Содержание

1 Введение

1.1 Постановка задачи

1.1.1 Передача данных по сети

Современные сети Интернет объединяют в единое целое многие десятки и более тысяч локальных сетей по всему миру, построенных на базе самых разных физических и логических протоколов. Эти сети объединяются друг с другом с помощью последовательных каналов, сетей ATM, SDH , Fibre Channel и многих других. В самих сетях используются протоколы TCP/IP (Интернет), IPX/SPX, Appletalk, Netbios и множество других, признанных международными, являющихся фирменными и т.д. Нельзя не отметить многообразие сетевых программных продуктов. На следующем уровне представлены разнообразные внутренние (RIP, IGRP, OSPF) и внешние (BGP и т.д.) протоколы маршрутизации и маршрутной политики, конфигурация сети и задание огромного числа параметров, проблемы диагностики и сетевой безопасности.

Рост сетевого трафика во многом обусловлен тем, что создатели базовых протоколов (TCP/IP) заложили в них несколько простых и эффективных принципов: инкапсуляцию пакетов, фрагментацию/дефрагментацию сообщений и динамическую маршрутизацию путей доставки. Именно эти идеи позволили объединить сети, базирующиеся на самых разных операционных системах (Windows, Unix, Sunos/Solaris и пр.), использующих различное оборудование (Ethernet, Token Ring, FDDI, ISDN, ATM, SDH и т.д.) и сделать сеть нечувствительной к локальным отказам аппаратуры [2].

1.1.2 Ботнеты и вирусные программы

Бот - это экземпляр вредоносного ПО, работающий на зараженном компьютере автономно и без ведома пользователя [3]. Как правило бот обладает обширными функциональными возможностями, которые позволяют ему выполнять большой набор вредоносных действий.

Ботнет - это сеть компьютеров, которые удаленно управляются хакером (бот-мастером). Криминальную деятельность ботнет-сетей можно разделить на следующие типы:

- Распределенные атаки типа «отказ в обслуживании» (Distributed Denial of Service, DDOS)

  Ботнету может быть отдана команда совершить целенаправленную, распределенную атаку типа «отказ в обслуживании» на любую систему в Интернете с целью поглотить ресурсы (например, пропускная способность) системы таким образом, что она не сможет должным обра-

зом обслуживать своих легитимных пользователей. В настоящее время практически все DDoS-атаки осуществляются с платформы ботнетов (рис. 1).

- Рассылка спама

  Более 95% электронной почты в сети Интернет является спамом, что составляет несколько миллиардов сообщений спама в интернет трафике ежедневно. Большинство этих сообщений спама, на самом деле, отправлены из ботнетов.

- Фишинг

  Ботнеты широко используются для размещения вредоносных поддельных сайтов. Обычно преступники рассылают сообщения спама (например, с использованием ботнетов) с целью обманом заманить пользователя посетить поддельные сайты (как правило, связанные с финансовой деятельностью – интернет банкингом). Таким образом, преступники могут получить доступ к конфиденциальной информации пользователей, такой как имена пользователей, пароли и номера кредитных карт. По данным "Лаборатории Касперского"в 3 квартале 2019 года было предотвращено 105 220 094 попытки подобных атак [4].

- Кликфрод

  Ботмастер может получать прибыль от управления кликами ботов на онлайн объявления (то есть посылать HTTP запросы на веб-страницы рекламодателя) с целью личной или коммерческой выгоды. Кликфрод может использоваться для повышения рейтинга веб-сайтов в поисковых системах.

- Кража информации

  Боты активно используются для кражи конфиденциальной информации, такой как номера кредитных карт, пароли или ключи авторизации на локальном компьютере пользователя.

- Распространение других нежелательных программ

  Например, рекламное/шпионское программное обеспечение. Ботнеты являются хорошей платформой для распространения множества других форм вредоносного ПО

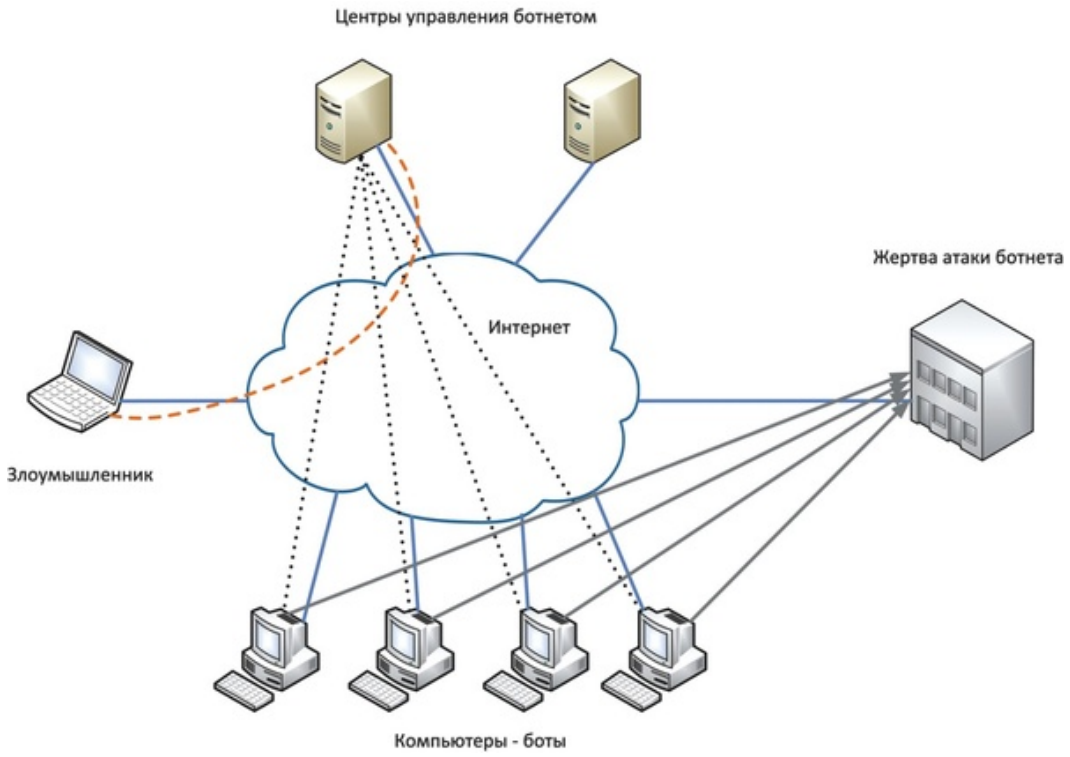

Рис. 1: Схема Ботнет сети

1.1.3 Системы обнаружения вредоносного ПО

В целях защиты информации, для устранения или предотвращения появления угрозы безопасности используются различные средства обеспечения безопасности. Главное свойство этих средств - это адекватность угрозам и снижение уровня их опасности. Можно выделить два варианта защиты критически важных ресурсов [5]:

Первый - система обнаружения вторжений (СОВ, IDS). Под такой системой понимается решение, позволяющее выявлять факты неавторизованного доступа в компьютерную систему или сеть. СОВ обнаруживает атаки (но не блокирует их) и оповещает сотрудников службы безопасности об атаках.

Второй вариант - система обнаружения и предотвращения вторжений (СОПВ, IPS). Это решение, которое обеспечивает дополнительный уровень защиты компьютерных систем. Такие системы не только выявляет факты несанкционированного доступа в корпоративную сеть, но и автоматически блокирует сетевые атаки и вредоносное ПО (рис. 2).

Мониторинг деятельности в системе производится с помощью детекторов вторжений. Они сверяют события с созданным шаблоном, который является описанием какого-либо инцидента (например botnet-атаки). Шаблон, связанный с определённым инцидентом называется "сигнатурой". Для инцидентов существует термин «сигнатурное определение». Сигнатура вторжения – признаки и атрибуты инцидентов, необходимые для их выявления.

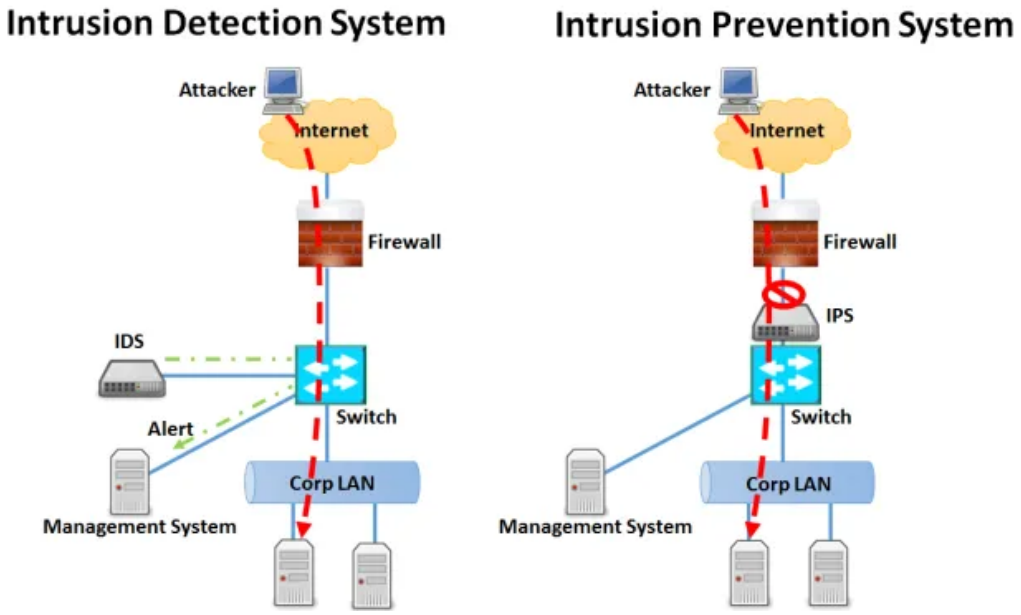

Рис. 2: СОВ и СОПВ

Большая часть антивирусных программ используют «синтаксические» сигнатуры, заимствованными прямо из кода вторжения (пакета, являющегося частью вредоносной программы). Задача СОВ - быстро найти аналогичные коды в банке сигнатур.

Однако, несмотря на точность определения известных аномалий, сигнатурные методы требуют постоянной и своевременной актуализации данных баз данных. Этот недостаток не позволяет находить новые угрозы и делает системы, построенные на сигнатурных методах неустойчивыми к минимальным изменениям в поведении вирусного ПО.

Поэтому, все чаще рассматриваются не сигнатурные методы поиска вторжений, а поведенческие. Эти методы основаны на выявлении поведенческих шаблонов, по которым действует вредоносное ПО. Многие работы в этой области опираются на использовании методов машинного обучения и нейронных сетей, поскольку они успели успешно зарекомендовать себя в задачах анализа поведения сложных систем.

1.2 Цель работы

Целью данной работы является поиск методов машинного обучения, подходящих для обнаружения вирусного трафика. Найденный алгоритм может быть использован в качестве альтернативы сигнатурным и статистическим методам анализа, которые используются сейчас.

В работе рассматриваются несколько наборов данных, размещенных в сети проектом stratosphere ips. Для этих данных были применены общепринятые методы предобработки и подготовки данных к обучению модели

(нормировки, кодирование категориальных признаков и пр.). Обученные модели были протестированы на отложенных выборках, были подсчитаны метрики качества. Помимо этого, было рассмотрено использование алгоритмов кластеризации для выявление кластеров с вредоносными пакетами в веб-трафике.

2 Обзор литературы

2.1 Структура сетевого трафика

Усложнение схем сетей и многообразие сетевых устройств привели к усложнению их настройки и поддержки сети в работоспособном состоянии. По этой причине, был необходим инструмент позволяющий, с одной стороны локализовать проблему, а с другой предоставить как можно более исчерпывающую информацию о природе проблемы. Это привело к зарождению технологий анализа сетевого трафика. [6].

Объектом, который содержит в себе всю необходимую информацию и является сетевой трафик. Одним из инструментов, изначально предназначенным для анализа трафика стал сетевой анализатор Wireshark (ранее Ethereal), созданный инженером Джеральдом Комбом в 1997 году. Wireshark продолжает активно развиваться и является стандартом в определённой области сетевого анализа [7].

Wireshark проводит анализ данных в трафике в рамках модели OSI.

Модель OSI чётко определяет уровни взаимодействия систем, стандартизует имена уровней и указывает услуги и функции каждого уровня.

Одним из важнейших принципов OSI является то, что сетевые системы взаи- модействуют друг с другом на одинаковых уровнях модели.

В модели OSI выделяют несколько уровней [8] (рис. 3):

- Физический уровень (Physical Layer).

  Обеспечивает передачу битовых потоков без каких-либо изменений между логическими объектами уровня звена данных по физическим соединениям. На данном уровне определяются базовые механизмы кодирования и декодирования двоичных данных в физическом носителе, а также специфицируются соединители, но не сама среда. Среда, согласно эталонной модели, рассматривается как нечто, лежащее ниже физического уровня. Битовый поток в носителе должен быть независим от типа среды.

- Канальный уровень (Data Link Layer)

  Канальный уровень обеспечивает функциональные и процедурные средства для установления, поддержания и разрыва соединений канального уровня между сетевыми логическими объектами и для передачи сервисных блоков данных это- го уровня. Соединение канального уровня строится на основе одного или нескольких физических соединений. Канальный уровень обнаруживает и по возможности исправляет ошибки,

которые могут возникнуть на физическом уровне. Кроме того, канальный уровень обеспечивает для сетевого уровня возможность управлять подключением кана- лов данных на физическом уровне. Единицу информации на канальном уровне называют кадром или фреймом.

- Сетевой уровень (Network Layer)

  Предоставляет средства установления, поддержания и разрыва сетевого соединения, а также функциональные и процедурные средства для обмена по сетевому соединению сетевыми сервисными блоками данных между транспортными логическими объектами. Сетевой уровень обеспечивает транспортным логическим объектам независимость от функций маршрутизации и ретрансляции, связанных с процессами установления и функционирования данного сетевого соединения. Все функции ретрансляции и расширенные протоколы последовательного переноса данных, которые предназначены для поддержания сетевых услуг между оконечными открытыми системами, функционируют ниже транспортного уровня. Единицу информации на сетевом уровне называют датаграммой.

