионизирующие излучения, применительно к дисплеям, можно разделить на следующие классы:

- электростатические поля (только ЭЛТ);
- переменные электрические поля (5 Гц – 400 кГц);
- переменные магнитные поля (5 Гц – 400 кГц).

Электростатические поля, возникающие на стеклянных поверхностях ЭЛТ-дисплеев, были большой проблемой для пользователей, начиная с 1990-х. Современные экраны изготавливаются из проводящих материалов, позволяя держать поверхность на нулевом потенциале. Несмотря на это, поле может возникнуть между пользователем и

дисплеем как следствие зарядки пользователя от синтетических тканей, ковровых покрытий и сухого воздуха. Меры борьбы заключаются в заземлении пользователя, например, через заземлённую клавиатуру.

Переменные электромагнитные поля генерируются ЭЛТ-дисплеем по разным причинам, например, из-за способа перемещения луча по экрану.

Трудно сравнить, какой же тип дисплеев — ЭЛТ или ЖК — имеет меньшие показатели облучения. Так, ЖКД традиционно располагают излучающие элементы в менее защищённых местах и гораздо ближе к оператору.

3.2.2.1. Электростатический потенциал

Применимость: только ЭЛТ-дисплеи.

Электростатический потенциал возникает на внешней поверхности ЭЛТ и зависит от разности потенциалов катода и внутренней поверхности. Эта разность потенциалов используется для ускорения электронного луча, формирующего видимое изображение на экране.

Поверхность экрана должна иметь низкий потенциал, чтобы не притягивать пыль ни к себе, ни к пользователю.

Поверхностный потенциал не должен превышать ±0,5 кВ.

Измерения проводят в лаборатории с заданной температурой воздуха (19…23 °С на расстоянии 1 м от дисплея), влажностью (20±5 %), скоростью воздушного потока (менее 0,3 м⁄с по периметру тестового стенда) и концентрацией ионов (не более 10⁹⁄м³).

Представляемый результат: качественный (“потенциал в пределах нормы”).

Допуск: ±10 %.

3.2.2.2. Переменные электрические поля

Когда разность потенциалов между двумя объектами постоянно изменяется во времени, возникает переменное электрическое поле, характеризующееся напряжённостью и частотой. Конструкция дисплея включает в себя много источников таких полей, влияние которых убывает с расстоянием.

В диапазоне I: 5 Гц – 2 кГц напряжённость на удалении 30 см вокруг дисплея и 50 см перед дисплеем не должна превышать 10 В⁄м.

В диапазоне II: 2 кГц – 400 кГц напряжённость на удалении 50 см вокруг дисплея и 30 см перед дисплеем не должна превышать 1 В⁄м.

Представляемый результат: качественный (“напряжённость в пределах нормы”).

Допуск: ±10 %.

4. ЭКОНОМИЧЕСКИЕ РЕЗУЛЬТАТЫ РАЗРАБОТКИ

Результатом работы является разработка метода идентификации сетевых протоколов прикладного уровня. В данной главе приводятся основные экономические результаты [52] разработки данного метода, расчет затрат на создание метода.

4.1. Расчет затрат

Исходные данные и расчет зарплаты исполнителей приведен в таблице. 13. Расходы на амортизацию и электроэнергию приведены в таблице 14. Суммарные затраты на создание программы приведены в таблице 15.

Таблица 13. Расчет прямой производственной зарплаты

Минимальный размер оплаты труда (МРОТ)	4 330 рублей
Отчисления на социальное обеспечение на текущий момент (в виде единого социального налога), $K_{ОТЧ}$	26%
Дополнительная заработная плата на предприятии, $K_{ДОП}$	50%
Тарифный коэффициент разряда разработчиков (11 разряд)	2,68
Тарифный коэффициент разряда научного руководителя (14 разряд)	3,36
Заработная плата разработчика/отладчика	11 604 рубля
Заработная плата научного руководителя	14 549 рублей
Время на разработку алгоритма, $T_{АЛГ}$	2 месяца
Время на разработку компонентов программы, $T_{РЕШ}$	2 месяца
Время на отладку и тестирование программы, $T_{ОТЛ}$	1 месяц
Время на консультационную поддержку со стороны научного руководителя, $T_{КОНС}$	1 месяц
Заработная плата за время создания алгоритма, $C_{АЛГ}(K_{АЛГ})$	43 863 рубля
Заработная плата разработчика за время разработки программы, $C_{РЕШ}$	43 863 рубля

