**Аннотация**

В данной выпускной квалификационной работе была рассмотрена задача анализа сетевого трафика при помощи машинного обучения, в ходе которой был проведен анализ актуальности проблемы, созданы функциональные модели проектной процедуры IDEF0 и IDEF3, были сформулированы цели и задачи работы. Также в работе изучены методы анализа сетевого трафика и алгоритмы машинного обучения, рассмотрены алгоритмы машинного обучения для анализа сетевого трафика. Была описана содержательная и математическая постановка задачи. В рамках разработки информационного обеспечения была создана концептуальная, логическая модель базы данных, был проведен анализ функциональных требований и разработан интерфейс программы. В завершении работы была разработана программа для анализа сетевого трафика при помощи машинного обучения.

**Annotation**

In this final qualifying work, the problem of analyzing network traffic using machine learning was considered, during which the relevance of the problem was analyzed, functional models of the IDEF0 and IDEF3 design procedures were created, and the goals and objectives of the work were formulated. Also, the paper studies methods for analyzing network traffic and machine learning algorithms, considers machine learning algorithms for analyzing network traffic. The substantive and mathematical formulation of the problem was described. As part of the development of information support, a conceptual, logical model of the database was created, an analysis of functional requirements was carried out and a program interface was developed. At the end of the work, a program was developed for analyzing network traffic using machine learning.

СОДЕРЖАНИЕ

[ГЛАВА 1. АНАЛИЗ ПРОЕКТНОЙ ПРОЦЕДУРЫ АНАЛИЗА СЕТЕВОГО ТРАФИКА ПРИ ПОМОЩИ МАШИННОГО ОБУЧЕНИЯ 13](#_Toc71124330)

[1.1 Современные тенденции и проблемы выполнения процедуры анализа сетевого трафика 13](#_Toc71124331)

[1.2. Разработка функциональных и поведенческих моделей (IDEF0 и IDEF3) процедуры анализа сетевого трафика 14](#_Toc71124332)

[1.3 Обзор систем автоматизации процедуры анализа сетевого трафика 22](#_Toc71124333)

[1.4 Цель и задачи автоматизации процедуры анализа сетевого трафика 23](#_Toc71124334)

[Выводы по главе 1 24](#_Toc71124335)

[ГЛАВА 2. РАЗРАБОТКА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЕКТНОЙ ПРОЦЕДУРЫ АНАЛИЗА СЕТЕВОГО ТРАФИКА 25](#_Toc71124336)

[2.1 Содержательная и математическая постановка задачи анализа сетевого трафика 25](#_Toc71124337)

[2.2 Методы и алгоритмы решения задачи анализа сетевого трафика 29](#_Toc71124338)

[2.3 Анализ сетевого трафика при помощи машинного обучения 31](#_Toc71124339)

[2.4 Алгоритмы машинного обучения для решения задачи анализа сетевого трафика. 34](#_Toc71124340)

[Выводы по главе 2 37](#_Toc71124341)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕДУРЫ АНАЛИЗА СЕТЕВОГО ТРАФИКА ПРИ ПОМОЩИ МАШИННОГО ОБУЧЕНИЯ 38](#_Toc71124342)

[3.1. Концептуальное проектирование базы данных 38](#_Toc71124343)

[3.3. Физическое проектирование базы данных 43](#_Toc71124344)

[Выводы по главе 3 44](#_Toc71124345)

[ГЛАВА 4. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕДУРЫ ОБНАРУЖЕНИЯ ДЕФЕКТОВ ПЕЧАТНЫХ ПЛАТ 46](#_Toc71124346)

[4.1. Анализ функциональных требований 46](#_Toc71124347)

[4.2. Архитектура программы для автоматизации процедуры обнаружения дефектов печатных плат 55](#_Toc71124348)

[4.3. Разработка пользовательского интерфейса программы для автоматизации процедуры обнаружения дефектов печатных плат 58](#_Toc71124349)

[Выводы по главе 4 63](#_Toc71124350)

[Заключение 64](#_Toc71124351)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 66](#_Toc71124352)

[ПРИЛОЖЕНИЕ 1. Глоссарий (словарь терминов предметной области). 71](#_Toc71124353)

[ПРИЛОЖЕНИЕ 2. Листинг программы. 72](#_Toc71124354)

**Введение**

В наше время интернет вошел вплотную в жизнь современного общества. Каждый человек, каждая социальная группа, каждая компания и даже каждое государство использует интернет в своих целях. Но главная суть заключается в том, что происходит общая информационная интеграция. Для этого используют компьютерные сети, которые объединяются благодаря использованию интернета. И чем больше людей, предприятий и государств используют интернет для передачи информаций, тем больше появляются проблем в области защиты информации, которая передается по интернету, и компьютерных сетей. Сейчас имеются множество способов защитить компьютерные сети и информацию от воздействия вирусов, деятельностей хакеров и т.п. Одно из решений в области информационной защиты компаний и государств является анализ сетевого трафика, который передается внутри компьютерных сетей.

Автоматизация процедуры анализа сетевого трафика необходима для диагностики сетевых ресурсов с целью обнаружения сетевых аномалий, которые могут быть результатом попыток разрушить работоспособность компании, совершения несанкционированного доступа или же совершений краж информации с целью шпионажа. Для автоматизации процедуры анализа сетевого трафика используют специализированные системы анализа сетевых данных и обнаружения сетевых аномалий.

Особенности этих систем заключается в том, что выявление аномалий происходит в реальном времени и путем использования сигнатурных методов, однако новые виды угроз и аномалий могут спокойно быть необнаруженными. Также анализ, например, прошлогоднего трафика невозможен. Для выявления возможных новых сетевых аномалий и анализа трафика, который был в прошлом, могут быть использованы алгоритмы машинного обучения.