- Транспортный уровень (Transport Layer)

  Обеспечивает передачу данных без каких-либо изменений между сеансовыми логическими объектами и освобож- дает их от выполнения операций, обеспечивающих надёжную и экономически эффективную передачу данных.

- Сеансовый уровень (Session Layer)

  Реализует службу имён (отображение логических имён в сетевые адреса), устанавливает сеансы между службами и создаёт точки для контрольной синхронизации в случае потери связи.

- Уровень представления (Presentational Layer)

  Устанавливает способы пред- ставления информации, которой обмениваются прикладные логические объекты или на которую они ссылаются в процессе этого обмена.

- Прикладной уровень (Application Layer)

  Является наивысшим уровнем в эталонной модели OSI. Поэтому прикладной уровень не имеет интерфейса с более высоким уровнем. Он

является единственным средством доступа прикладных процессов к функциональной среде OSI. Единицу информации на прикладном уровне называют сообщением (Message).

На прикладном уровне выполняются все функции связи между открытыми системами, которые не выполняются нижележащими уровнями. В их число включаются функции, выполняемые программными средствами, и функции, выполняемые людьми.

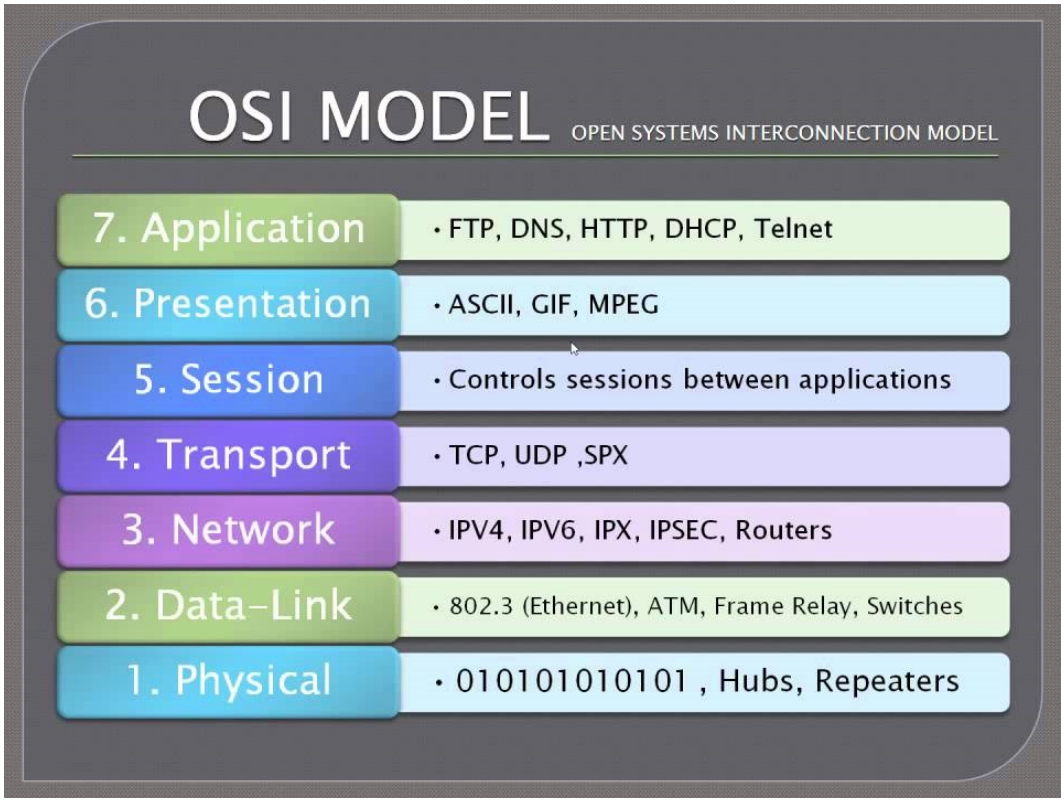

Рис. 3: Модель OSI

2.2 Обзор систем обнаружения (предотвращения) вторжений

Вирусное ПО находиться в процессе постоянного совершенствования, представляя все большую угрозу даже при наличии передового оборудования и программ для защиты от угроз. Существенный вклад среди всех угроз вносят угрозы производимые с использованием локальных и глобальных вычислительных сетей.

Процесс осуществления угроз получил название "атака"или "вторжение".

Атака – любое действие нарушителя, направленное на нарушение заданной функциональности вычислительной системы или получение несанкционированного доступа к информационным, вычислительным или сетевым ресурсам.

Можно выделить следующие этапы в проведении атаки:

1. Подготовка. Получение информации об объекте.
2. Вторжение. Получение доступа к объекту.
3. Атакующее воздействие. Реализуются цели, для которых была задумана атака.
4. Развитие атаки. Попытки распространения атаки на другие объекты.

Для предотвращения атак используют СОВ и СОПВ. Их целью является контроль над сетевыми ресурсами, возможность определения их аномального поведения или неправильного использования

Рассмотрим наиболее популярные из них.

Suricata На данный момент времени наиболее быстро развивающимся и эффективным проектом в области СОВ. Является проектом с открытым исходным кодом, распространяемым под лицензией GPLv2, разработкой которой занимается организация OpenIn-formationSecurityFoundation.

Принцип работы Suricata основан на прослушивании трафика в сети и его анализа на основе различных наборов правил. Данные наборы правил записываются в специальные файлы в широко используемом формате YAML, что дает возможность для обработки таких файлов в утилитах для анализа. Suricata может являться как системой обнаружения вторжений, так и системой предотвращения вторжений путем блокировки подозрительного трафика внутри сети.

Основные особенности [9]:

- Работа в многопоточном режиме
- Поддержка автоматического определения протоколов: IP, TCP, UDP, ICMP, HTTP, TLS, FTP и SMB. Пользователь системы имеет возможность определения типа протокола в правилах, без привязки к номеру порта
- Поддержка стандартных интерфейсов для перехвата трафика NFQueue, IPFRing, LibPcap, IPFW. Унифицированный формат вывода результатов проверки позволяет использовать стандартные утилиты для анализа
- Возможность использовать ускорение NVIDIA CUDA для своей работы

Snort Snort была создана в 1998 году. Очень быстро завоевала популярность, благодаря тому, что являлась бесплатной сетевой системой обнаружения и предотвращения вторжений, обладала открытым исходным кодом, позволяла в режиме реального времени анализировать трафик в сети, а так же позволяла самостоятельно писать дополнительные правила для обнаружения атак [10].

Систему обнаружения вторжений Snort по способу мониторинга системы можно отнести как к узловой, так и к сетевой системе в зависимости от параметров настройки. Обычно она защищает определённый сегмент локальной сети от внешних атак из интернета. При всем этом данная система является кросс-платформенным ПО.

СОВ Snort выполняет протоколирование, анализ, поиск по содержимому, а также широко используется для активного блокирования или пассивного обнаружения целого ряда нападений, таких как переполнение буфера, стелс-сканирование портов, атаки на веб-приложения, SMB-зондирование и попытки определения ОС. Программное обеспечение в основном используется для предотвращения проникновения, блокирования атак, если они имеют место [11].

Среди достоинств этой СОВ можно выделить следующие: легковесность, простота расширения правил, открытый исходный код.

Bro (Zeek) Bro (Zeek) - это СОВ и инструмент для мониторинга сетевой безопасности, который может выявлять аномалии, такие, как подозрительные или опасные действия вредоносного ПО. Zeek отличается от традиционных СОВ тем, что, в отличие от систем, основанных на правилах, выявляющих исключения, захватывает метаданные, связанные с происходящим в сети. Делается это для лучшего понимания контекста необычного сетевого поведения. Это позволяет, например, анализируя HTTP-трафик или процедуру обмена сертификатами безопасности, взглянуть на протокол, заголовки пакетов, доменные имена [12].

Табл. 1 отображает сводные данные по приведенным СОВ.

Таблица 1: Сравнительная таблица программных продуктов СОВ [13]

СОВ	Вид исполнения	Метод определения атаки	Поддерживаемые ОС
Snort	Сетевая	Сигнатурный с обратной связью	GNU/Linux, Windows
Suricata	Сетевая	Сигнатурный с обратной связью	GNU/Linux, Windows
Bro (Zeek)	Гибридная	Граф переходов, соответствующий атаке	Unix, GNU/Linux

2.3 Сигнатурные методы обнаружения вредоносного трафика

Сигнатурные методы работы заключаются в том, что циркулирующий сетевой трафик анализируется установленной системой обнаружения вторжений и производится сравнение с имеющейся базой данных сигнатур [14].

Можно выделить следующие сигнатурные подходы:

Методы контекстного поиска заключаются в обнаружении в исходной информации определенного множества символов. Так, например, для выявления атаки на Web-сервер под управлени- ем операционной системы семейства Unix, направленной на получение несанкционированного доступа к файлу паролей, проводится поиск последовательности символов "GET */etc/passwd"в заголовке HTTP-запроса.

Методы анализа состояний основаны на формировании сигнатуры атак в виде последовательности переходов ИС (информационной системы) из одного состояния в другое. По сути, каждый такой переход определяется по наступлению в ИС определенного события, а набор этих событий задается параметрами сигнатуры атаки [15].

2.4 Несигнатурные методы обнаружения вредоносного ПО

Несигнатурные методы (поведенческие) как правило основаны на использовании методов машинного или глубокого обучения. В качестве данных для алгоритмов используется информация из фреймов интернет-трафика, например payload, тип протокола, длина послдовательности, частота передачи пакетов для TCP трафика и тд.

В силу того, что такие подходы являются более универсальными чем сигнатурные методы, они все чаще используются для обнаружения вредоносных соединений. Так, В США уже зарегистрированы патенты на IDS, которые используют нейронные сети для обнаружения угроз [16].

2.5 Методы машинного обучения

Машинное обучение (Machine Learning, ML) — область искусственного интеллекта (ИИ), создающая алгоритмы, которые самостоятельно обучаются и прогнозируют ситуацию на основе имеющихся данных, не требуя вмешательства программиста. Применяется в таких областях, как исследование компьютерного зрения, борьба с онлайн-мошенничеством, прогнозирование цен и курсов, обработка естественного языка и пр. [17].

В машинном обучении выделяют 3 наиболее распространенных подхода построения моделей [18]:

- Обучение с учителем (Supervised learning)

  Все входные и выходные данные помечены, и алгоритмы учатся предсказывать выходные данные из входных данных

- Обучение с подкреплением (Reinforcement learning)

  Это наиболее общая форма обучения. В нем рассматривается вопрос о том, как выучить последовательность действий, называемых стратегией управления, из косвенной и отложенной информации о вознаграждении.

- Обучение без учителя (Unsupervised learning)

  Все данные не помечены, и алгоритмы учатся присваивать структуру из входных данных. Модель пытается изучить шаблоны на входе, для которых нет доступных выходных значений. Примером является кластеризация.

  Алгоритмы используют в качестве входных данных так называемые признаки (features). Признаки - это предобработанные данные. В случае текстовой информации данные переводятся в вектора признаков на основе метода bag of words или Count Vectoriser. В случае табличных данных применяется нормализация данных - приведение всех численных данных к одному масштабу. В случае категориальных признаков - производится кодирование целым числом либо другим интерпретируемым численным значением.

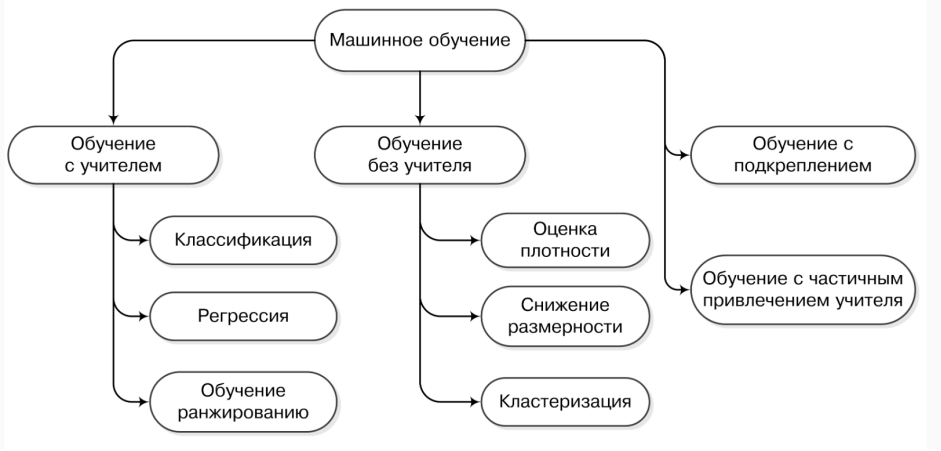

Рис. 4: Распространенные подходы построения моделей в машинном обучении

2.5.1 Распространенные алгоритмы

В области построения моделей при обучении с учителем можно выделить 2 больших класса задач, которые наиболее часто встречаются в различных приложениях. Первый - это классификация. В задачах классификации выходы алгоритмов дискретны и, как правило, характеризуют интерпретируемые различия, которые есть в данных. Второй класс - это задачи регрессии. Они отличаются от классификации тем, что выходы алгоритмов представляют собой действительные числа. Примером задачи регрессии может служить прогнозирование дневной температуры, а задачи классификации - определение наличия рака молочной железы.

Как правило, для решения задач классификации или регрессии рассматривается устоявшийся набор моделей, которые успешно зарекомендовали себя в различных приложениях, например при решении задач медицинской классификации, поиска аномалий, прогнозирования спроса в продуктовом ритейле и тд. Наиболее распространенные модели приведены на рис 5.