Заработная плата отладчика за время отладки программы, $C_{ОТЛ}$	21 932 рубля
Затраты на консультационную поддержку, $C_{КОНС}$ ($К_{КОНС}$)	27 498 рублей
Итого заработная плата исполнителей с ЕСН	137 156 рублей

Таблица 14. Расчет затрат на материалы, комплектующие, энергию

Эффективное время разработки программного продукта (программист работает 5 дней в неделю), $Т_{ЭФ}$	192 часа/месяц
Длительность рабочего дня	8 часов
Рабочих дней в году	240 дней
Количество рабочих часов в году	1920 часов
Время на разработку компонентов программы, $Т_{РЕШ}$	2 месяца
Время на отладку и тестирование программы, $Т_{ОТЛ}$	1 месяц
Стоимость ПЭВМ для разработки и тестирования программного продукта, $Ц_{ПЭВМ}$	21 000 рублей
Срок службы ПЭВМ, $Т_{СЛ}$	3 года
Стоимость 1 кВт/ч электроэнергии, $ЭЛ$	2,31 рублей
Потребляемая мощность ПЭВМ	1 кВт/ч
Количество ПЭВМ для отладки программы	1
Сумма амортизационных отчислений за время разработки программы, $А_{РЕШ}$	1 400 рублей
Сумма амортизационных отчислений за время отладки программы, $А_{ОТЛ}$	700 рублей
Расходы на электроэнергию за время разработки программы, $З_{ЭЛ.РЕШ}$	887 рублей
Расходы на электроэнергию за время отладки программы, $З_{ЭЛ.ОТЛ}$	444 рубля
Расходные материалы (ручки, карандаши, бумага, краска для принтера)	500 рублей
Итого	6 051 рубль

Таблица 15. Смета общих затрат

Заработная плата за время создания алгоритма, $C_{АЛГ}$	43 863 рубля
Заработная плата разработчика за время разработки программы, $C_{РЕШ}$	43 863 рубля
Заработная плата отладчика за время отладки программы, $C_{ОТЛ}$	21 932 рубля
Затраты на консультационную поддержку, $C_{КОНС}$	27 498 рублей
Сумма амортизационных отчислений за время разработки программы, $A_{РЕШ}$	1 400 рубля
Сумма амортизационных отчислений за время отладки программы, $A_{ОТЛ}$	700 рублей
Расходы на электроэнергию за время разработки программы, $З_{ЭЛ.РЕШ}$	887 рублей
Расходы на электроэнергию за время отладки программы, $З_{ЭЛ.ОТЛ}$	444 рубля
Затраты на создание алгоритма, $К_{АЛГ}$	43 863 рубля
Затраты на разработку программы, $К_{РЕШ}$	46 150 рублей
Затраты на отладку программы, $К_{ОТЛ}$	23 076 рубля
Затраты на консультационную поддержку, $К_{КОНС}$	27 498 рублей
Сумма единовременных затрат, $К$	140 520 рублей
Накладные расходы 140% от основной ЗП	192 018 рублей
Затраты, $З$	332 538 рублей

4.2. План расчета затрат на создание программного продукта

Затраты рассчитываются по формуле

$$З = К+И \qquad (9)$$

где К — единовременные затраты заказчика; И — текущие издержки заказчика.

$$К = К_{АЛГ} + К_{ОТЛ} + К_{РЕШ} + К_{КОНС} \qquad (10)$$

где $К_{АЛГ}$ — затраты на создание алгоритма; $К_{РЕШ}$ — затраты на разработку компонентов программного продукта; $К_{ОТЛ}$ — затраты на отладку и тестирование компонентов программного

продукта; $К_{КОНС}$ – затраты на консультационную поддержку научным руководителем. Составляющие $К_{АЛГ}$, $К_{РЕШ}$, $К_{ОТЛ}$, $К_{КОНС}$ включают оплату труда разработчика, отладчика и консультанта.

Затраты на создание алгоритма включают заработную плату за период разработки алгоритма с учетом начислений на фонд оплаты труда (коэффициент 1,26). Расчет заработной платы в организации производится на основе МРОТ и коэффициента разряда разработчика (как в государственных организациях), но с учетом дополнительной заработной платы 50%, гибко распределяемой в качестве премиальных выплат (коэффициент 1,5):

$$К_{АЛГ} = ЗП*1,26*1,5*Т_{АЛГ} \quad (11)$$

где $Т_{АЛГ}$ – время разработки программы, $ЗП$ – заработная плата разработчиков, вычисляемая по формуле:

$$ЗП = МРОТ*Р \quad (12)$$

$МРОТ$ – текущая величина минимального размера оплаты труда, $Р$ – коэффициент разряда разработчика.