# ГЛАВА 1. АНАЛИЗ ПРОЕКТНОЙ ПРОЦЕДУРЫ АНАЛИЗА СЕТЕВОГО ТРАФИКА ПРИ ПОМОЩИ МАШИННОГО ОБУЧЕНИЯ

## Современные тенденции и проблемы выполнения процедуры анализа сетевого трафика

Современные системы для автоматизации анализа сетевого трафика представляют собой комплексы программных обеспечений, например, анализаторы сетевых протоколов, и технологических оборудований, например, серверы приложений и сетевые шлюзы. Для повышения эффективности безопасности и защищенности сетевых каналов компании используют различные методы обнаружения аномалий в сетевом трафике. Благодаря совершенствующимся ПО и оборудованиям анализ трафика происходит постоянно и непосредственно без участия пользователя. Основными методами обнаружения аномалий сетевого трафика являются [1]:

* Сигнатурный анализ;
* Экспертные системы;
* Поведенческий анализ;
* Статистические системы.

Все данные методы используются во многих IDS системах и показали свою эффективность при обнаружении аномалий сетевого трафика, однако основной недостаток данных IDS и используемых методов заключается в следующем: для применения данных систем для анализа и обнаружения аномалий в сетевом трафике компания должна иметь достаточно большой капитал и доходность, чтобы покупать данные системы и покупать обновления данных систем, т.к. методы, на которых основан анализ трафика, не могут выявить новые или даже предвидеть возможные новые аномалии сетевого трафика, тем самым обеспечить защиту сетевых каналов компании в будущем. Также весь процесс установки и обеспечения оборудованиями для работы анализаторов тоже дорогостоящая возможность. Всё это является основной проблемой выполнения анализа сетевого трафика.

Рассматриваемые в рамках данной выпускной квалификационной работы методы и алгоритмы для анализа сетевого трафика на основе машинного обучения являются решением проблем при выполнении анализа сетевого трафика, т.к. для работы данных алгоритмов достаточно данных трафика, которые собирает обычная программа – сниффер, которых на просторах интернета большое количество и многие из них являются бесплатными. Также для функционирования алгоритмов машинного обучения нужен лишь один компьютер, а не большое количество оборудований. Это обусловлено тем, что для функционирования алгоритмов машинного обучения необходим компьютер и данные сетевого трафика в «чистом» виде.

Таким образом, актуальность данной выпускной квалификационный работы обусловлена тем, что методы, которые сейчас используются для анализа сетевого трафика устаревши и требуют больших производительных мощностей. Также использование машинного обучения гарантирует безопасность и защищенность сетевых каналов от вторжений в будущем, т.к. алгоритмы обучаются и выявляют результат анализа сетевого трафика, обучив их лишь однажды. Использование машинного обучения для анализа сетевого трафика позволит существенно снизить стоимость данной процедуры, и обеспечить безопасность в будущем, даже при появлении новых аномалий, которых еще не успели изучить.

В данной работе будет показан результат эффективности использования различных алгоритмов машинного обучения для выполнения процедуры анализа сетевого трафика, произведена оценка эффективности каждого алгоритма.

## 1.2. Разработка функциональных и поведенческих моделей (IDEF0 и IDEF3) процедуры анализа сетевого трафика

Диаграмма IDEF0 является популярнейшим инструментом для создания функциональной схемы проектируемой процедуры, благодаря которой можно точно определить все процессы для представления требований к системе. На рис.1 представлена диаграмма IDEF0 (А-0) для процедуры анализа сетевого трафика при помощи машинного обучения.

1. Функции диаграммы IDEF0 (A-0).

На уровне А-0 диаграммы выполняется функция «Анализ сетевого трафика при помощи машинного обучения». Данная функция является главным процессом проектной процедуры и включает в себя при декомпозиции функции подготовки исходных данных и оценки алгоритмов машинного обучения для анализа сетевого трафика.

2. Входные потоки данных диаграммы IDEF0 (A-0).

Уровень А-0 диаграммы IDEF0 включает в себя следующие входные потоки данных:

* набор данных сетевого трафика NSL – KDD – данные сетевого трафика, включающий 41 информационный признак;
* тестовые данные сетевого трафика NSL – KDD – данные сетевого трафика необходимые для оценки эффективности алгоритмов машинного обучения.

3. Выходные потоки данных диаграммы IDEF0 (A-0).

Уровень А-0 включает в себя следующий поток выходных данных:

* результаты эффективности алгоритмов машинного обучения для анализа сетевого трафика – вывод по использованию алгоритмов исходя, из изображений ROC – кривых и результатов метрик каждого используемого алгоритма.

4. Потоки управления диаграммы IDEF0 (A-0).

Уровень А-0 включает в себя следующие потоки управления:

* ГОСТ Р ИСО/МЭК 27033-3-2014 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 3. Эталонные сетевые сценарии. Угрозы, методы проектирования и вопросы управления. Данный документ устанавливает аномалии и угрозы в сетевом трафике.

5. Потоки механизмов диаграммы IDEF0 (A-0).

* ЭВМ – комплекс средств, используемых для выполнения математических операций, обработки информации и управления;
* инженер – аналитик – специалист по анализу алгоритмов для автоматизации процедуры анализа сетевого трафика.

Диаграмма IDEF0 A0 представлена на рис.2:

1. Функции диаграммы IDEF0 (A0).

На уровне А0 выполняются следующие функции:

* «Подготовка исходных данных». В рамках данной функции в работе

происходит подготовка данных для дальнейшей работы алгоритмов;

* «Оценка алгоритмов машинного обучения для анализа сетевого трафика». В рамках данной функции происходит отображение изображений ROC – кривых и результатов метрик каждого используемого алгоритма.

2. Входные потоки данных диаграммы IDEF0 (A0).

* набор данных сетевого трафика NSL – KDD;
* тестовые данные сетевого трафика NSL – KDD.
* фреймданные для выполения работы алгоритмов – изменение формата данных, которые нужны для выполнения работы алгоритмов.

3. Выходные потоки данных диаграммы IDEF0 (A0).

* результаты эффективности алгоритмов машинного обучения для анализа сетевого трафика.