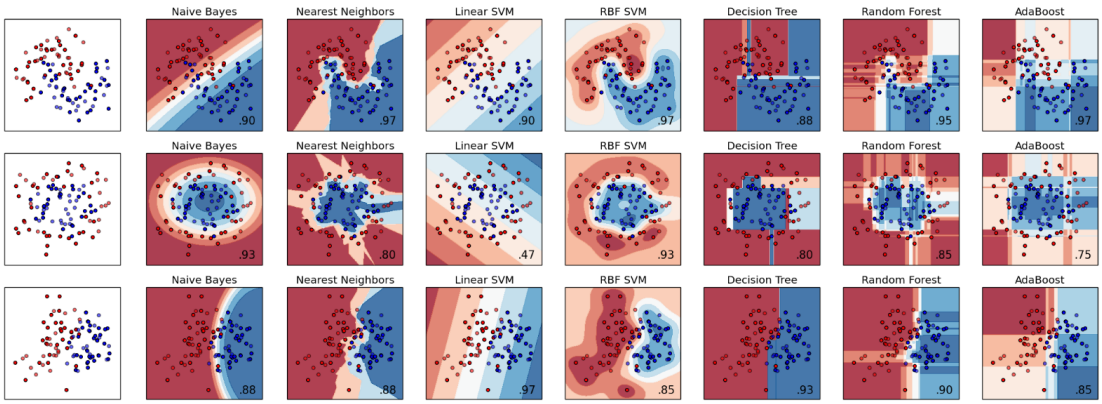

Рис. 5: Наиболее распространенные методы машинного обучения и примеры построенных ими разделяющих плоскостей для различных наборов двумерных данных

Метод K-ближайших соседей Метод K-ближайших соседей - это один из самых простых, но в то же время точных алгоритмов машинного обучения. Он относится к семейству непараметрических алгоритмов, что означает, что алгоритм не делает каких-либо предположений о структуре данных. Это является одним из достоинств этого подхода, поскольку во многих задачах, данные не имеют четко выраженной структуры.

Этот алгоритм не требует обучения и подходит одинаково как для задач классификации, так и для задач регрессии. При решении каждого типа задач, выборка разбивается на целевые классы, а затем, для новых объектов в выборке измеряется значение расстояния до каждого объекта в класте-

рах. Объект получает класс ближайшего к нему кластера. В случае задачи регрессии, в качестве выходного значения алгоритма используется среднее значение расстояний до объектов ближайшего кластера.

В силу того, что алгоритм основан на попарном измерении расстояний, время его работы зависит от размера выборки как $O(n \cdot k + n \cdot d)$. В случае очень больших выборок это приводит в высокой длительности вычислений. Для того, чтобы уменьшить время вычислений используются аппроксимации методов ближайших соседей, основанные на использовании решающих деревьев [19], [20].

Машина опорных векторов Машина опорных векторов (SVM) - это алгоритм машинного обучения, который используется преимущественно для задач классификации. Идея алгоритма состоит в том, чтобы подобрать наиболее оптимальную разделяющую гиперплоскость. Понятие опорных векторов опирается на то, что в методе существуют точки, находящиеся на около границы разделяющей гиперплоскости и меняющие вид плоскости в случае, если их убрать. Расстояние между опорными векторами и гиперплоскостью называется отступом 6.

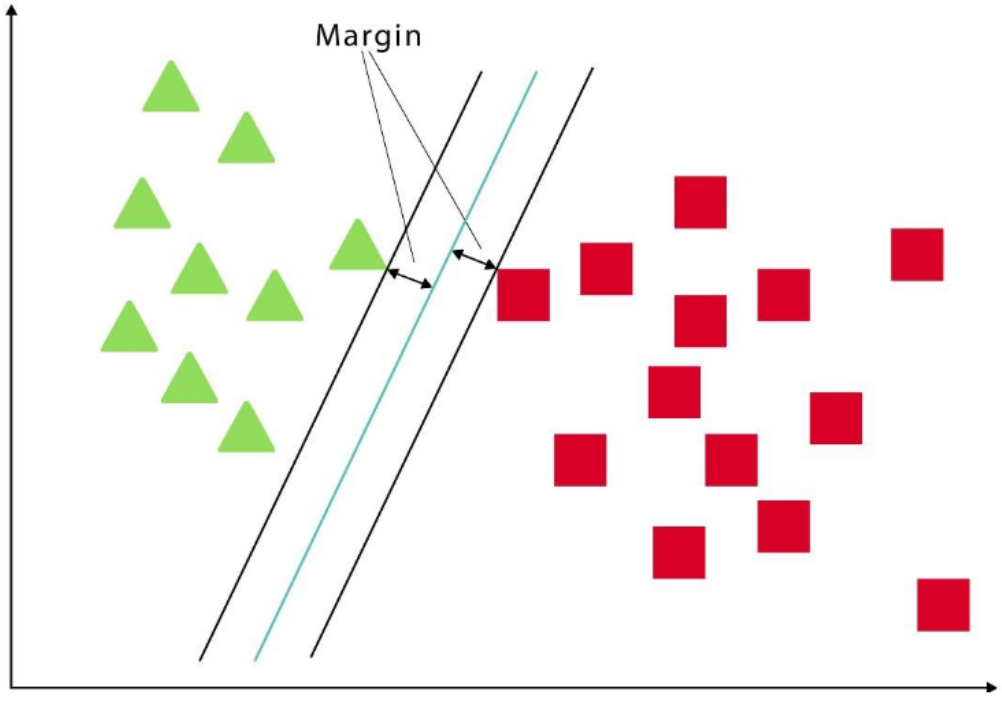

Рис. 6: Пример разделяющей гиперплоскости для алгоритма linearSVM

Шаги обучения алгоритма могут быть описаны следующим образом:

1. Разделить данные на обучающую и тестовую выборки

2. Выбрать функционал эмпирического риска. В случае машины опорных векторов:

$$R_{emp}(a) = \frac{1}{m}\sum_{i=1}^{m}(l(f(x_i, a), y_i))$$

3. Решить задачу оптимизации:

$$\begin{cases} \frac{1}{m}\sum_{i=1}^{m} l(w \cdot x_i + b, y_i) + ||w||^2 \\ min_i|w \cdot x_i| = 1 \end{cases}$$

SVM, как правило, достигает хороших результатов, особенно на "чистых"наборах данных. Более того, алгоритм показывает хорошие результаты на наборах данных, а которых размерность больше числа примеров [21].

Случайный лес Случайный лес - это один из наиболее популярных алгоритмов машинного обучения. Его преимущество - отсутствие необходимости в сложной подготовке данных. Случайный лес базируется на использовании ансамбля решающих деревьев. За счет того, что в результате обучения возможно получить нескоррелированный ансамбль алгоритмов, что позволяет в ряде случаев получать более высокие значения метрики, чем у некоторых линейных алгоритмов. В задаче регрессии ответы деревьев усредняются, в задаче классификации принимается решение голосованием по большинству. Все деревья строятся независимо по следующей схеме [22] (рис. 7):

- Выбирается подвыборка обучающей выборки фиксированного размера – по ней строится дерево (для каждого дерева — своя подвыборка).
- Для построения каждого расщепления в дереве идет рассмотрение определенного числа случайных признаков (max_features) (для каждого нового расщепления — свои случайные признаки).
- Выбираем наилучшие признак и расщепление по нему (по заданному критерию). Дерево строится пока в листьях не останутся представители только одного класса, но в современных реализациях алгоритма есть параметры, которые ограничивают высоту дерева, число объектов в листьях и число объектов в подвыборке, при котором проводится расщепление.

Достоинством алгоритма является малое время обучения, более высокое значение метрик, чем у линейных алгоритмов. Недостатком является плохая интерпретируемость результатов.

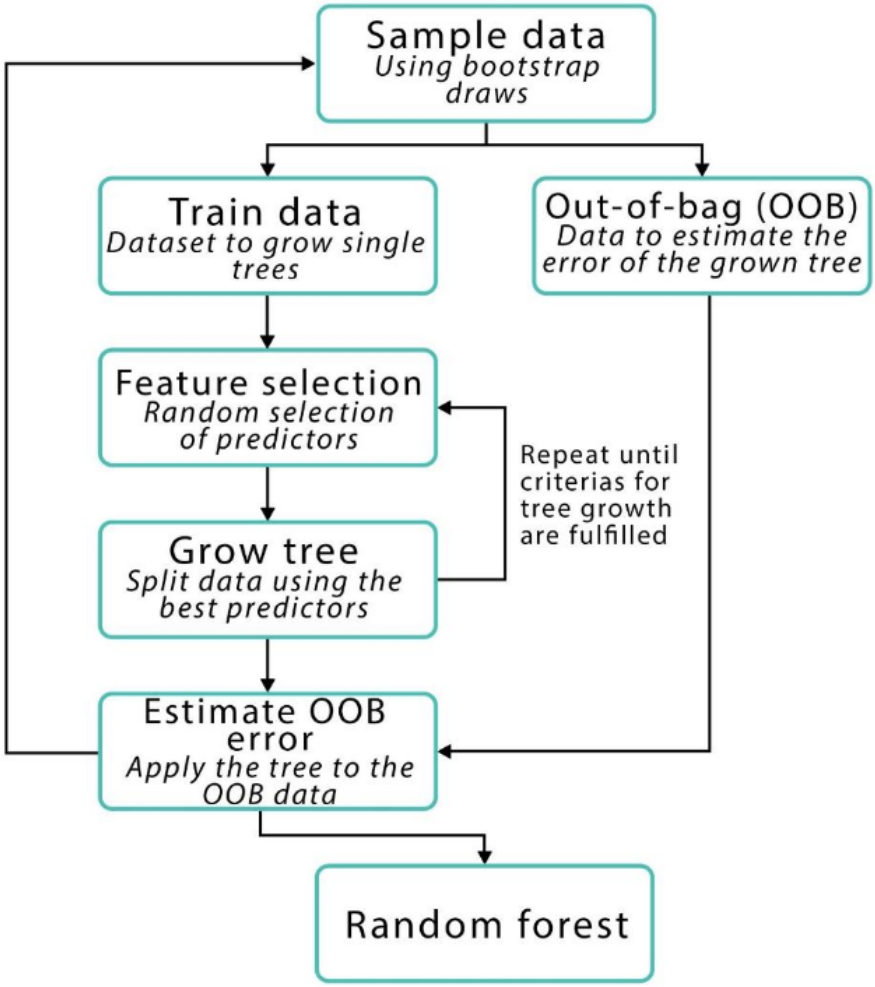

Рис. 7: Схема алгоритма случайного леса

Градиентный бустинг Градиентный бустинг - это алгоритм машинного обучения, представляющий собой линейную комбинации простых классификаторов, каждый из которых обучается таким образом, чтобы минимизировать ошибку предыдущего [23] (рис. 8).

Достоинством алгоритма бустинга является хорошая обобщающая способность. В задачах удаётся строить композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться (в некоторых задачах) по мере увеличения числа базовых алгоритмов. Другим достоинством алгоритма является устойчивость к переобучению.

Недостатком бустинга является то, что жадная стратегия последовательного добавления приводит к построению неоптимального набора базовых алгоритмов. Для улучшения композиции можно периодически возвращаться к ранее построенным алгоритмам и обучать их заново. Для улучшения коэффициентов можно оптимизировать их ещё раз по окончании процесса бустинга с помощью какого-нибудь стандартного метода постро-

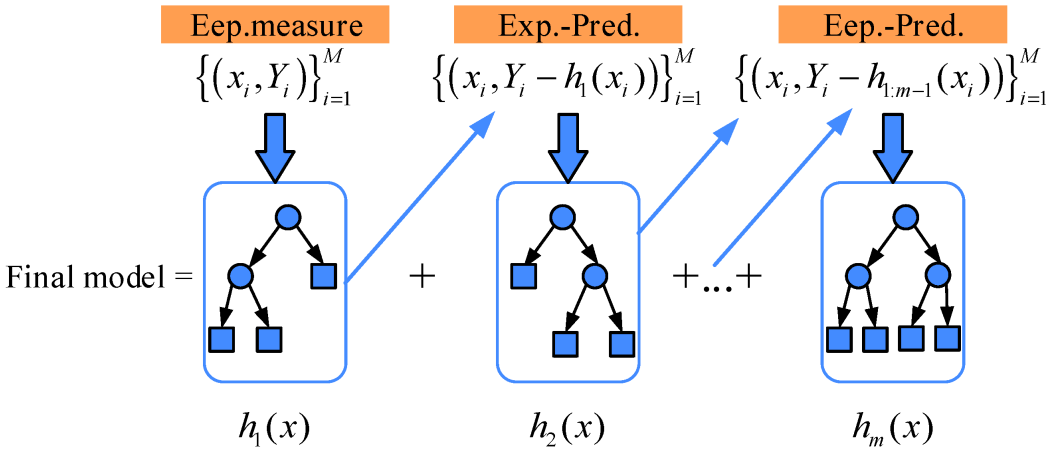

Рис. 8: Схема построения алгоритма градиентного бустинга

ения линейной разделяющей поверхности.

Наиболее известные реализации алгоритма бустинга есть в библиотеках scikit-learn, XGBoost, CatBoost [24, 25, 26].

2.5.2 Алгоритмы поиска аномалий в данных

Среди общепринятых алгоритмов, используемых в задачах классификации и регрессии можно выделить подкласс алгоритмов, задача которых искать аномалии в рассматриваемых данных.

Можно выделить два направления обнаружения аномалий: обнаружение выбросов (Outlier Detection) и поиск новизны (Novelty Detection). И в том и в другом случае речь идет об объектах, которые по своим свойствам отличаются от объектов обучающей выборки (рис. 9).

Выбросы можно разделить на следующие типы:

- Ошибок в данных (неточности измерения, округления, неверной записи и т.п.)
- Наличия шумовых объектов
- Присутствия объектов «других» выборок

Новизна, как правило, появляется в результате принципиально нового поведения объекта. Например, для информационной системы, "новизной"является проникновение вирусного ПО. Признаковое описание нового объекта при этом будет принципиально отличаться от других объектов выборки.