Затраты на разработку компонентов программного продукта:

$$К_{РЕШ}=А_{РЕШ}+С_{РЕШ}+З_{ЭЛ} \quad (13)$$

где $А_{РЕШ}$ – сумма амортизационных отчислений за период разработки программы, $С_{РЕШ}$ – ставка разработчика за этот период, $З_{ЭЛ}$ – затраты на электроэнергию.

$$А_{РЕШ} = (Ц_{ПЭВМ}/Т_{СЛ}*Т_{ЭФ.ПЭВМ})*Т_{РЕШ}*Т_{ЭФ.РЕШ} \quad (14)$$

где $Ц_{ПЭВМ}$ – балансовая стоимость ПЭВМ, $Т_{СЛ}$ – срок службы ПЭВМ в годах, $Т_{ЭФ.ПЭВМ}$ – эффективный фонд времени ПЭВМ в году (240 рабочих дней * 8 ч/день = 1920 часов), $Т_{РЕШ}$ – длительность разработки программы в месяцах, $Т_{ЭФ.РЕШ}$ – эффективный фонд времени разработки в месяц.

Ставка разработчика за период разработки рассчитывается следующим образом:

$$С_{РЕШ}=ЗП*1,26*1,5*Т_{РЕШ} \quad (15)$$

Затраты на электроэнергию:

$$З_{ЭЛ} = ЭЛ*Т_{РЕШ}*Т_{ЭФ.РЕШ} \quad (16)$$

Затраты на отладку и тестирование компонентов программного продукта (аналогично):

$$К_{ОТЛ} = ЭЛ* Ч_{ПЭВМ.ОТЛ.}*Т_{ОТЛ}*Т_{ЭФ.\ ОТЛ} + ЗП*1,26*1,5*Т_{ОТЛ} +$$
$$+(Ч_{ПЭВМ.ОТЛ.}*Ц_{ПЭВМ}/Т_{СЛ}*Т_{ЭФ.ПЭВМ})*Т_{ОТЛ}*Т_{ЭФ.\ ОТЛ} \quad (17)$$

$Ч_{ПЭВМ.ОТЛ.}$ – количество ПЭВМ, необходимое для отладки и тестирования программного продукта (поскольку продукт представляет из себя многокомпонентную распределенную

систему, применяемую в вычислительных сетях, для его тестирования необходимо несколько ПЭВМ в соответствии с утвержденной программой и методикой испытаний).

Затраты на консультационную поддержку вычисляются аналогично затратам на создание алгоритма с учетом повышенного тарифного коэффициента:

$$К_{КОНС} = ЗП*1,26*1,5*Т_{КОНС} \quad (18)$$

где $Т_{КОНС}$ – время на консультационную поддержку, $ЗП$ – заработная плата консультанта, вычисленная по формуле (4) с учетом коэффициента $Р_{КОНС}$.

4.3. Пояснения к расчету затрат на создание программного продукта

Заработная плата одного разработчика, вычисленная по формуле (13), составит

$$ЗП = 4\ 330*2,68 = 11\ 604 \text{ рубля}$$

Затраты на создание алгоритма (формула (3)):

$$К_{АЛГ} = 11\ 604*1,26*1.5*2=43\ 863 \text{ рубля}$$

Сумма амортизационных отчислений за время разработки программы (формула (15)):

$$А_{РЕШ} = (21\ 000/(3*1\ 920))*2*192=1\ 400 \text{ рубля}$$

Заработная плата разработчика за время разработки программы (формула (16)):

$$С_{РЕШ}=11\ 604*1,26*1.5*2=43\ 863 \text{ рубля}$$

Расходы на электроэнергию за время разработки программы (формула (18)):

$$З_{ЭЛ.РЕШ.} = 2,31*2*192=887 \text{ рублей}$$

Затраты на разработку компонентов программы:

$$К_{РЕШ} = А_{РЕШ}+С_{РЕШ}+З_{ЭЛ.РЕШ} =1\ 400+43\ 863+887=46\ 150 \text{ рубля}$$