4. Потоки управления диаграммы IDEF0 (A0).

* ГОСТ Р ИСО/МЭК 27033-3-2014 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 3. Эталонные сетевые сценарии. Угрозы, методы проектирования и вопросы управления.

5. Потоки механизмов диаграммы IDEF0 (A0).

* ЭВМ;
* инженер – аналитик.

Диаграмма IDEF0 A1 представлена на рис.3:

1. Функции диаграммы IDEF0 (A1).

На уровне А1 выполняются следующие функции:

* «Формирование множества данных для выполнения работы алгоритмов». В рамках данной функции происходит кодировка данных для выполнения работы алгоритмов;
* «Верификация». В рамках данной функции происходит проверка целостности данных после выполнения кодировки.
* «Оценка эффективности алгоритмов машинного обучения для анализа сетевого трафика». В рамках данной функции происходит оценка результатов работы алгоритмов.

2. Входные потоки данных диаграммы IDEF0 (A1).

* набор данных сетевого трафика NSL – KDD;
* тестовые данные сетевого трафика NSL – KDD.
* результаты анализа сетевого трафика – изображения ROC – кривых и результаты метрик каждого используемого алгоритма.

3. Выходные потоки данных диаграммы IDEF0 (A1).

* фреймданные для выполнения работы алгоритмов.
* результаты эффективности алгоритмов машинного обучения для анализа сетевого трафика.
* кодированные данные сетевого трафика – измененные данные, где они имеют значения 0 и 1, для выполнения работы алгоритмов.

4. Потоки управления диаграммы IDEF0 (A1).

* ГОСТ Р ИСО/МЭК 27033-3-2014 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 3. Эталонные сетевые сценарии. Угрозы, методы проектирования и вопросы управления.

5. Потоки механизмов диаграммы IDEF0 (A1).

* ЭВМ;
* инженер – аналитик.

Диаграмма IDEF0 A2 представлена на рис.4:

1. Функции диаграммы IDEF0 (A2).

На уровне А2 выполняются следующие функции:

* «Обучение алгоритмов машинного обучения». В рамках данной функции в работе происходит обучение алгоритмов выявлять аномалии сетевого трафика;
* «Выполнение алгоритмов машинного обучения». В рамках данной функции происходит работа алгоритмов машинного обучения с тестовыми данными сетевого трафика.

2. Входные потоки данных диаграммы IDEF0 (A2).

* фреймданные для выполнения работы алгоритмов.

3. Выходные потоки данных диаграммы IDEF0 (A2).

* результаты анализа сетевого трафика;
* обученные алгоритмы машинного обучения – алгоритмы, обученные для выявления аномалий в сетевом трафике.

4. Потоки механизмов диаграммы IDEF0 (A2).

* ЭВМ;
* Инженер – аналитик.

IDEF 3 – диаграмма, с помощью которой происходит сбор информации и документирование процессов, выполняемых в рамках определенной процедуры а так же в ней отражаются действия, события и порядок выполнения функций.

На рисунке 5 представлена диаграмма IDEF3 проектной процедуры анализа сетевого трафика при помощи машинного обучения:

Основными процедурами на диаграмме IDEF3 являются:

* загрузка исходных данных для обучения и тестирования алгоритмов машинного обучения для анализа сетевого трафика;
* подготовка исходных данных для выполнения работы алгоритмов машинного обучения;
* обучение алгоритмов машинного обучения;
* выполнение работы алгоритмов машинного обучения на тестовой выборке данных;
* результаты работы алгоритмов машинного обучения для анализа сетевого трафика;
* оценка результатов работы алгоритмов машинного обучения для анализа сетевого трафика.

## 1.3 Обзор систем автоматизации процедуры анализа сетевого трафика

Для выполнения задачи анализа сетевого трафика используются различные IDS и сканеры уязвимостей, каждая из которых имеет свои преимущества и недостатки. Использование данных систем помогает инженеру – аналитику автоматизировать трудоемкий процесс. На данный момент существуют следующие программы для обнаружения аномалий в сетевом трафике [2]:

1. «Snort».Классическая IDS, которая анализирует трафик на совпадение с базой правил (сигнатур). Одна из известных IDS, на которой было реализовано множество модулей. Обнаружение аномалий сетевого трафика происходит при помощи использования статистического детектора аномалий.
2. «Bro». IDS с упором на обнаружение уже известных нарушений. Обнаружение аномалий происходит на основе проверок передаваемых данных на нормальность.
3. «OpenVAS». Фреймворк, сочетающий в себе сканер и решение для управления уязвимостями. Содержит базу с более чем 50000 уязвимостями, опознаёт указанные цели и проверяет их реакцию на применение соответствующих правил.
4. «XSpider». Классический сканер уязвимостей с базой, эвристиками, поиском аномального поведения. Для обнаружения аномалий в сетевом трафике используется эвристический метод определения типов и имен сервисов для определения настоящих имен и типов серверов.

Таким образом, в ходе обзора представленных на рынке систем анализа сетевого трафика для обнаружения аномалий было выявлено, что данные системы для анализа трафика используют методы для обнаружения аномалий методы, которые реагируют только на известные атаки. Для обеспечения безопасности сетевого трафика использованием данных систем необходимо поддерживать постоянную актуальность баз аномалий сетевого трафика.

Также с течением времени методы, которые используются для обнаружения аномалий, могут быть переобучены злоумышленниками таким образом, что атакующие действия будут рассматриваться как нормальные.

Поэтому можно сделать вывод, что на данный момент систем для анализа сетевого трафика, которые используют алгоритмы машинного обучения, либо не представлены на рынке, либо не имеют результатов эффективности их работы. Использование алгоритмов машинного обучения позволит удешевить использование таких систем и повысить безопасность сетевых каналов. С учетом этого, можно сделать вывод, что использование новых и актуальных алгоритмов, позволяющие обнаруживать нынешние и возможные будущие аномалии сетевого трафика, является важной задачей.