Один из способов поиска новизны в данных - это проведение модельных тестов. Идея модельных тестов заключается в построении модели, которая тем или иным образом описывает данные. Это можно представить как

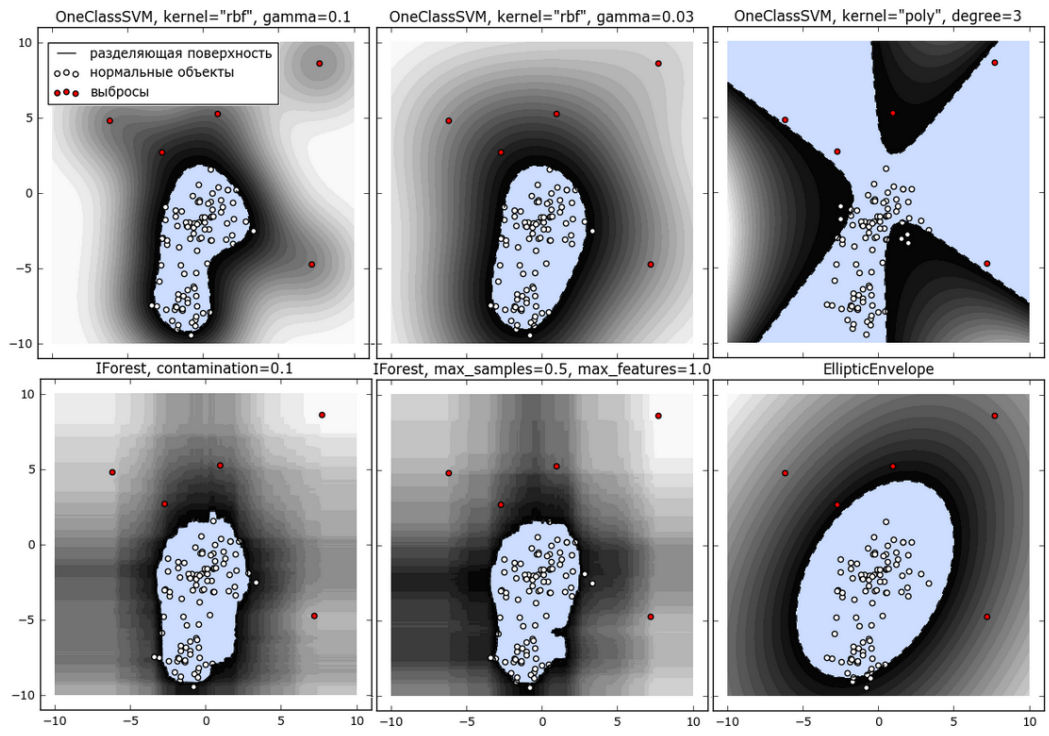

Рис. 9: Визуализация работы разных алгоритмов поиска аномалий.

построение апроксимирующей кривой в n-мерном пространстве точек. Те, точки, которые лежат вне кривой и являются выбросами.

Также, аномалии в данных могут быть выявлены на основе статистических тестов (критерий стьюдента, z-критерий для доли и другие). В рамках теста устанавливается базовая (нулевая гипотеза) о принадлежности данных тому или иному статистическому распределению против альтернативы, которая говорит о том, что данные принадлежат другому распределению. Нюансом этого подхода является интерпретации полученных результатов: гипотезу можно отвергнуть по результатам стат. теста, но не всегда можно подтвердить [27].

Помимо модельных и статистических тестов можно использовать следующие модели:

- Изолирующий лес (Isolation Forest)
- Метод опорных векторов для одного класса (OneClassSVM)
- Эллипсоидальная аппроксимация данных (EllipticEnvelope)

OneClassSVM - это частный случай машины опорных векторов. Отличаем является лишь фиксированное ядро - берется RBF ядро, поскольку для других ядер разделяющие плоскости получается более низкого качества. Из-за сильной заточенности под выборку этот алгоритм можно считать

алгоритмом поиска новизны, нежели алгоритмом поиска выбросов. Тем не менее, эта модель успешно зарекомендовала себя в решении многих задач поиска поломок для производственных датчиков.

IsolationForest - вариация идеи случайного леса. Деревья в алгоритме строятся таким образом, чтобы выбирать случайный признак и случайное расщепление. При этом, мерой однородности данных является среднее арифметическое глубин листьев, в которые попал объект. Считается хорошим алгоритмом для поиска выбросов.

EllipticEnvelope - Идея алгоритма в моделировании точек, как внутренности эллипсоида. Этот метод показывает высокие результаты на одномодальных данных и еще более высокие в случае нормально распределенных данных.

2.5.3 Методы построения композиций базовых алгоритмов

Композицией или ансамблем (Ensemble, Multiple Classifier System) называется алгоритм, который состоит из нескольких алгоритмов машинного обучения. Процесс построения ансамбля называется ансамблированием (ensemble learning) [28].

Техники ансамблирования опираются на идею, что ошибка множества базовых алгоритмов (комитета большинства) может экспоненциально убывать с возрастанием числа базовых алгоритмов. Эта мысль может быть выражена с помощью неравенствам Хёвдинга [29]:

$$\sum_{t=0}^{\frac{n}{2}} C_n^t (1-p)^t p^{n-t} \leq e^{-\frac{1}{2}n(2p-1)^2}$$

Это неравенство верно, если выходы базовых классификаторов/регрессоров t является независимыми случайными величинами. Однако, при решении прикладных задач, зачастую, это предположение не может быть выполнено по следующим причинам:

- Алгоритмы обучаются решать одну задачу
- Используют одни данные для обучения
- Классификаторы могут представлять собой модификации одной базовой модели

Поэтому, методы построения композиций направлены на построение достижение нескоррелированности ансамбля. Тогда ошибки одних алгоритмов могут быть скорректированы работой других.

Достичь этого можно с помощью следующих эвристик: варьирования модели, признаков, выборки.

Рассмотрим наиболее распространенные техники построений композиций алгоритмов:

Bagging В этом подходе каждый базовый алгоритм обучается на случайном подмножестве обучающей выборки. В этом случае, даже используя одну модель алгоритмов, обучаются различные базовые алгоритмы. Случайные подмножества рассматриваются для того, чтобы итоговый ансамбль был нескореллирован. Отбор подпространств может идти как по данным, так и по отдельным признакам.

В качестве примера этого подхода можно рассматривать алгоритм случайного леса, как композиции решающих деревьев. Однако, не все алгоритмы могут быть использованы в качестве базовых для этого метода. SVM и KNN, в частности, не могут рассматриваться по причине того, что они являются алгоритмами, устойчивыми к выборке, а, значит, их композиция будет скореллированной.

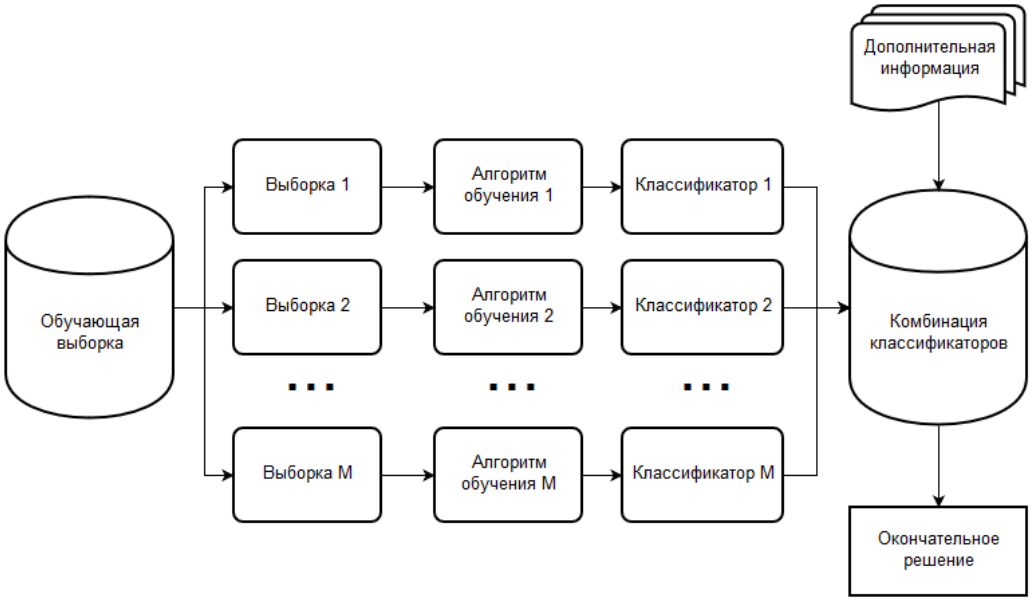

Рис. 10: Bagging: Схема построения

Stacking Выборку разбивают на части (фолды). Затем, последовательно перебирая фолды обучают базовые алгоритмы на всех частях, кроме одной, а на оставшейся получают ответы базовых алгоритмов и трактуют их как значения соответствующих признаков на этом фолде. Для получения метапризнаков объектов тестовой выборки базовые алгоритмы обучают на всей обучающей выборке и берут их ответы на тестовой [**guryaaev**] (рис. 11).

Недостатком этого подхода является то, что мета-признаки могут отличаться для обучающей и тестовых выборок. Более того, дисперсия значений мета-признаком в обучении может быть много больше дисперсии значений для теста и наоборот. Для предотвращения возникновения такой ситуации общепринятой эвристикой является использование регуляризации значений. Например, пороговое отсечение значений.

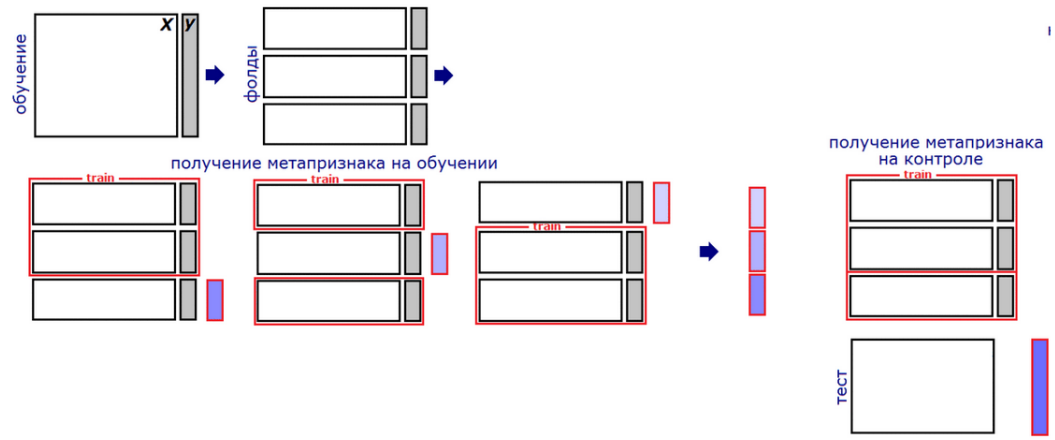

Рис. 11: Stacking: Схема построения

Blending Обучающую выборку делят на две части. На первой обучают базовые алгоритмы. Затем получают их ответы на второй части и на тестовой выборке. Ответ базового алгоритма на частях, на которых не происходило обучение можно рассматривать как новый признак (мета-признак). На мета-признаках второй части обучения настраивают итоговый алгоритм (т.н мета-алгоритм). После этого, алгоритм запускают на мета-признаках теста и получают итоговый ответ (рис. 12).

Недостатком этого метода построения композиции является необходимость деления выборки и, как следствие, уменьшения объема данных при обучении. Поэтому, для получения более высоких значений метрик качества хорошей эвристикой является построение нескольких блендингов и их усреднение.

2.5.4 Автоматическое машинное обучение

Построение моделей, как правило, может быть сведено к задаче перебора параметров, таких как регуляризационные слагаемые, число деревьев и их высота (для градиентного бустинга и случайного леса) и многие другие. Подбор параметров является одним из наиболее трудозатратных этапов при подготовке целевого алгоритма, поэтому, с целью ускорения перебора и его оптимизации были созданы фреймворки для итерации параметров на заданной сетке [30].

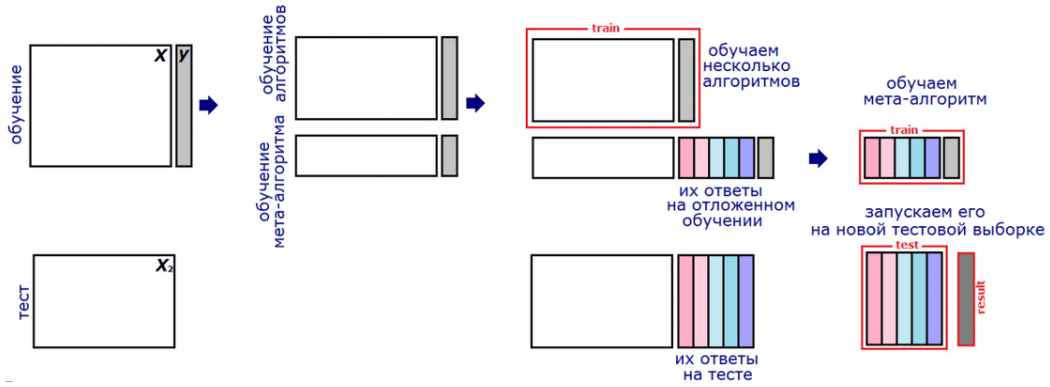

Рис. 12: Blending: Схема построения

Однако, идея итерации параметров пошла дальше и привела к появлению новых библиотек и фреймворков для автоматического подбора моделей и их параметров на основе входных данных [31]. Их использование позволяет свести задачу обучения модели к задаче предобработки входного набора данных (обучающей и валидационной выборок).

В ходе выполнения работы был рассмотрен фреймворк H2O благодаря которому был обучен ряд моделей для обнаружения типа угроз по признакам, извлеченным из сетевого трафика.

2.6 Глубокие нейронные сети

Нейронная сеть (ИНС, искусственная нейронная сеть) - это математический алгоритм, работа которого построена по принципам, по которым работают биологические нейроны.

Сеть состоит из слоев, функциональной единицей которых является формальный нейрон. Формальный нейрон - это простой элемент, у которого есть ограниченное количество входов, к каждому из этих входов привязан некоторый весовой коэффициент. Задача нейрона - осуществление взвешенного суммирования значений от предыдущих нейронов (рис. 13).

Можно выделить несколько общих типов слоев в нейронной сети:

- Input Layer - Начальный слой, который служит для того, чтобы пропустить через сеть очередную порцию данных. У него нет параметров, количество нейронов в нём фиксированно и не является гиперпараметром.

- Hidden layers - Это несколько различных слоёв, размер которых фиксирован архитектурой, но значения параметров меняются в процессе обучения.