Сумма амортизационных отчислений за время отладки программы (для отладки и тестирования в соответствии с методикой испытаний исследуемой системы необходимо 1 ПЭВМ и дополнительное сетевое оборудование, стоимость которого здесь включается в стоимость ПЭВМ):

$$А_{ОТЛ} = (21\ 000/(3*1\ 920))*1*192=700 \text{ рублей}$$

Заработная плата отладчика за время отладки программы:

$$С_{ОТЛ} =11\ 604*1,26*1.5*1=21\ 932 \text{ рубля}$$

Расходы на электроэнергию за время отладки программы (формула (17)):

$$З_{ЭЛ.ОТЛ} = 2,31*(3*1)*192=444 \text{ рубля}$$

Затраты на отладку и тестирование компонентов программы:

$$К_{ОТЛ} = А_{ОТЛ}+С_{ОТЛ}+З_{ЭЛ.ОТЛ} =700+21\ 932+444=23\ 076 \text{ рублей}$$

Заработная плата научного руководителя, вычисленная по формуле (13), составит

$$ЗП = 4\ 330*3,36=14\ 549\ рублей$$

Затраты на консультационную поддержку (формула (15)):

$$К_{КОНС} = 14\ 549*1,26*1.5 = 27\ 498\ рублей$$

Таким образом, сумма единовременных затрат составит (формула (10)):

$$К = 43\ 863+46\ 083+23\ 076+27\ 498=140\ 520\ рублей$$

С учётом издержек получаем по формуле (9) затраты:

$$З = 140\ 520 + 192\ 018 * 0,6 = 332\ 538\ рублей$$

5. ЗАКЛЮЧЕНИЕ

1. В результате проведенных экспериментов по идентификации сетевого трафика прикладного уровня были рассмотрены несколько алгоритмов классификации и определен алгоритм, допустивший при своей работе наименьшее число ошибок (неправильно классифицированных сетевых пакетов). Таким алгоритмом стал J4.8, показавший точность в 98.71% и обошедший своих конкурентов по классификации: OneR, SVM и Naïve Bayes. Такой высокий процент правильно классифицированных пакетов алгоритмом J4.8 объясняется универсальностью подхода, который в нем используется. Надо отметить, что в качестве атрибутов классификации по условию задачи использовались свойства пакетов только транспортного (TCP-протокол) и сетевого (IP-протокол) уровней, поэтому наилучшие результаты принадлежат универсальным алгоритмам, способным хорошо работать также с небольшим количеством параметров. Достаточно плохие результаты показал алгоритм классификации Naïve Bayes. Этот метод отличается простотой реализации и низкой вычислительной затратой при обучении и классификации, но требует независимости признаков.

2. В ходе выполнения работы были рассмотрены алгоритмы сокращения числа признаков. Наилучшие результаты показал метод PCA, который в отличие от методов InfoGain, Wrapper и CFS, генерирует меньшее количество новых признаков на основе уже существующих без существенной потери информации.

3. Проведенные эксперименты показали, что использование алгоритмов кластеризации EM и k-средних для выделения групп протоколов не применимы в условиях использования небольшого числа параметров.

4. Проведенное исследование может послужить основой для дальнейшего изучения этой области. В качестве перспективных направлений идентификации сетевого трафика можно выделить:
    - идентификация трафика в реальном времени;
    - захват потоков, вместо отдельных пакетов;
    - увеличение числа атрибутов (рассмотрение производных аргументов).

ПРИЛОЖЕНИЕ 1. ОБЗОР СОВРЕМЕННОГО СОСТОЯНИЯ ПРОБЛЕМЫ

Таблица 16. Обзор современного состояния проблемы

МО алгоритм	Характеристики	Трафик	Уровень классификации
Максимизация ожидания [53]	- Статистика длины пакета (минимум, максимум, квартили и т.д.) - Статистики межпакетных интервалов - Количество байт - Длительность соединения - Число переходов между транзакционным режимом и общим режимом передачи - Время простаивания Рассчитывается на полных потоках	Смесь HTTP, SMTP, FTP (управление), NTP, IMAP, DNS …	Крупная грануляция (общая передача, небольшие транзакции, составные транзакции ..)
AutoClass [54]	- Статистики длины пакета (среднее и отклонение в прямом и обратном направлениях) - Статистики межпакетного временного интервала (среднее и отклонение в прямом и обратном направлениях) - Длина потока (байты) - Длительность потока Рассчитывается на полных потоках	Half-Life, Napster, AOL, HTTP, DNS, SMTP, Telnet, FTP (данные)	Хорошая гранулярность (изучено 8 приложений)
Метод ближайших соседей, линейный дискриминантный анализ и квадратичный дискриминантный анализ [7]	- Пакетный уровень - Потоковый уровень - Уровень соединения - Внутри поточные/соединительные характеристики - Сложные потоковые характеристики Рассчитывается на полных потоках	Telnet, FTP (данные), Kazaa, Медиа поток реального времени, DNS, HTTPS	Хорошая грануляция (3, 4 и 7 классов отдельных приложений)
Технологии Байеса (“наивный” Байес, “наивный” Байес с ядерной оценкой, метод быстрой фильтрации на основе корреляции) [55]	Всего 248 характеристик, среди них: - Длительность потока - TCP порт - Статистика межпакетного временного интервала - Эффективная полоса пропускания, основанная на энтропии	Большой диапазон БД, P2P, Buck, Mail, Сервисы, … трафик	Крупная грануляция