## 1.4 Цель и задачи автоматизации процедуры анализа сетевого трафика

Цель данной выпускной квалификационной работы - решение актуальной научно-технической задачи – анализ процедуры и разработка программного обеспечения для анализа сетевого трафика при помощи машинного обучения.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проведение обзора предметной области с помощью диаграмм IDEF0 и IDEF3.
2. Постановка задачи анализа сетевого трафика при помощи машинного обучения.
3. Обзор методов и алгоритмов решения задачи анализа сетевого трафика.
4. Составление алгоритма анализа сетевого трафика при помощи машинного обучения.
5. Обзор алгоритмов машинного обучения для анализа сетевого трафика.
6. Анализ функциональных требований и разработка программного обеспечения для решения задачи.

## Выводы по главе 1

В главе 1 описана актуальность разработки проекта для анализа сетевого трафика при помощи машинного обучения. Разработана функциональная модель процедуры анализа сетевого трафика, для данной процедуры разработаны диаграммы IDEF0, IDEF3.

В рамках данной главы был проведен обзор и сравнение имеющихся на рынке продуктов и методов для задачи анализа сетевого трафика.

Были поставлены цели и задачи выпускной квалификационной работы.

Была обоснована цель выпускной квалификационной работы – разработка программы для анализа сетевого трафика при помощи машинного обучения.

**ГЛАВА 2. РАЗРАБОТКА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЕКТНОЙ ПРОЦЕДУРЫ АНАЛИЗА СЕТЕВОГО ТРАФИКА**

**2.1 Содержательная и математическая постановка задачи** **анализа сетевого трафика**

Сетевой трафик — какой – либо объем информации, которая передается по компьютерной сети за определенный период времени [3]. В сетевом трафике имеются множество данных для определения их свойств.

Машинное обучение – это методики анализа данных, благодаря которым аналитическая система имеет возможность обучаться в ходе решения множества сходных задач. Идея машинного обучения заключается в том, что данные системы имеют способность учиться выявлять закономерности и принимать решения с минимальным участием человека [4]. Также можно сказать, что машинное обучение – это тренировка математической модели на данных, чтобы прогнозировать какое-то событие или явление на новых данных.

Существуют следующие виды атак, которые отображаются в сетевом трафике:

* DoS (от англ. Denial of Service) – отказ в обслуживании. Характерна генерация большого объема трафика, что приводит к перегрузке и блокированию сервера;
* R2L (от англ. Remote to Local) – получение доступа незарегистрированного пользователя к компьютеру со стороны удаленной машины;
* U2R (от англ. User to Root) – получение зарегистрированным пользователем привилегий локального суперпользователя (администратора);
* Probe – сканирование системы на наличие уязвимостей с целью их дальнейшей эксплуатации.

Алгоритмы машинного обучения, используемые в этой работе, будут иметь возможность выявлять любые из этих атак (аномалий).

Задача анализа сетевого трафика для обнаружения аномалий является классической задачей классификации.

Классификация — один из разделов машинного обучения, посвященный решению следующей задачи: имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества [5].

Классифицировать объект — значит, указать номер (или наименование класса), к которому относится данный объект [5].

Исходными данными для решения данной задачи будет являться датасет (библиотека) NSL – KDD.

Результирующими данными будут являться значения метрик для оценки эффективности работы алгоритмов [6]:

* доля правильных ответов алгоритма (accuracy)
* точность (precision);
* полнота (recall);
* F-мера (F-score),
* ROC – кривые (Receiver Operating Characteristic curve – кривая ошибок).

Для математической модели задачи анализа сетевого трафика при помощи машинного обучения введены следующие обозначения, представленные в табл. 1:

Таблица 1. Обозначения математической модели

|  |  |  |
| --- | --- | --- |
| **Обозначение компонента объекта и результата проектирования** | **Наименование компонента объекта и результата проектирования** | **Наименование соответствующего элемента математической модели** |
|  | *Данные сетевого трафика* | Множество описаний объектов |
|  | *Тип сетевых пакетов* | Множество номеров классов |
|  |  | Целевая зависимость – отображение |
|  |  | Алгоритм классификации |
|  | *Сетевой пакет* | Объект |
|  |  | Объекты обучающей выборки |

Пусть — множество описаний объектов, — конечное множество номеров классов. Существует неизвестная целевая зависимость — отображение, значения которой известны только на объектах конечной обучающей выборки. Алгоритм, способный классифицировать произвольный объект [5].

Типом входных данных в задаче классификации является признаковое описание. Каждый объект описывается набором своих характеристик, называемых признаками. В данной работе используются числовые и нечисловые признаки.

Признаком называется отображение, где — множество допустимых значений признака. Если заданы признаки, то вектор называется признаковым описанием объекта . Признаковые описания допустимо отождествлять с самими объектами. При этом множество называют признаковым пространством.

В зависимости от множества признаки делятся на следующие типы:

* Бинарный признак: ;
* Номинальный признак: — конечное множество;
* Порядковый признак: — конечное упорядоченное множество;
* Количественный признак: — множество действительных чисел.

После проведения классификации возможно получение четырех видов результатов [6]: TP (от англ. True Positive – истинно положительный), TN (от англ. True Negative – истинно отрицательный), FP (от англ. False Positive – ложно положительный) и FN (от англ. False Negative – ложно отрицательный). Эти результаты можно представить в виде матрицы ошибок в таблице 2, где 𝑦′– ответ алгоритма на объекте, а 𝑦 – истинная метка класса на этом объекте.

Таблица 2. Матрица ошибок

|  |  |  |
| --- | --- | --- |
|  | 𝑦 = 1 | 𝑦 = 0 |
| 𝑦′ = 1 | True Positive (TP) | False Positive (FP) |
| 𝑦′ = 0 | False Negative (FN) | True Negative (TN) |

Доля правильных ответов алгоритма (accuracy) показывает общую точность работы классификатора:

Точность (precision) показывает долю объектов, названных классификатором положительными и при этом действительно являющимися положительными:

Полнота (recall) показывает долю правильно помеченных положительных объектов среди всех объектов положительного класса:

Точность чувствительна к распределению данных, в то время как полнота – нет. Полнота не отражает, сколько объектов помечены как положительные неверно, а точность не дает никакой информации о том, сколько положительных объектов помечены неправильно.