- Output layer, количество нейронов в котором фиксированно, не содержит параметров. Обычно используется для подсчете функции ошибки.

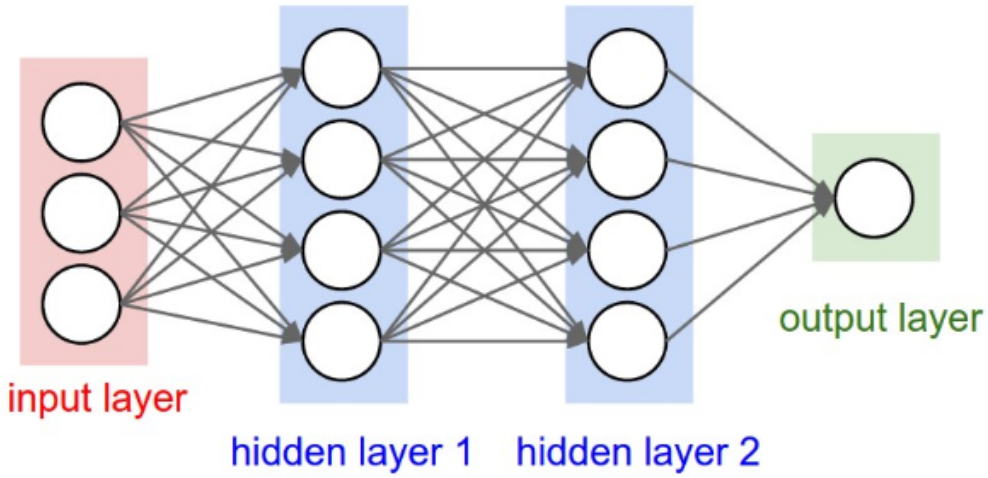

Рис. 13: Схема простой нейронной сети

Работа нейронной сети, построенной по такой схеме равносильна аффинному преобразованию данных в многомерном пространстве. Во многих случаях, в сеть вводятся специальные слои, которые осуществляют нелинейные преобразования ($tanh$, $sigmoid$ и др.). Эти слои называются слоями активации, а преобразование - функцией активации.

Важным моментом является задание функции потерь (Loss Function), которая представляет из себя ошибку, которая минимизируется в ходе обучения сети. Примером такой функции может быть среднеквадратичная ошибка. Однако, в ряде приложений могут использоваться функции, имеющие более сложную структуру либо комбинации из простых функций.

Ошибка на контрольной выборке при обучении вычисляется с помощью метода обратного распространения ошибки [32]. Расчет весовых коэффициентов на каждой итерации определяется методом оптимизации, например стохастическим градиентным спуском.

Среди сетей можно выделить несколько классов:

- Рекуррентные нейронные сети (РНС)

  Основное назначение: обработка текстовой информации, последовательностей, временных рядов.

- Сверточные нейронные сети (СНС)

  Основное назначение: Обработка изображений. Обнаружение, сегментация, классификация изображений.

- Искусственные генеративные модели

  Основное назначение: синтез изображений, перенос информации с изображения на изображения, генерация последовательностей (биоинформатика)

- Полносвязные нейронные сети

  Основное назначение: Обработка табличных данных, альтернатива стандартным классификаторам.

2.7 Примеры использования машинного обучения для выявления вредоносного ПО

Можно выделить несколько общих подходов в несингнатурных методах поиска зараженного трафика:

1. Использование методов машинного обучения
2. Использование нейронных сетей
3. Классические подходы

Рассмотрим использование нейронных сетей в задаче поведенческого поиска зараженного трафика. Поскольку фреймы веб-трафика аккумулируют текстовую информацию, то это открывает возможности для использования реккурентных нейронных сетей.

Например, для выявления действий вредоносного ПО в IoT-сетях (сети, использующие сенсоры, датчики и другие устройства, помимо серверов и ПК) в работе [33] рассматривается использование двунаправленной рекуррентной нейронной сети. Тестирование сети производилось на данных, полученных с зараженной botnet'ом Mirai машины. Mirai - это вредоносное ПО, цель которого заражение IoT устройств. Полученные результаты получились выше результата стандартной LSTM-сети. Для 4 тестовых наборов данных суммарная точность оказалась выше 95

В ряде случаев возможно также использование и сверточных нейронных сетей. Можно выделить два возможных применения: использование сверточных нейронных сетей для обработки временных рядов и использование CHC для обработки изображений, которые могут быть, например, производными рекуррентных нейронных сетей. Интуиция второго подхода - использование CHC для сжатия информации, представления ее в более удобном для обработки виде.

В работе [34] рассматривается комбинированный подход: PCAP данные, собранные с сервера, зараженного вирусным ПО. Данные были пропущены через рекуррентную нейронную сеть, которая извлекала признаковое описание. Признаки представляли из себя тензоры и могли быть интерпретированы как своеобразные изображения. Для обработки тензоров использовалась CHC-классификатор, который в свою очередь определял тип трафика.

Работа [35] предлагает нейросетевой фреймворк для поиска вирусного ПО в бинарных файлах. Для извлечения признаков используется нейронная сеть типа Stacked AutoEncoder (SAE). После извлечения признаки подаются в классификатор, который представляет из себя полносвязную сеть. Авторы статьи указывают, что точность алгоритма на тестовой выборке составляет 97%.

Схожий подход используется в фреймворке BoTShark [36]. Отличие заключается в использовании SAE для поиска BotNet-атак. Автоэнкодеры принимают данные, полученные парсингом PCAP файла, а именно: данные о портах, по которым передаются данные, данные о размере переданных пакетов и частота их передачи. В итоге, получается вектор признаков, который обобщает NetFlow-данные. Этот вектор подается в softmax классификатор, который присваивает каждому объекту выборки метку класса и уверенность алгоритма в прогнозе.

Рассматривая подходы на основе машинного обучения можно выделить использование преимущественно нелинейных алгоритмов: Машины опорных векторов, Случайного леса, Бустинга над решающими деревьями. При этом, ключевым является вопрос отбора признаков. Отмечается, что обнаружение вредоносного трафика сопряжено с ограничениями такими как сложность обобщения методов для всех ботнетов и необходимость рассмотрения частных случаев, например http или IRC ботнетов. Также отмечается вычислительная сложность и временные затраты на анализ трафика при использовании методов машинного обучения как составляющих IPS и IDS.

При отборе признаков рассматривают статистические данных из Flow пакетов. К таким данным относят: порт, адреса компьютеров в сети, средний объем переданных пакетов, количество переданных пакетов за единицу времени и тд. В ряде работ используются алгоритмы машины опорных векторов и решающих деревьев в качестве классификаторов трафика [37, 38, 39].

Стоит отметить наличие патентов и разработок в области защиты информации и обнаружении ботнет атак на основе применения машинного обучения. В патентах [40, 41, 42] описываются системы, извлекающие признаки из веб-трафика и использующие алгоритмы классификации для обнаружения атак.

В некоторых работах рассматриваются подходы обучения с подкреплением (reinforcement learning). Для таких алгоритмов вводится понятие функции награды. Функция награды поощряет алгоритм за правильные шаги и штрафует в случае ошибки. Таким образом, алгоритм подстраивается в процессе анализа данных. Обычно, эти методы используются в игровых приложениях, например, для разработки искусственного интеллекта в AlphaGo, шахматах и тд. Однако в работе [43] рассматривается использование этого подхода для обнаружения вредоносного трафика.

3 Основная часть

В рамках выполнения задачи, поставленной в работе, были обучены алгоритмы машинного обучения для поиска как всех типов вредоносных угроз, так и алгоритмы поиска botnet-атак.

3.1 Данные

3.1.1 Публичные наборы данных

В ходе работы были использованы 4 набора данных, размещенных в сети интернет проектом stratosphere lab [44].

Рассмотрим данные и вредоносное ПО, поведение которого рассматривалось специалистами в ходе их сбора.

- CTU-Mixed-Capture-1

  Этот набор данных содержит вредоносное ПО Bubble Dock .По данным VirusTotal ПО представляет собой бинарный .exe файл, после запуска которого в веб-клиентах появляются рекламные сообщения, баннеры и спонсорские ссылки. Данные содержат смешанный веб-траффик в формате ,pcap, таким образом в данных содержатся как пакеты, соответствующие нормальному поведению пользователя в сети, так и вирусные пакеты.

  К данным прилагается fast.log файл от СОВ Suricata, а также выходные данные СОВ bro. Однако, в силу обновления баз сигнатур, в ходе работы, данные были повторно проведены через СОВ suricata (базы актуальны на момент: 22.03.2020).

- CTU-Mixed-Capture-3

  В наборе содержится веб-трафик, содержащий пакеты, которыми обменивается с удаленным сервером вредоносное ПО a0840a39ec90e1f603e2f4be42a8
  ПО типу поведения - это также ПО размещающее рекламу в веб-клиентах. Характерно то, что среди большинства антивирусного только одна программа находит сигнатуру вируса (SentinelOne) [45].

  Так же как и для предыдущего набора данных для выявления пакетов, которыми обменивается вредоносное ПО была использована СОВ suricata последней версии.

- CTU-Mixed-Capture-2

В данных содержится вредоносное ПО 37e7f6598126096eaa9beea19377f936f9475(ПО представляет собой бинарный .exe файл, который при запуске заражает машину пользователя и блокирует доступ к сети, в том числе и после перезагрузки.

- CTU-Mixed-Capture-9-2

В данных содержится вредоносное ПО Emolet. Это сетевой червь, который является загрузчиком троянских программ на компьютер пользователя. Основное назначение ПО - извлечение информации о финансовых операциях пользователя, например, через онлайн-банкинг. Распространяется через почтовую рассылку в качестве pfd-файлов, word-файлов, подозрительных ссылок на веб-ресурсы.

- CTU-Mixed-Capture-6

В данных содержится ПО DarkVNC. Схема влияния ПО на другие программы на компьютере приведена на рис 14.

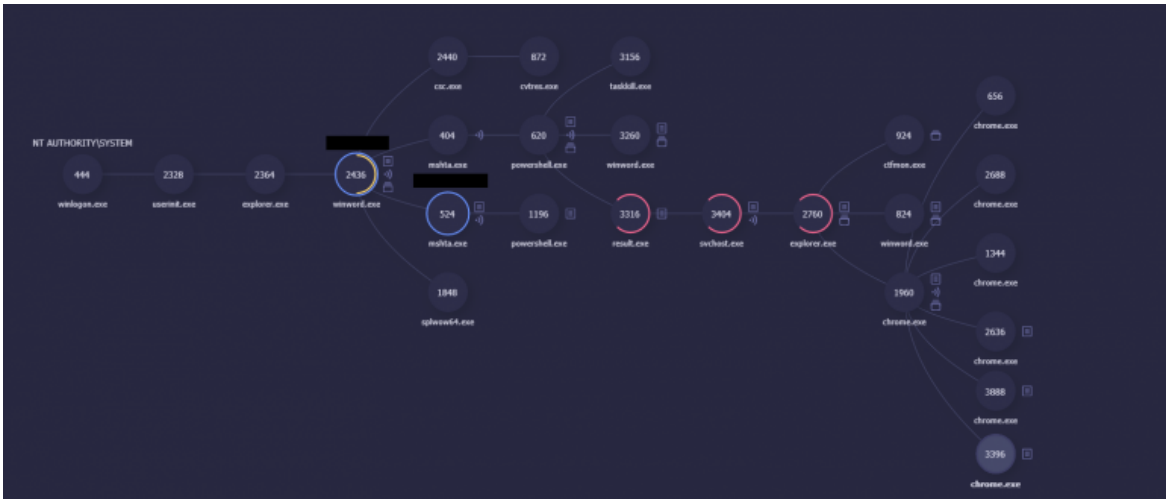

Рис. 14: Схема распространения вредоносного ПО DarkVNC

Сбор таких данных устроен следующим образом: на сервере, на котором стоит чистый образ ОС, без каких либо установленных программ, не считая тех, что извлекают трафик и IDS/IPS в течении некоторого времени наблюдают за поведением вирусной программы. Как правило эксперимент разбивают на период, когда Вирусное ПО не активно и на период, когда экспериментатор активирует его. При этом, экспериментатор журналирует свою деятельность (какие сайты он посещал и какие действия выполнял на них).

Системный параметр	Значение
Версия ОС	Windows 10
Процессор	Intel Core i5 7300HQ
ОЗУ	DDR4, 8192 Мб
Жесткий диск	SSD 256 Гб
Версия VirualBox	6.1

Таблица 2: Параметры хост-машины для сбора реальных данных

После наблюдения за вирусом идет сохранение записанного сетевого трафика (например средствами программы wireshark). Трафик в последствии пропускается через СОВ, которые маркируют выявленную угрозу. Это позволяет описать поведение вируса.

3.1.2 Реальные данные

Помимо анализа и обучения на публичных наборах данных, была собрана выборка реальных данных захвата сетевого трафика от вредоносного ПО.

Для этого на хост-машине было установлено ПО Oracle VirtualBox для создания изолированных виртуальных машин на базе образа ОС. Параметры хост машины приведены в табл 3.1.2.

В среде VirtualBox была создана виртуальная машина на базе ОС Windows 7 Basic. Кроме ОС в виртуальной машине не было установлено какого-либо дополнительного ПО. Параметры виртуальной машины приведены в табл 3.1.2.

В качестве типа сетевого подключения было выбрано NAT-подключение. Оно позволяет виртуальной машине иметь доступ к сети интернет, не оказывая влияния на хост-систему.

Запись трафика осуществлялась встроенными методами VirtualBox в pcap файл, который в последствии был предобработан и оформлен в виде набора данных для тестирования.

На виртуальную машину был установлен веб-клиент google chrome 81.0. После этого, с сайте Strtospherelab был скачан exe-файл из набора данных CTU-Malware-Capture-Botnet-349-1. Файл содержит экземпляр вредоносного ПО AdLoad. Это тип вредоносного ПО, которое приносит доход своему разработчику за счет показа рекламы, например, в веб-клиентах. Особенностью этой программы является то, что она подменяет веб-клиент Internet Explorer на его модифицированную версию, которая вставляет HTTP-верстку в файлы сайтов для отображения рекламы.