	- Преобразование Фурье межпакетного временного интервала Рассчитывается на полных потоках		
Простой метод k-средних [56]	Длина пакета первых нескольких пакетов потоков трафика в обоих направлениях	eDonkey, FTP, HTTP, Kazaa, NTP, POP3, SMTP, SSH, HTTPS, POP3S	Хорошая грануляция (изучено 10 приложений)
“наивный” Байес с ядерной оценкой, дерево решений J48 и приведенная погрешность отсекаемого дерева [57]	- Длительность потока - Начальное байтовое объявление окна - Число фактических пакетов с данными - Число пакетов с опцией PUSH - Длины пакета - Байтовое объявление окна - Межпакетный временной интервал - Размер всех разбиваемых пакетов	WWW, Telnet, Chat (Messenger), FTP, P2P (Kazaa, Gnutella), Multimedia, SMTP, POP, IMAP, NDS, Oracle, X11	Неизвестно (сравнительная работа)
“наивный” Байес с учителем [58]	- Длины пакета (минимальная, максимальная, средняя, стандартное распределение) - Статистика внутри пакетной длины (минимальная, максимальная, средняя, стандартное распределение) - Статистика межпакетного временного интервала (минимальная, максимальная, средняя, стандартное отклонение) - Подсчет через наименьшее число (например, 25 пакетов) последовательных пакетов (классификационное окно), взятых из разных точек времени жизни потока, где есть значительные изменения характеристик потока	Трафик онлайн игр (Enemy Territory), другой - (HTTP, HTTPS, DNS, NTP, SMTP, Telnet, SSH, P2P ...)	Специфичные приложения (Онлайн игры, основанный на UDP трафик, First Person Shooter, Enemy Territory)
“наивный” Байес и дерево выбора в комбинации с кластерными алгоритмами для автоматического выбора подпотоков [59]	- Длины пакета (минимальная, максимальная, средняя, стандартное распределение) - Статистика внутри пакетной длины (минимальная, максимальная, средняя, стандартное распределение) - Статистика межпакетного временного интервала (минимальная, максимальная, средняя, стандартное отклонение) - Подсчет через наименьшее число (например, 25 пакетов) последовательных пакетов (классификационное окно), взятых из разных точек времени жизни потока, где есть значительные изменения характеристик потока	Трафик онлайн игр (Enemy Territory), другой - (HTTP, HTTPS, DNS, NTP, SMTP, Telnet, SSH, P2P ...)	Специфичные приложения (Онлайн игры, основанный на UDP трафик, First Person Shooter, Enemy Territory)
K-средних [60]	- Общее число пакетов	Web, P2P, FTP, другие	Крупная грануляция (29