F-мера (F-score) сочетает в себе вышеупомянутые две метрики – среднее гармоническое точности и полноты:, где – принимает значения в диапазоне, если приоритет отдается точности, и, если приоритет отдается полноте.

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

ROC-кривая или кривая ошибок – график, позволяющий оценить качество классификации, который отображает соотношение между чувствительностью (TPR, True Positive Rate) алгоритма и долей из объектов отрицательного класса, которые алгоритм предсказал неверно (FPR, False Positive Rate) при варьировании порога решающего правила:

**2.2 Методы и алгоритмы решения задачи анализа сетевого трафика**

Для выполнения задачи анализа сетевого трафика на данный момент применяются различные методы, каждый из которых представлен на рис. 6 [1]:

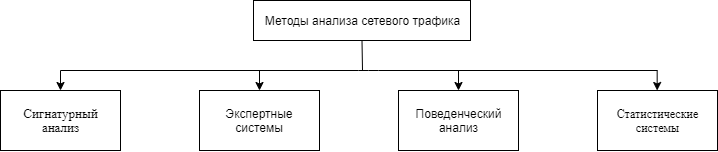


Рис.6. Методы анализа сетевого трафика

При использовании сигнатурного анализа можно обнаружить аномалии сетевого трафика на основе детерминированных признаков (сигнатур) атак. Достоинствами метода сигнатурного анализа является высокая скорость обработки запроса. Недостатками являются высокое количество ложных срабатываний, низкая точность выявления атак, отсутствие возможностей выявления новых атак, невозможность составления под каждую атаку собственной сигнатуры.

При использовании экспертных систем происходит кодировка информации об аномалиях по правилам: if (если) причина и then (то) решение. Когда все условия в левой части правила удовлетворены, выполняется действие (решение), заданное в правой его части. Основные недостатками являются недостаточная эффективность при работе с большими объемами данных и трудность составления зависимостей между данными.

Одним из вариантов анализа сетевого трафика является метод на основе поведенческого анализа. Суть данного метода заключается в следующем: имеется база данных сценариев атак, каждая из которых объединяет последовательность поведений, составляющих атаку. В любой момент времени существует возможность того, что в системе имеет место одно из этих подмножеств сценариев атак. Делается попытка проверки предположения об их наличии путем поиска информации в записях аудита. Результатом поиска является какое-то количество фактов, достаточное для подтверждения или опровержения гипотезы**.**Преимуществами данного метода является возможность уменьшения количество существенных обработок информации, тем самым увеличить скорость обнаружения атак. Основным недостатком является непосредственное участие человека при анализе сетевого трафика.

Использование статистических методов анализа информации было предусмотрено в первых системах обнаружения атак в 80 – х годах прошлого века. В настоящее время к статистическому анализу добавляются новые методики и алгоритмы. Статистические методы универсальны, поскольку для проведения анализа не требуется знания о возможных атаках и используемых ими уязвимостях. Основные преимущества статистического подхода — использование уже разработанного и зарекомендовавшего себя аппарата математической статистики и адаптация к поведению субъекта. Недостатками являются трудность определения пороговых значений для идентификации аномалии и нечувствительность к порядку следования событий.

В данной работе будут использоваться алгоритмы машинного обучения для анализа сетевого трафика на основе статистического подхода.

**2.3 Анализ сетевого трафика при помощи машинного обучения**

Для решения задачи анализа сетевого трафика при помощи машинного обучения будет применяется следующий алгоритм:

1. Загрузка данных NSL – KDD;
2. Предварительная обработка загружаемых данных;
3. Кодирование данных;
4. Выбор тринадцати характеристик в фрейме данных;
5. Обучение алгоритмов;
6. Работа алгоритмов на тестовой выборке;
7. Вывод результатов работы алгоритмов машинного обучения при анализе сетевого трафика.

Блок схема алгоритма анализа сетевого трафика при помощи машинного обучения представлена на рис. 7:

Для безошибочной и эффективной работы алгоритмов с загружаемыми данными необходимо эти данные закодировать. Кодировка данных будет происходить путем использования кодировщиков Label encoding и One – hot encoding.

Опишем основные шаги кодировки данных:

1. Загрузка кодировщика Label encoding;
2. Выявление текстовых значений в таблице;
3. Замена существующих текстовых значений числовыми значениями;
4. Загрузка кодировщика One – hot encoding;
5. Выявление зависимостей значений столбцов в категориальных данных;
6. Замена числовых значений на единицы и нули исходя из зависимости значений;

Блок-схема алгоритма кодировки данных представлена на рис. 8:

**2.4 Алгоритмы машинного обучения для решения задачи анализа сетевого трафика.**

В данной работе алгоритмы машинного обучения будут использованы с библиотеки scikit – learn.

Scikit – learn — это Python – библиотека, в которой имеется огромное количество алгоритмов для решения задач, связанных с машинным обучением и классификации данных.

В данной работе будут использованы следующие алгоритмы для анализа сетевого трафика при помощи машинного обучения: Random Forest (случайный лес), SVM (метод опорных векторов), KNN (метод k-ближайших соседей) и Ensemble Learning (ансамблевое обучение). Данные алгоритмы машинного обучения наиболее популярны и наиболее используемые для анализа различных данных.

Random Forest (случайный лес) – алгоритм машинного обучения, который использует большое количество ансамбля решающих деревьев, в результате чего появляется лес деревьев. Каждое дерево совершает классификацию данных. И выбирается та классификация данных, за которую проголосовало большее количество деревьев. Сами деревья тренируются методом бэггинга. Обучающие выборки генерируются при помощи метода бутстрэп.