Была произведена установка exe файла с AdLoad после чего в течении 2 часов осуществлялась активность в веб-клиенте. Это был просмотр различных сайтов для накопления обычного трафика. После сбора данных, виртуальная машина была выключена.

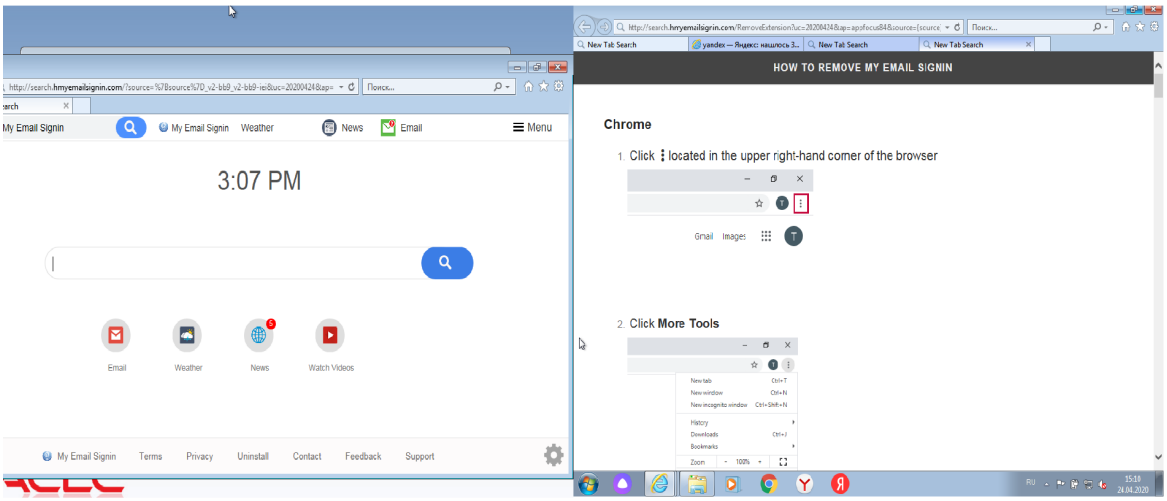

Рис. 15: Скриншот экрана зараженной вирусом AdLoad системы. При заражении вирус предлагает изменить основную страницу веб-клиента на страницу, содержащую рекламу. Слева приведено основное окно браузера после заражения, справа окно, имитирующее деинсталяцию вирусной программы с компьютера

Параметр виртуальной машины	Значение
Версия ОС	Windows 7
Объем виртульной памяти	3072 Мб
Тип виртуального сетевого адаптера	NAT
Объем образа жесткого диска	32 Гб

Таблица 3: Параметры виртуального машины для установки вредоносного ПО и захвата сетевого трафика

3.2 Извлечение признаков

С помощью библиотеки pyshark, которая делает системные вызовы программы wireshark данные были разобраны на набор фреймов (информации о каждом слое в модели OSI), которые, в свою очередь, были приведены к табличном виду (формату csv) средствами языка python.

Классификация строилась на признаках из различных слоев трафика, рассмотрим специфические признаки.

3.2.1 Предобработка

3.2.2 Признаки, извлеченные из TCP трафика

Из TCP трафика были извлечены следующие признаки: Порт исходящего/приходящих пакетов, суммарная длина пакета, частота передачи сетевых пакетов и другие. Вторая группа признаков включает в себя признаки

созданные вручную - это различные статистики: средние значения признаков, медианы и квантили.

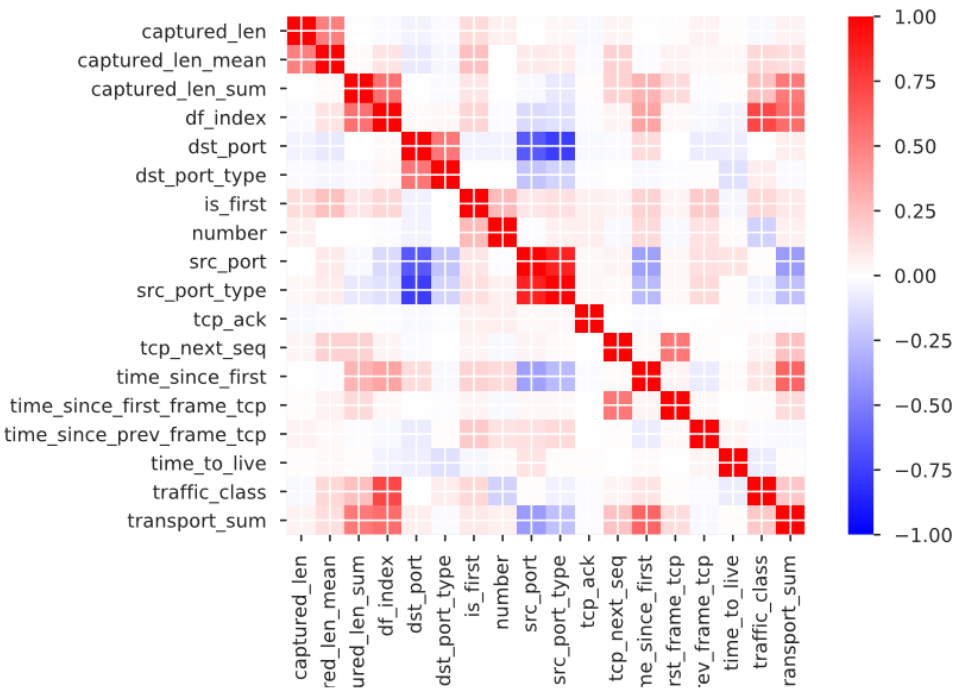

Рис. 16: Пирсоновская корреляция между признаками. Характеризует степень линейной зависимости числовых признаков. Красный и синий цвет обозначает наличие линейной связи между признаками либо производными этих признаков, но указывает на направление этой связи: увеличение или уменьшение значения признака при изменении другого. Белый цвет - отсутствие связи.

Диаграмма кореляции признаков показывает, что наибольшую корелляцию с целевым признаком (traffic_class) имеет средняя длина пакета в соединении и среднее время, которое проходит между посылкой первого пакета.

Признаки также были проанализированы на предмет их влияния на конечную классификацию. Для этого был взят алгоритм Random Forest и построен график важности признаков с точки зрения конечной классификации. Значение важности признака основано на суммарном вкладе, который вносит этот признак в уменьшение ошибки на обучении. Эта информация косвенно указывает на то, какие признаки могут быть связаны с целевой переменной, но не являются точным доказательством наличия этой связи.

Помимо этого, были рассмотрены алгоритмы поиска аномалий в данных. В частности, был рассмотрен алгоритм Isolaton Forest и OneClassSVM, которые были описаны в обзоре.

Результаты алгоритмов приведены на рис. 18

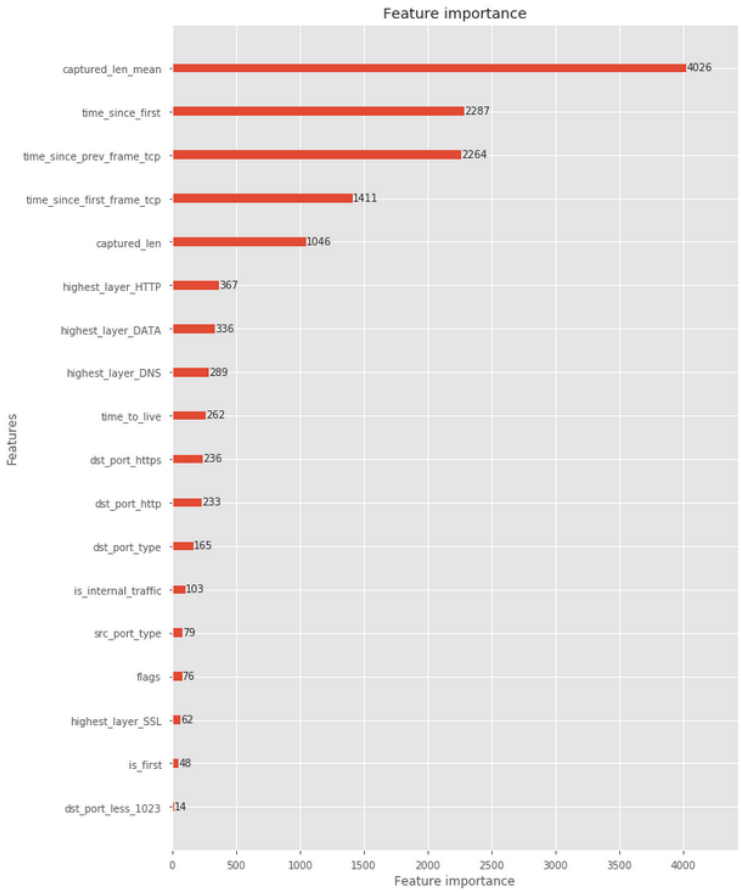

Рис. 17: Диаграмма важности признаков для классификационного алгоритма RandomForest. Так же как и в случае с диаграммой корреляции заметна связь признаков связанных с длиной пакета, временем отправки и целевой переменной

3.2.3 Признаки протоколов прикладного уровня

Помимо признаков из TCP/IP слоев веб-фреймов были рассмотрены признаки протоколов прикладного уровня. В частности были рассмотрены HTTP и SSL слои. При отсутствии этих слоев, значение соответствующего этим слоям признакам приравнивается нулю.

Для HTTP были рассмотрены следующие признаки: наличие cookie, длина cookie, веб-хост, уровень домена, User-Agent.

Для SSL: версия SSL-записи, длина сессии, SSL cipherspec.

Причина рассмотрения таких признаков - признаков, полученных от протоколов более низких уровней недостаточно для классификации. Зачастую, характерные атрибуты вирусного ПО могут быть найдены в текстах запросов, наличии характерных ключей в base64 (строковом) представлении изображений [46]. Часто, в запросе происходит подделка поля User-agent [47]. Например, характерной чертой трояна Gameover Zeus является длина домена от 23 до 28 символов.

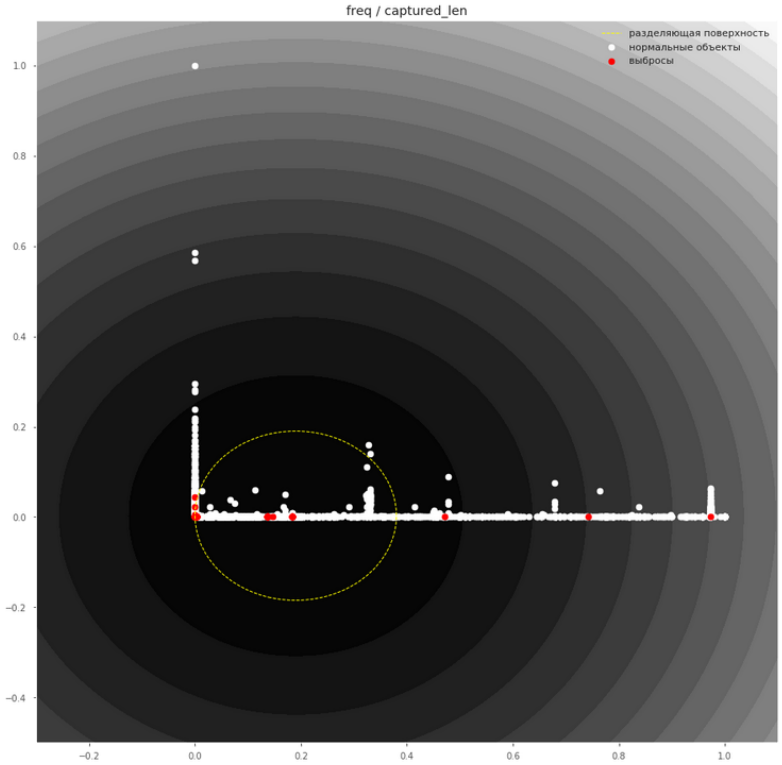

Рис. 18: Результат применения алгоритма OneClassSVM для пары признаков частота передачи пакетов/длина пакета. Объекты, представляющие угрозу выделены красным цветом, обычные объекты (нормальный трафик) - белый цвет. Разделяющая поверхность отделяет часть трафика от остального. Можно наблюдать, попадание пакетов, выделенных Suricata внутрь, что косвенно говорит о связи типа трафика с объемными и временными характеристиками.

При постановке экспериментов с обучением моделей были рассмотрены и данные с признаками прикладного уровня и признаки без них. Стоит отметить, что использование HTTP/SSL признаков при валидации давало до 15% увеличения метрики качества.

3.3 Визуализация данных и кластеризация

Отдельно, перед обучением моделей, предобработанные данные для каждого набора были визуализированы с использованием алгоритмов понижения размерности таких как PCA и TSNE.

PCA (Principal Component Analysis, Метод главных компонент) - алгоритм, позволяющий уменьшить размерность данных, потеряв наименьшее количество информации. Применяется метод во многих областях: распознавание образов, компьютерное зрение, интеллектуальный анализ данных. Метод главных компонент заключается в том, что строится мини-

мальное число новых признаков, по которым исходные признаки восстанавливаются с помощью линейного преобразования с минимальными погрешностями [48]. Эти признаки строятся в базисе так называемых главных компонент - это ортогональный базис в котором один из базисных векторов характеризует направление максимальной дисперсии данных, а остальные базисные вектора ортогональны ему [49].

TSNE (t-distributed stochastic neighbor embedding)- это алгоритм машинного обучения для визуализации данных основанный на методе нелинейного уменьшения размерности. Основная идея TSNE - уменьшить размерное пространство, сохраняя относительное попарное расстояние между точками [50].

Алгоритм работы таков TSNE таков: сначала вычисляется матрицу попарного расстояния P от исходных точек. Затем алгоритм оценивает матрицу попарных расстояний Q в пространстве меньшой размерности так,чтобы расстояние между P и Q было минимизировано [51].

Основным методом этого алгоритма является использование распределения Стьюдента для вычисления расстояния Q .Имея более длинный хвост,чем гауссов, распределение Стьюдента не отделяет разнородные точки данных, расположенные слишком далеко друг от друга, в то время как сохраняет похожие точки данных достаточно близко.