	- Средняя длина пакета - Средняя длина полезной нагрузки, исключая заголовки - Число переданных байт - Длительность потока - Среднее время межпакетного временного интервала		различных протоколов, сгруппированных по числу приложений)
Отпечатки протоколов (плотность распределения вероятности векторов) и оценка аномалий (от протокола ФРП по отпечаткам протокола) [61]	- Длины пакета - Межпакетный временной интервал - Очередность прибытия пакета	TCP приложения (HTTP, SMTP, POP3, SSH)	Хорошая грануляция (4 TCP протокола)
“наивный” Байес, AdaBoost, регуляризация максимальной энтропии [62]	Дискретное байтовое кодирование первых n-байт полезной нагрузки однонаправленного TCP потока	FTP (управление), SMTP, POP3, IMAP, HTTPS, HTTP, SSH	Хорошая грануляция
Обучение без учителя (произведение распределений, процессы Маркова, общие графические подцепочки) [63]	Дискретное байтовое кодирование первых n-байт полезной нагрузки однонаправленного TCP потока	FTP (управление), SMTP, POP3, IMAP, HTTPS, HTTP, SSH	Хорошая грануляция
Нейронная сеть Байеса [64]	246 характеристик, включающие: - Метрики потока (длительность, подсчет пакетов, всего байт) - Статистика по межпакетному временному интрвалу - Размер TCP/IP контрольных полей - Всего пакетов потока в каждом направлении и в оба направления - Размер полезной нагрузки - Эффективная полоса пропускания, основанная на энтропии - Верхняя десятка преобразования Фурье, состоящая из межпакетных временных интервалов для каждого направления - Многочисленные, специфичные для TCP значения, извлеченные из tcptrace (например, общая переданная полезная нагрузка в байтах, общее число PUSH пакетов, общее число ACK пакетов, переносимых SACK информацию и т.д.)		Крупная грануляция
“наивный” Байес с дискретизацией, “наивный” Байес с ядерной оценкой, C4.5 дерево решений, сеть “наиного” Байеса и дерево “наивного” Байеса [65]	- Протокол - Длительность потока - Объем потока в байтах и пакетах - Длина пакета (минимальная, средняя, максимальная,	FTP(данные), Telnet, SMTP, DNS, HTTP	Неизвестно (сравнительная работа)

	стандартное отклонение) - Межпакетный временной интервал (минимальный, средний, максимальный, стандартное отклонение)		
K-средних, DB-SCAN и AutoClass [66]	- Общее число пакетов - Средняя длина пакета - Средняя длина полезной нагрузки, исключая заголовки - Количество переданных байт (в каждом направлении и совместно) - Среднее значение межпакетного временного интервала	HTTP, P2P, SMTP, IMAP, POP3, MSSQL, другие	Неизвестно (сравнительная работа)
“наивный” Байес и AutoClass [67]	- Общее число пакетов - Средняя длина пакета - Средняя длина полезной нагрузки, исключая заголовки - Количество переданных байт (в каждом направлении и совместно) - Среднее значение межпакетного временного интервала	HTTP, SMTP, DNS, SOCKS, FTP(управление) FTP (данные), POP3, Limewire	Неизвестно (сравнительная работа)
“наивный” Байес и пирсоновский критерий хи-квадрат [68]	- Размер сообщения (длина сообщения, инкапсулированного в транспортный уровень сегмента протокола) - Средний межпакетный интервал	Skype	Особое приложение

СПИСОК ЛИТЕРАТУРЫ

1. “Snort – The de facto standard for intrusion detection/prevention”, August 14, 2007. Доступно с http://www.snort.org
2. “Bro intrusion detection system - Bro overview”, August 14, 2007. Доступно с http://bro-ids.org
3. V. Paxson, “Bro: A system for detecting network intruders in real-time,” Computer Networks, no. 31(23-24), pp. 2435–2463, 1999.
4. Tarek Abbes, Michael Rusinowitch, Alekesh Haloi. “Network Traffic Classification for Intrusion Detection.” Juin 2004.
5. Annie De Montigny-Leboeuf . “Flow Attributes For Use In Traffic Characterization”. Доступно с http://www.crc.gc.ca/files/crc/home/research/network/system_apps/network_systems/network_security/publications/ADeMontigny_CRCTN2005003.pdf
6. Internet Assigned Numbers Authority (IANA), August 14, 2007. Доступно с http://www.iana.org/assignments/port-numbers
7. M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, “Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification,” in Proceedings of ACM/SIGCOMM Internet Measurement Conference (IMC) 2004, Taormina, Sicily, Italy, October 2004.
8. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC 1889, IETF, 1996.
9. A. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” in Proc. Passive and Active Measurement Workshop (PAM2005), Boston, MA, USA, March/April 2005.
10. A. Madhukar and C. Williamson, “A longitudinal study of P2P traffic classification,” in 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, September 2006.
11. S. Sen, O. Spatscheck, and D. Wang, “Accurate, scalable in network identification of P2P traffic using application signatures,” in WWW2004, New York, NY, USA, May 2004.
12. F. Risso, A. Baldini, M. Baldi, P. Monclus, O. Morandi. “Lightweight, Session-Based Traffic Classification”.