Метод бутстрэп заключается в разбитии общей обучающей выборки на несколько выборок случайным образом путем выбора произвольных объектов.

Работа алгоритма «Случайный лес», который состоит из деревьев выглядит следующим образом:

Для каждого:

* + 1. Происходит генерация выборки при помощи использования метода бутстрэпа;
    2. Происходит построение дерева по выборке (метод бэггинга):
    - по заданному критерию выбирается лучший признак, происходит разбиение в дереве по критерию до исчерпания значений выборки;
    - построение дерева происходит до тех пор, пока в каждом листе не будет иметься объектов;
    - при каждом разбиении будет в начале выбираться – случайных признаков из исходных.

Далее происходит выбор решения путем голосования по большинству деревьев. Функция классификатора будет иметь вид

Преимущества алгоритма «Случайный лес»:

* + 1. Эффективная обработка данных, которые имеют большое количество признаков и классов;
    2. Стабильность работы алгоритма независимо от ввода новых данных;
    3. Независимость выполнения работы от характеристик признаков (категориальные и числовые);
    4. Независимость выполнения работы от наличия пропущенных значений и масштабируемость данных.

Недостатки алгоритма «Случайный лес»:

* + - 1. Выполнение алгоритма требует большое количество вычислительных ресурсов.
      2. Необходимость большого количества времени для обучения.

KNN (метод k-ближайших соседей) – один из популярных и основных алгоритмов машинного обучения для решения задач классификации и регрессии. Каждый объект представляется в виде вектора в некотором N – мерном пространстве. Исследуемый объект должен присваиваться тому классу, к которому относятся его ближайшие соседи, классы которых уже известны. Для определения «соседства» объектов применяют метрику расстояний – евклидовое расстояние.

Функция классификации данного алгоритма имеет вид:, где – расстояние, в пределах которого объекты считаются соседями, – выбранное число соседей, необходимое для присвоения к классу объекта, – метрика расстояний (евклидовое расстояние), – исследуемый объект, – выборка данных.

Преимущества метода k – ближайших соседей:

* + 1. Легко реализуемый алгоритм;
    2. Данный алгоритм – это алгоритм ленивого обучения, т.е. данный алгоритм может и не требовать обучающей выборки данных;
    3. Алгоритм легко работает при вводе новых данных;
    4. Для реализации алгоритма необходимо минимальное количество признаков;
    5. Не требует больших вычислительных мощностей.

Недостатки метода k – ближайших соседей:

* + 1. Алгоритм плохо работает с данным, которые имеют высокую размерность, т.к. происходит затруднение вычислений расстояний между объектами;
    2. Только при больших объемах данных алгоритм будет иметь высокую степень прогнозирования, т.к. расстояний между объектами меньше в огромных наборах данных;
    3. Хорошо работает только с числовыми признаками.

SVM (метод опорных векторов) – это классический алгоритм, который используется для решения задач классификации и регрессии. Суть данного алгоритма заключается в нахождении некой гиперплоскости, которая разделяет объекты на разные классы. При работе алгоритма должна сформироваться такая гиперплоскость, от которой ближайшие к ней объекты классов будут находится максимально далеко.

При нахождении оптимального вектора гиперплоскости решается задача квадратичного программирования. Функция классификации для метода опорных векторов будет иметь вид:, где уравнение описывает разделяющую гиперплоскость.

Преимущества метода опорных векторов:

1. Алгоритм хорошо работает с большими количествами признаков;

2. Алгоритм хорошо работает с небольшим объемом данных;

3. Алгоритм строит разделяющую гиперплоскость так, что позволяет уменьшить количество ошибок при решении задачи классификации;

4. Решение задачи квадратичного программирования всегда будет иметь единственное верное решение (ошибок будет меньше);

Недостатки метода опорных векторов:

1. Необходимость большого количество времени для обучения;

2. Высокая неустойчивость к шуму, т.к. какие – либо неправильные данные могут напрямую влиять на построение гиперплоскости;

3. Нет отбора признаков.

Ensemble Learning (ансамблевое обучение) – это метод, который использует несколько алгоритмов обучения. Для Ensemble Learning будет использовано объединение алгоритмов SVM, Random Forest и KNN.

Выводы по главе 2

Было приведена содержательная и математическая постановка задачи анализа сетевого трафика при помощи машинного обучения.

Был проведен обзор имеющихся методов и алгоритмов для решения задачи анализа сетевого трафика, разработан алгоритм анализа сетевого трафика при помощи машинного обучения, кодировки данных для работы алгоритмов машинного обучения. Также были рассмотрены более детально алгоритмы машинного обучения, которые будут использованы в данной работе: определили математические функции классификации для каждого алгоритма, преимущества и недостатки каждого используемого алгоритма.

**ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕДУРЫ АНАЛИЗА СЕТЕВОГО ТРАФИКА ПРИ ПОМОЩИ МАШИННОГО ОБУЧЕНИЯ**

**3.1. Разработка модели потока данных (DFD) и концептуальной модели базы данных**

Диаграмма DFD – диаграмма потоков данных, которая также дополняет функциональные модели данных. Благодаря использованию DFD – диаграмм имеется возможность более детального моделирования информационных систем.

Диаграмма DFD cостоит из моделей:

1. Модель окружения;
2. Логическая модель;
3. Модель поведения.

Модель окружения описывает информационные связи между основной функцией информационной системы и внешними объектами. На рис. 9 представлена модель окружения диаграммы DFD для процедуры анализа сетевого трафика при помощи машинного обучения. По данной модели можно определить участников в процедуре анализа сетевого трафика при помощи машинного обучения.

Логическая модель процедуры анализа сетевого трафика при помощи машинного обучения представлена на рис. 10. По данной модели можно определить функции, которые выполняются в рамках выполнения этой процедуры и их связи.

Модель поведения процедуры анализа сетевого трафика при помощи машинного обучения представлена на рис. 11. Данная модель дополняет логику функциональной модели, описывая тех, кто из внешних объектов будет выполнять и контролировать процесс анализа сетевого трафика при помощи машинного обучения.