Применение этих алгоритмов позволяет оценить насколько возможно выделить какие либо кластеры в данных. В случае наличия таковых возможно использование алгоритмов K-means, DBScan или K-nearest neighbours для классификации новых данных по выделенным кластерам.

Рассмотрим визуализацию наборов данных CTU-Mixed-Capture-1 и CTU-Mixed-Capture-3. В первом случае сессия была достаточно длинная (несколько дней), поэтому получившиеся облака точек получаются довольно плотными. Во втором, данных относительно меньше и они имеют более разрозненную структуру.

Для PCA-анализа характерно аккумуляция точек около нижних осей. Это обусловленно тем, что ряд признаков имеют большую дисперсию и при этом значения распределены непропорционально - в данных есть высокие значения отдельного признака, но также есть и околонулевые значения, которых значительно больше.

TSNE-анализ дает более интересную структуру облака точек. Формируются отдельные кривые из точек имеющие схожие значения признаков. 3D визуализация приведена на рис.

Выбор данных для визуализации обусловлен тем что CTU-1 это данные в которых довольно много пакетов переданных вирусным ПО, в то время как в CTU-3 suricata выявила всего один подозрительный пакет. Более того, дальнейший анализ показывает, что обученные модели не находят единственный подозрительный пакет. Однако, TSNE выделяет отдельный кластер с точкой, соответствующей этому пакету, причем соседние точки

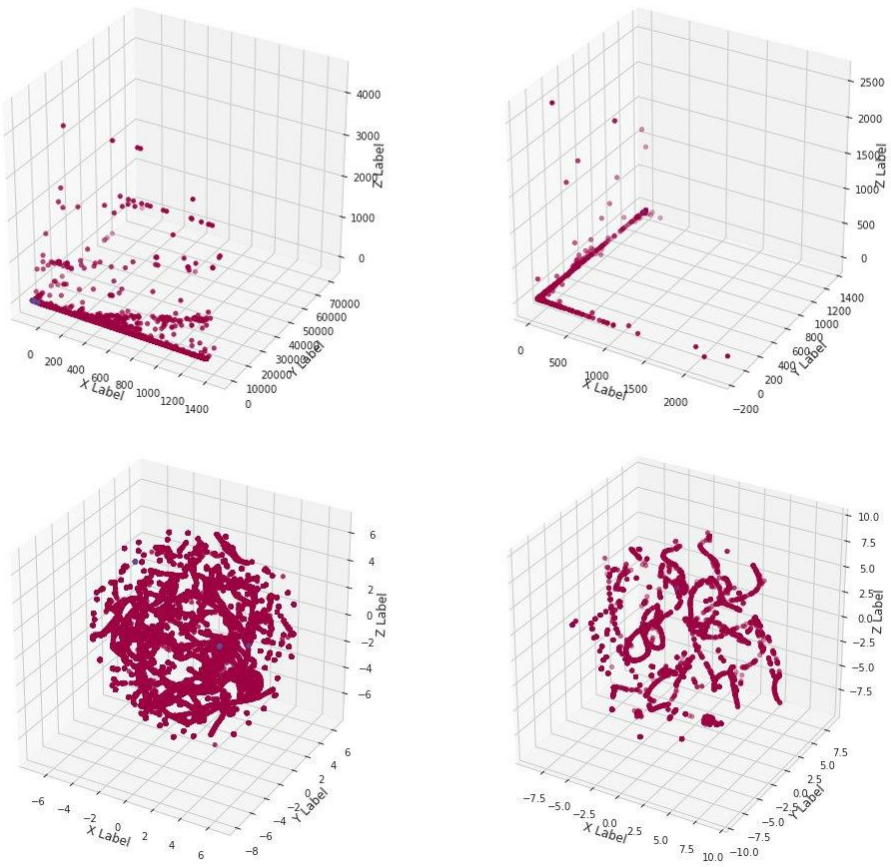

Рис. 19: PCA и TSNE анализ наборов данных CTU-Mixed-Capture-1 и CTU-Mixed-Capture-3 (сверху PCA, снизу TSNE). Левый столбик соответствует данным CTU-Mixed-Capture-1, правый - CTU-Mixed-Capture-3. Красным цветом выделены пакеты с обычным трафиком, голубым/фиолетовым цветом выделены вредоносные пакеты. Видно, что структура облаков точек более плотная для TSNE и образует своеобразный "клубок"из кривых.

в кластере имеют схожие характеристики и тот же ip-адрес отправителя. Таким образом можно выделить определенную комбинацию значений признаков, которые позволят дообучить классификатор определять такие сложные случаи.

Для того, чтобы выделить кластер рассмотрим снижение размерности до двумерной плоскости (рис. 20)

3.4 Обучение моделей

Для обучения был использован набор данных CTU-Mixed-Capture-1. Он был разит на обучающую и валидационную выборку в соотношении 4:1. При разбиении была произведена стратификация данных по классам. Таким образом, в разбиениях была одна пропорция данных целевого класса.

Все признаки, не имеющие логического или числового представления

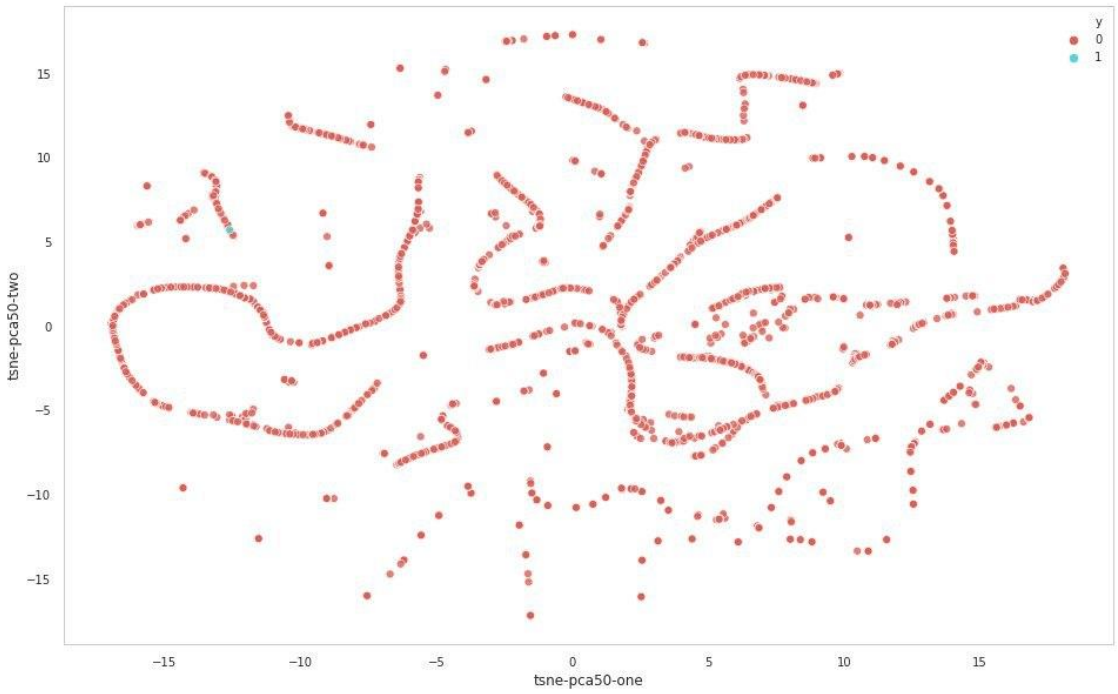

Рис. 20: Двумерный TSNE для набора данных CTU-Mixed-Capture-3. Пакет от вредоносного ПО попадает на линию слева сверху (пакет выделен голубым). Точки, находящиеся на одной кривой с этим пакетом имеют те же ip адреса, что может говорить о том, что их можно так же использовать в обучении.

Модель	Реализация	Параметры
Gradient Boosting	XGBoost	num_leaves = 15, n_estimators = 1000, max_depth = 5
Random Forest	Scikit-learn	n_estimators=4000, max_depth=15, max_features=6
K-nearest neighbours	Annoy	n_trees = 8, metric='euclidean'

были удалены из выборки.

Были обучены следующие модели: Gradient Boosting (XGBoost), Random Forest (Sklearn), K-nearest neighbours.

Поскольку KNN является вычислительно трудоемким алгоритмом, вместо него использовалась аппроксимация алгоритма из библиотеки annoy [52]. В качестве расстояния использовалось евклидово расстояния в 36-мерном пространстве признаков.

Для алгоритмов случайного леса и градиентного бустинга использовались сеточные методы оптимизации параметров из библиотеки hyperopt. Отобранные параметры приведены в табл 3.4.

Помимо сеточных методов перебора параметров также были использованы AutoML-методы. С помощью библиотеки H2O на основе входных

данных был сформирован список моделей. потенциально подходящих для задачи классификации трафика. Однако, во многом, отобранные модели и их параметры совпали с теми, которые были отобраны hyperopt.

Всего было обучено несколько моделей каждого типа. Причиной этому стали различные стратегии отбора признаков. В частности был использована стратегия поочередного добавления и удаления признаков [53].

Стоит отметить, что зачастую, необходимо понимать, каким образом модель принимает решение после обучения. На какие признаки она опирается. Для этого для моделей, использующих деревья можно сделать оценку и визуализировать примерную структуру дерева. Такая визуализация была построена и может служить ориентиром, в каком порядке модели рассматривают признаки после обучения (рис. 21)

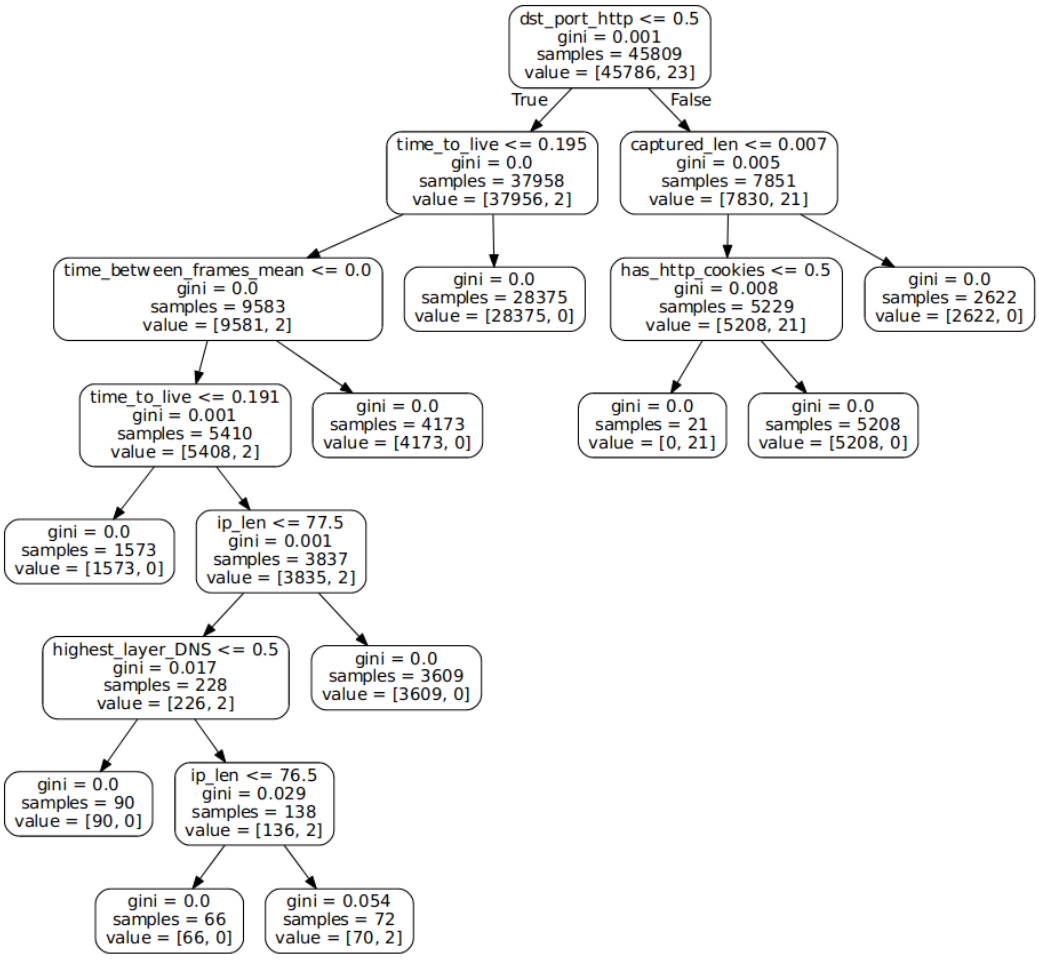

Рис. 21: Схема решающего дерева, созданная на основе классификатора DecisionTreeClassifier. На ее основе можно оценить каким образом обученные модели используют признаки при классификации.

3.5 Результаты

3.5.1 Метрики качества

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

В качестве основной метрики для валидации алгоритма использовалась $F1-$. Она равна гармоническому среднему между точностью и полнотой. характеризует скольким объектам не целевого класса были присвоены ложные метки, а полнота показывает сколько объектов целевого класса были упущены алгоритмом (рис. 23). Формула F-меры:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

Достоинством F-меры является то, что она устойчива к наборам данных, в которых присутствует сильный дисбаланс классов, в отличие от метрики accuracy. В данном случае, число зараженных пакетов сильно меньше количества обычных. Это обосновывает выбор метрики.

Часто результат работы алгоритма на фиксированной тестовой выборке визуализируют с помощью ROC-кривой (ROC = receiver operating characteristic, иногда говорят «кривая ошибок»), а качество оценивают как площадь под этой кривой – AUC (AUC = area under the curve).

AUC ROC – площадь под ROC-кривой – часто используют для оценивания качества упорядочивания алгоритмом объектов двух классов. Значение ROC AUC лежит на отрезке [0, 1] [54].