13. Mario Baldi, Fulvio Risso. “NetPDL: An Extensible XML-Based Language for Packet Header Description”. In Elsevier Computer Networks Journal (COMNET), Volume 50, Issue 5, Pages 688-706, April 2006.
14. Cisco Systems. “Network Based Application Recognition (NBAR)”.
15. Opher Reviv. “Inside network programming with SML”. EE Times, August 2003
16. Ruoming Pang, Vern Paxson, Robin Sommer, Larry Peterson. “Binpac: a yacc for writing application protocol parsers”. In Proceedings of the 6th ACM SIGCOMM on Internet Measurement, pages 289-300, Rio de Janeiro, Brazil, October 2006
17. T. Karagiannis, K. Papagiannaki, and M. Faloutsos. “Blinc: Multilevel traffic classification in the dark”. In Proceedings of ACM SIGCOMM, Philadelphia, PA, August, 2005.
18. A Survey of Techniques for Internet Traffic Classification using Machine Learning
19. V. Paxson, “Empirically derived analytic models of wide-area TCP connections,” IEEE/ACM Transactions on Networking, vol. 2, no. 4, pp. 316–336, 1994.
20. C. Dewes, A. Wichmann, and A. Feldmann, “An analysis of Internet chat systems,” in ACM/SIGCOMM Internet Measurement Conference 2003, Miami, Florida, USA, October 2003.
21. K. Claffy, “Internet traffic characterisation,” PhD Thesis, University of California, San Diego, 1994.
22. T. Lang, G. Armitage, P. Branch, and H.-Y. Choo, “A synthetic traffic model for Half-life,” in Proceedings of Australian Telecommunications Networks and Applications Conference 2003 ATNAC2003, Melbourne, Australia, December 2003.
23. T. Lang, P. Branch, and G. Armitage, “A synthetic traffic model for Quake 3,” in Proceedings of ACM SIGCHI International Conference on Advances in computer entertainment technology (ACE2004), Singapore, June 2004.
24. Jeffrey Erman, Anirban Mahanti, and Martin Arlitt. Traffic classification using flow information. In Proceedings of the 2006 SIGCOMM workshop on Mining network data, 2006.
25. Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In Proceedings of the 2006 SIGCOMM workshop on Mining network data, 2006.
26. S. Zander, T. Nguyen, and G. Armitage, “Automated traffic classification and application identification using machine learning,” in IEEE 30th Conference on Local Computer Networks (LCN 2005), Sydney, Australia, November 2005.

27. J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, “Identifying and discriminating between web and peer-to-peer traffic in the network core,” in WWW ’07: Proceedings of the 16th international conference on World Wide Web. Banff, Alberta, Canada: ACM Press, May 2007, pp. 883–892.
28. Официальный сайт Weka: http://www.cs.waikato.ac.nz/ml/weka/
29. Официальный сайт Wireshark: http://www.wireshark.org
30. Официальный сайт TcpDump: http://www.tcpdump.org
31. Официальный сайт pcap2mysql: http://sourceforge.net/projects/pcap2mysql
32. Официальный сайт MySQL: http://www.mysql.com
33. RFC4180. Доступно с http://www.rfc-editor.org/rfc/rfc4180.txt
34. Mark A. Hall. Correlation-based Feature Selection for Machine Learning, April 1999.
35. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification.
36. M. Hall and G. Holmes, “Benchmarking attribute selection techniques for discrete class data mining,” IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 6, pp. 1437– 1447, 2003.
37. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
38. R. Kohavi and G. H. John, “Wrappers for feature subset selection,” Artificial Intelligent, vol. 97, no. 1-2, pp. 273–324, 1997.
39. P. H. Winston, Artificial intelligence (2nd ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984.
40. J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA., 1993.
41. R. Kohavi and G. H. John, “Wrappers for feature subset selection,” Artificial Intelligent, vol. 97, no. 1-2, pp. 273–324, 1997.
42. H. D. Fisher, J. M. Pazzani, and P. Langley, Concept Formation: Knowledge and Experience in Unsupervised Learning. Morgan Kaufmann, 1991.
43. I. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations (Second Edition). Morgan Kaufmann Publishers, 2005.
44. Технический справочник Майкрософт по алгоритмам кластеризации. Доступно с *msdn.microsoft.com/ru-ru/library/cc280445.asp*
45. W. Rand, “Objective criteria for the evaluation of clustering methods,” Journal of the American Statistical Association, vol. 66, no. 336, pp. 846–850, 1971.