Концептуальная модель базы данных – это информационная модель предметной области, в котором описываются взаимосвязанные понятия, которые используются для описания предметной области, вместе со свойствами и характеристиками, классификацией этих понятий по признакам, типам и т.п.

В данной работе будет использоваться готовая база данных (датасет) NSL – KDD [7].

NSL – KDD – база данных, являющаяся модернизацией базы данных KDD Cup ’99. Каждая запись – это образ сетевого соединения и имеет 42 информационных признака.

В данной работе база данных NSL\_KDD будет разделена на обучающую выборку (NSL\_KDD\_Test) и тестовую выборку (NSL\_KDD\_Train).

Для разбития базы данных будет использоваться кросс – валидация.

Кросс – валидация – метод, благодаря которому алгоритм будет способен работать на наборе данных, который не используется при обучении. В данной работе будет использовано k – fold кросс – валидация.

K – fold кросс – валидация – метод, который разбивает данные на несколько групп (рис. 12). Количество групп задается значением .

Алгоритм разбития данных k – fold кросс – валидацией:

* + - 1. База данных разбивается на непересекающихся одинаковых по объему частей;

1. Производится итераций. На каждой итерации происходят следующие действия:
2. Модель обучается на части обучающей выборки;
3. Модель тестируется на части данных, которые не участвовали в обучении.

Каждая из частей используется для тестирования один раз.

* + - * 1. Установление состава сущностей.

В таблице 3 представлено описание названий и описаний сущностей и описание количества экземпляров в таблице 3.

Таблица 3. Обозначение сущностей

|  |  |  |
| --- | --- | --- |
| **Обозначение сущности (связи)** | **Наименование сущности (связи)** | **Количество экземпляров** |
| S1 | NSL\_KDD | 1 |

Сущность «NSL\_KDD» содержит информацию о сетевом трафике для обучения и тестирования алгоритмов;

* + 1. Определение совокупности атрибутов и установление домена каждому атрибуту (таблица 4):

Таблица 4. Определение атрибутов и доменов

|  |  |  |
| --- | --- | --- |
| **Наименование сущности (связи)** | **Наименование атрибутов (описание атрибута)** | **Наименование домена** |
| NSL\_KDD | number\_of\_session (номер сессии) | Номер сессии |
| duration (продолжительность сессии) | Количество |
| protocol\_type (тип протокола) | Тип протокола |
| service (удаленный сервис) | Удаленный сервис |
| flag (статус соединения) | Статус соединения |
| src\_bytes (количество исходящих байт) | Количество |
| dst\_bytes (количество входящих байт) | Количество |
| land (подключение хоста) | Логика |
| wrong\_fragment (количество «неправильных» пакетов) | Количество |
| urgent (количество срочных пакетов) | Количество |
| hot (количество «hot» индикаторов) | Количество |
| num\_failed\_logins (количество неудачных авторизаций) | Количество |
| logged\_in (успешность авторизации) | Логика |
| num\_compromised (количество «скомпрометированных» условий) | Количество |
| root\_shell (вход по root) | Логика |
| su\_attempted (попытка входa по root) | Логика |
| num\_root (Количество доступов суперпользователя) | Количество |
| num\_file\_creations (количество оперaций по созданию файла) | Количество |
| num\_shells (количество сессий терминала) | Количество |
| num\_access\_files (количество операций по доступу к файлам) | Количество |
| num\_outbound\_cmds (количество исходящих команд в ftp – сессии) | Количество |
| is\_host\_login (логин в host) | Логика |
| is\_guest\_login (логин гостевой) | Логика |
| count (количество подключений на один хост в рамкaх текущей сессии за последние 2 секунды) | Логика |
| srv\_count (количество подключений к одному сервису в рамках текущей сессии за последние 2 секунды) | Логика |
| serror\_rate (процент от подключений с «SYN» ошибкой) | Процент |
| srv\_serror\_rate (процент от подключений с «SYN» ошибкой при подключении на один сервис) | Процент |
| rerror\_rate (процент от подключений с «REJ» ошибкой) | Процент |
| srv\_rerror\_rate (процент от подключений с «REJ» ошибкой при подключении на один сервис) | Процент |
| same\_srv\_rate (процент от подключения к одному и тому же сервису) | Процент |
| diff\_srv\_rate (процент от подключения к разным сервисам) | Процент |
| srv\_diff\_host\_rate (процент от подключения к разным хостам) | Процент |
| dst\_host\_count (количество подключений на один хост в рамках текущей сессии за последние 100 секунд) | Количество |
| dst\_host\_srv\_count (количество подключений на один сервер в рамках текущей сессии за последние 100 секунд) | Количество |
| dst\_host\_same\_srv\_rate (процент от подключения к одному и тому же сервису) | Процент |
| dst\_host\_diff\_srv\_rate (процент от подключения к разным сервисам) | Процент |
| dst\_host\_same\_src\_port\_rate (процент от подключения с одного и того же порта источника) | Процент |
| dst\_host\_srv\_diff\_host\_rate (процент от подключения к одному и тому же хосту) | Процент |
| dst\_host\_serror\_rate (процент от подключений с «SYN» ошибкой) | Процент |
| dst\_host\_srv\_serror\_rate (процент от подключений с «SYN» ошибкой при подключении нa один сервис) | Процент |
| dst\_host\_rerror\_rate (процент от подключений с «REJ» ошибкой) | Процент |
| dst\_host\_srv\_rerror\_rate (процент от подключений с «REJ» ошибкой при подключении нa один сервис) | Процент |
| intrusion\_type (тип атаки) | Тип атаки |

1. Определение для сущности первичного ключа (таблица 5):

Таблица 5. Таблица первичного ключа

|  |  |
| --- | --- |
| **Наименование сущности** | **Первичный ключ** |
| NSL – KDD | Номер сессии |