Можно выразить ROC AUC с помощью следующей формулы:

$$ROCAUC = \frac{\sum_{i=1}^{q}\sum_{j=1}^{q} I[y_i < y_j] I'[a_i < a_j]}{\sum_{i=1}^{q}\sum_{j=1}^{q} I[y_i < y_j]}$$

Где a_i - ответ алгоритма а I-ом объекте, y_i - его метка, q - число объектов в тестовой выборке:

$$I'[a_i < a_j] = \begin{cases} 0, & \text{если } a_i > a_j \\ 0.5, & \text{если } a_i = a_j \\ 1, & \text{если } a_i < a_j \end{cases},$$

$$I'[y_i < y_j] = \begin{cases} 0, & \text{если } y_i > y_j \\ 1, & \text{если } y_i < y_j \end{cases}$$

В случае бинарной классификации предыдущая формула может быть переписана в виде [55]:

$$ROCAUC = \frac{1 + TPR - FPR}{2}$$

Таким образом, AUC ROC равен доле пар объектов вида (объект класса 1, объект класса 0), которые алгоритм верно упорядочил.

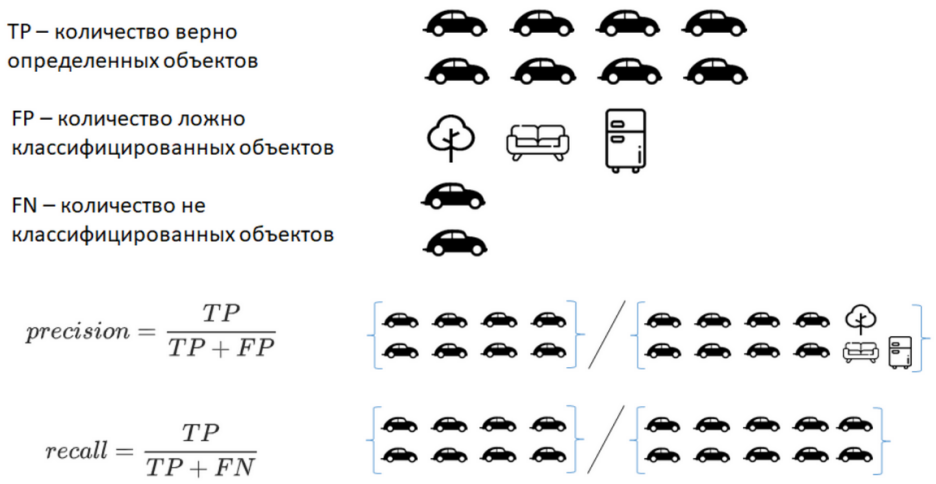

Рис. 22: Схема подсчета полноты и точности

3.5.2 Тестирование

Данные тестирования приведены в табл 3.5.2.

Тестирование проводилось на всех наборах данных. Для набора данных CTU-1 приведены данные, полученные на валидации.

Наилучшие результаты среди всех моделей были продемонстрированы алгоритмом градиентного бустинга. Также высокие результаты были продемонстрированы алгоритмом случайного леса. Однако, для набора данных CTU-3 только алгоритм K-ближайших соседей (с использованием доразметки на основе выделения TSNE-кластеров) смог отобрать пакеты от вирусного ПО. Это обусловлено малым количеством примеров целового класса в данных.

Рассмотрим структуру F1-меры в полученных результатах (значения полноты и точности для каждого алгоритма). Данные приведены в табл 5.

	CTU-1	CTU-2	CTU-3	CTU-6	CTU-9-2	Real Traffic
Gradient Boosting	98%	88%	0%	82%	91%	41%
Random Forest	96%	87%	0%	81%	90%	43%
K-nearest Neightbours	88%	68%	34%	59%	56%	39%

Таблица 4: Значения F1-меры для отобранных алгоритмов. Для большинства наборов данных алгоритмы градиентного бустинга и случайного леса показывают наилучшие результаты. Однако, для набора данных CTU-3 (тип ПО AdWare) в силу малого числа примеров целевого класса (1 пример), алгоритмы не смогли обнаружить уязвимость, что привело к нулевому значению метрики. Использование TSNE для доразметки данных и алгоритма K-ближайших соседей для классификации позволило отобрать несколько зараженных пакетов.

	CTU-1		CTU-2		CTU-3	
	precision	recall	precision	recall	precision	recall
Gradient Boosting	100%	100%	46%	63%	0%	0%
Random Forest	100%	100%	100%	76%	0%	0%
K-nearest Neightbours	81%	96%	88%	55%	21%	100%
	CTU-6		CTU-9-2		Real Traffic	
	precision	recall	precision	recall	precision	recall
Gradient Boosting	74%	92%	100%	90%	28%	100%
Random Forest	73%	92%	100%	90%	27%	100%
K-nearest Neightbours	51%	70%	53%	63%	25%	90%

Таблица 5: Значения точности и полноты для рассмотренных алгоритмов

Падение f1-метрики для ряда данных связанно с выделением нецелевых пакетов (для данных CTU-6). Также, присутствуют случаи падения метрики из-за выделения не всех подозрительных пакетов.

Отдельно стоит рассмотреть работу алгоритмов на реальных данных, полученных с помощью захвата сетевого трафика. Для вредоносного ПО AdLoad были выделены все пакеты, связанные с управляющими командами для внедрения рекламных баннеров (эти пакеты были помечены как целевые). Однако, помимо этого, были выделены пакеты, в которых передавались данные для интеграции в баннеры. Это привело к падению значения F1-меры, поскольку упало значение метрики precision. Тем не менее, значения признаков для выделенных моделями пакетов могут быть склатеризованы в TSNE-кластер, что говорит о наличии схожих паттернов для этих данных.

Помимо, F1-метрики были построенные ROC-кривые для градиентного бустинга и случайного леса. Графики приведены на рис 23.

ROC-кривая характеризует алгоритм, хорошо упорядочивающий данные по классам в случае если график проходит через левый верхний угол

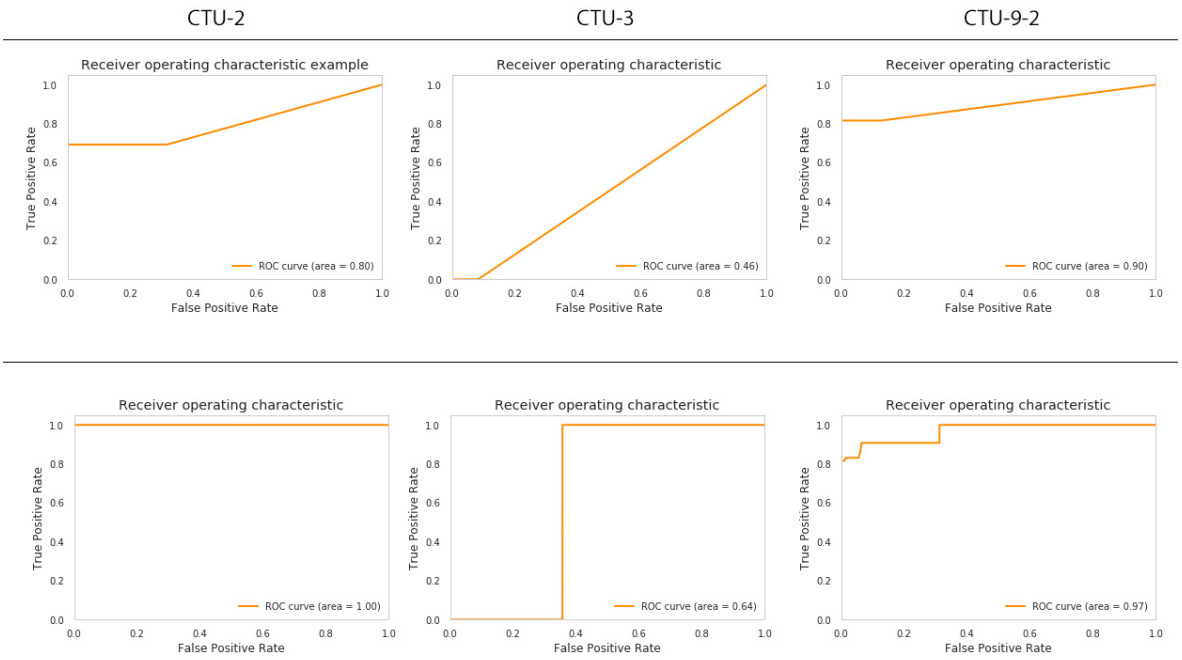

Рис. 23: ROC-кривые для модели случайного леса и градиентного бустинга. Первая, вторая и третья колонки соответствуют наборам данных CTU-2, CTU-3, CTU-9-2. Верхний ряд - модель случайного леса, Нижний - градиентный бустинг.

либо имеется ситуация близкая к этому. В случае, если график представляет собой наклонную линию, алгоритм не случайным образом упорядочивает данные. Это означает, что его поведение в ряде случаев мало отличается от примитивного случайного классификатора. Алгоритм не упорядочивает пары в случае, если большая часть графика лежит на оси x либо близко к ней.

Таким образом, можно заключить. что на общедоступных наборах данных, за исключением случаев, когда объекты целевого класса присутствуют в единичных экземплярах в данных, нелинейные модели. строящие разделяющие гиперплоскости, а именно случайный лес и градиентныый бустинг, показывают значения целевой метрики порядка 90%, Это позволяет использовать их для выявления вредоносного ПО тех типов которые представлены в данных.

В случае малы объемов данных от вредоносного ПО возможно использование комбинации алгоритмов понижения размерности и алгоритмов кластеризации данных. В частности, возможна доразметка данных на основе использования TSNE и классификация новых данных с помощью метода K-ближайших соседей на основе выявленной информации о структуре кластеров. В работе, это позволило найти зараженные пакеты для набора данных CTU-3, в которых остальные алгоритмы показали низкие показа-

тели метрики качества.

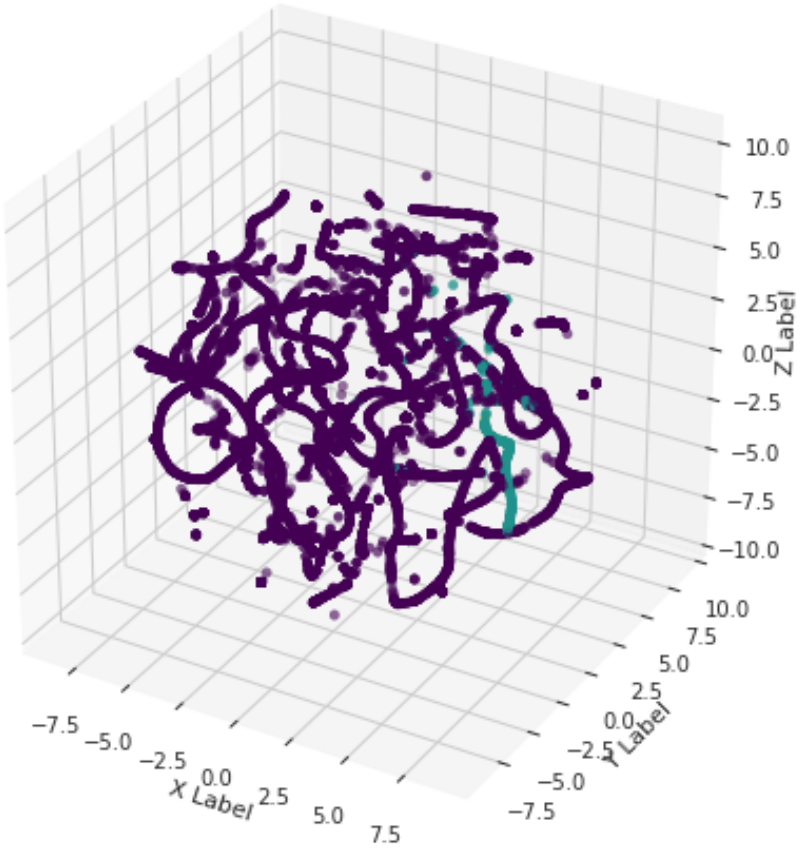

Рис. 24: TSNE кластеризация, построенная на базе реальных данных (Real Traffic). Более светлая область соответствует трафику, выделенному моделью RandomForest. Видно, что эта область образует регулярную кривую в трехмерном пространстве.

4 Выводы

В ходе работы было рассмотрено использование алгоритмов машинного обучения для обнаружения вредоносного ПО на основе анализа веб-трафика. Было рассмотрено несколько моделей, которые были обучены на реальных данных, собранных с зараженных вирусным ПО машин. Каждая из этих моделей была протестирована на отложенных выборках и были оценены метрики качества.

Также, было рассмотрено применение алгоритмов кластеризации для выявления подозрительных пакетов в данных. Визуализация работы этих алгоритмов позволила определить набор признаков, которые позволяют упростить обнаружения вредоносного ПО.

Помимо этого, для тестирования моделей были собраны данные, содержащие реальный сетевой трафик от зараженной системы. Для этого была использована изолированная виртуальная машина на базе ОС Windows 7, которая была заражена вирусом типа AdWare. При этом, в данных присутствовал как нормальный трафик, так и трафик от вируса.

Анализ полученных результатов показал, что, для ряда данных, нелинейные алгоритмы, такие как случайный лес, градиентный бустинг, метод K-ближайших соседей могут выявить пакеты отправленные вредоносным ПО и отличить их от пакетов данных, относящихся к обычному веб-трафику.

В частности, для данных с реальным трафиком, полученные результаты продемонстрировали, что модели смогли выявить все пакеты, которые были размечены как подозрительные. То есть, модели способны выявлять угрозу не только для публичных наборов данных, но и для реальных данных.

Это говорит о том, что алгоритмы машинного обучения являются очень многообещающим подходом для выявления угроз. И текущие исследования показывают, что появляется все больше систем обнаружения вторжений и систем обнаружения и предотвращения вторжений, основанных на этих алгоритмах, не использующих, ставшие уже традиционными, сигнатурные методы. В ряде стран защищены патенты на использование технологий машинного обучения в антивирусном ПО.

Стоит отметить, что рассмотренные методы обладают необходимым быстродействием и могут быть легко масштабированы для анализа большого количества трафика. Это также позволяет рассматривать их как альтернативу устоявшимся алгоритмам.

Таким образом, можно говорить о появлении антивирусного ПО, которое будет гибко адаптироваться к появлению новых угроз, что в перспективе позволит предотвратить такие явления как атаки на сервера, утечку данных пользователей и финансовые потери, обусловленные действиями вирусов.