46. M. Halkidi, Y. Batistakis, and M. Vazirgiannis, “Cluster validity methods: part I,” SIGMOD Rec., vol. 31, no. 2, pp. 40–45, 2002.
47. R. Xu and D. Wunsch, “Survey of clustering algorithms,” IEEE Transactions on Neural Networks, no. Vol.16, Issue 3, pp. 645– 678, May 2005.
48. M. Halkidi, Y. Batistakis, and M. Vazirgiannis, “Clustering validity checking methods: part II,” SIGMOD Rec., vol. 31, no. 3, pp. 19–27, 2002.
49. Y. Reich and J. S. Fenves, “The formation and use of abstract concepts in design,” in Fisher, D. H. and Pazzani, M. J. (editors), Concept Formation: Knowledge and Experience in Unsupervised Learning. Morgan Kaufmann, 1991.
50. А.А.Барсегян, М.С.Куприянов. “Методы и модели анализа данных OLAP, Data Mining”.
51. Кечиев Л. Н. Элементы эргономической безопасности работы с компьютерами: Учебное пособие / Л. Н. Кечиев, И. И. Литвак — М.: МГИЭМ, 1997.
52. Горчакова Л.И. Экономические расчеты в дипломных проектах по техническим специальностям: Метод. указания для студентов ФТК. СПбГПУ, 2003.
53. A. McGregor, M. Hall, P. Lorier, and J. Brunskill, “Flow clustering using machine learning techniques,” in Proc. Passive and Active Measurement Workshop (PAM2004), Antibes Juan-les-Pins, France, April 2004.
54. S. Zander, T. Nguyen, and G. Armitage, “Automated traffic classification and application identification using machine learning,” in IEEE 30th Conference on Local Computer Networks (LCN 2005), Sydney, Australia, November 2005.
55. A. Moore and D. Zuev, “Internet traffic classification using Bayesian analysis techniques,” in ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) 2005, Banff, Al-berta, Canada, June 2005.
56. L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, “Traffic classification on the fly,” ACM Special Interest Group on Data Communication (SIGCOMM) Computer Communication Review, vol. 36, no. 2, 2006.
57. J. Park, H.-R. Tyan, and K. C.-C.J., “Internet traffic classification for scalable QoS provision,” in IEEE International Conference on Multimedia and Expo, Toronto, Ontario, Canada, July 2006.
58. T. Nguyen and G. Armitage, “Training on multiple sub-flows to optimize the use of Machine Learning classifiers in real-world IP networks,” in Proc. IEEE 31st Conference on Local Computer Networks, Tampa, Florida, USA, November 2006.

59. T. Nguyen and G. Armitage, “Synthetic sub-flow pairs for timely and stable IP traffic identification,” in Proc. Australian Telecommunication Networks and Application Conference, Melbourne, Australia, December 2006.
60. J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, “Identifying and discriminating between web and peer-to-peer traffic in the network core,” in WWW ’07: Proceedings of the 16th international conference on World Wide Web. Banff, Alberta, Canada: ACM Press, May 2007, pp. 883–892.
61. M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, “Traffic classification through simple statistical fingerprinting,” SIGCOMM Comput. Commun. Rev., vol. 37, no. 1, pp. 5–16, 2007.
62. P. Haffner, S. Sen, O. Spatscheck, and D. Wang, “ACAS: automated construction of application signatures,” in MineNet ’05: Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data. Philadelphia, Pennsylvania, USA: ACM Press, August 2005, pp. 197–202.
63. J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. Voelker, “Unexpected means of protocol inference,” in IMC ’06: Proceedings of the 6th ACM SIGCOMM on Internet measurement. Rio de Janeriro, Brazil: ACM Press, October 2006, pp. 313–326.
64. T. Auld, A. W. Moore, and S. F. Gull, “Bayesian neural networks for Internet traffic classification,” IEEE Transactions on Neural Networks, no. 1, pp. 223–239, January 2007.
65. N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification,” Special Interest Group on Data Communication (SIGCOMM) Computer Communication Review, vol. 36, no. 5, pp. 5–16, 2006.
66. J. Erman, M. Arlitt, and A. Mahanti, “Traffic classification using clustering algorithms,” in MineNet ’06: Proceedings of the 2006 SIGCOMM workshop on Mining network data. New York, NY, USA: ACM Press, 2006, pp. 281–286.
67. J. Erman, A. Mahanti, and M. Arlitt, “Internet traffic identification using machine learning techniques,” in Proc. of 49th IEEE Global Telecommunications Conference (GLOBECOM 2006), San Francisco, USA, December 2006.
68. D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, “Revealing Skype traffic: when randomness plays with you,” in SIGCOMM ’07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications. New York, NY, USA: ACM, August 2007, pp. 37–48.