1. Определение типов значений для доменов (таблица 6):

Таблица 6. Тип данных доменов

|  |  |  |
| --- | --- | --- |
| **Наименование домена** | **Описание типов данных (максимальное количество знаков)** | **Примеры значений** |
| Номер сессии | Числовой тип данных (1000000) | 0, 1, ...., 4856123 |
| Количество | Числовой тип данных (10000) | 0, 1, ...., 99999 |
| Тип протокола | Текстовый тип данных (4) | tcp, icmp |
| Удаленный сервис | Текстовый тип данных (20) | telnet, smtp, eco\_i, private |
| Статус соединения | Текстовый тип данных (6) | SF, REJ |
| Логика | Числовой тип данных (1) | 0, 1 |
| Процент | Вещественный тип данных (3) | 0.39, 1, 100 |
| Тип атаки | Текстовый тип данных (20) | neptune, normal, satan |

1. Выявление связей между сущностями:

В данной базе данных существует одна сущность, поэтому связи между сущностями отсутствуют.

* 1. **Разработка логической модели базы данных**

Разработка логической модели базы данных представляет собой создание модели базы данных, в которой выделяются основные объекты базы данных и определяют связи между ними. Связь атрибутов представлены в таблице 7.

Таблица 7. Связь атрибутов

|  |  |  |
| --- | --- | --- |
| **Наименование сущности (связи)** | **Наименование атрибутов** | **F1** |
| NSL\_KDD | number\_of\_session | \* |
| duration | ← |
| protocol\_type | ← |
| service | ← |
| flag | ← |
| src\_bytes | ← |
| dst\_bytes | ← |
| land | ← |
| wrong\_fragment | ← |
| urgent | ← |
| hot | ← |
| num\_failed\_logins | ← |
| logged\_in | ← |
| num\_compromised | ← |
| root\_shell | ← |
| su\_attempted | ← |
| num\_root | ← |
| num\_file\_creations | ← |
| num\_shells | ← |
| num\_access\_files | ← |
| num\_outbound\_cmds | ← |
| is\_host\_login | ← |
| is\_guest\_login | ← |
| count | ← |
| srv\_count | ← |
| serror\_rate | ← |
| srv\_serror\_rate | ← |
| rerror\_rate | ← |
| srv\_rerror\_rate | ← |
| same\_srv\_rate | ← |
| diff\_srv\_rate | ← |
| srv\_diff\_host\_rate | ← |
| dst\_host\_count | ← |
| dst\_host\_srv\_count | ← |
| dst\_host\_same\_srv\_rate | ← |
| dst\_host\_diff\_srv\_rate | ← |
| dst\_host\_same\_src\_port\_rate | ← |
| dst\_host\_srv\_diff\_host\_rate | ← |
| dst\_host\_serror\_rate | ← |
| dst\_host\_srv\_serror\_rate | ← |
| dst\_host\_rerror\_rate | ← |
| dst\_host\_srv\_rerror\_rate | ← |
| intrusion\_type | ← |

**3.3. Физическое проектирование базы данных**

База данных (датасет) NSL – KDD для обучения и тестирования алгоритмов машинного обучения имеется в свободном доступе на сайте Массачусетского технологического института (Massachusetts Institute of Technology) и площадке GitHub.

Данная база данных имеется во многих форматах, что упрощает работу с ней, особенно, для работы с машинным обучением. Также данную базу данных можно подключить напрямую, благодаря чему можно обойтись без использования сторонних программ.

На рис. 13 представлена часть базы данных NSL – KDD в формате CSV.

* 1. **Описание файловых форматов представления входных и выходных данных процедуры анализа сетевого трафика**

В данной работе будет использована база данных NSL – KDD в формате CSV.

CSV – формат (comma separated values) – формат для представления табличных данных. Данный формат используется для хранения таблиц в текстовом виде. Строка в данном формате представляет с собой значения столбцов в строке таблицы. Значения столбцов разделяются запятыми.

Для работы с CSV – файлами для машинного обучения используется модуль csv в языке программирования Python.

Выходные данные будут представлены в виде текста, где каждая строка будет описывать результат одной и метрик для анализа эффективности работы алгоритма машинного обучения.

Также выходными данными будут представлены ROC – кривые, описывающие графически результаты метрик алгоритмов машинного обучения.

### Выводы по главе 3

В рамках данной главы была описана база данных NSL – KDD, разработана концептуальная модель базы данных, определены и описаны атрибуты и выявлены отношения между указанными атрибутами. На основе концептуальной модели базы данных была разработана логическая модель базы данных

Также был описан метод разделения базы данных на обучающую выборку и тестовую выборку для обучения и теста алгоритмов машинного обучения.

Были описаны и указаны форматы входных и выходных данных для процедуры анализа сетевого трафика при помощи машинного обучения.

# ГЛАВА 4. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕДУРЫ АНАЛИЗА СЕТЕВОГО ТРАФИКА

## 4.1. Архитектура программы для автоматизации анализа сетевого трафика при помощи машинного обучения

Для разработки программы анализа сетевого трафика при помощи машинного обучения будут использоваться:

* + 1. Язык программирования – Python. Python – это высокоуровневый язык программирования, который используется во многих сферах IT, например, машинное обучение, нейронные сети, разработка web – приложений, приложений и т.д.
    2. Web – приложение – Jupiter Notebook. Jupiter Notebook – инструмент для разработки и представления проектов в области наук о данных. Данный инструмент объединяет код и вывод в виде одного документа, который содержит текст, математические уравнения и изображения. Благодаря Jupiter Notebook обеспечивается быстрый и последовательный процесс разработки, т.к. вывод для каждого блока программы показывается сразу же, что позволяет более детально показать и просмотреть работу программы в целом;
    3. Python – библиотеки:

1. Pandas – программная библиотека на языке Python, которая используется для обработки и анализа данных;
2. NumPy – программная библиотека на языке Python, которая используется для операций с большими массивами данных и матриц, также в данной библиотеке имеются числовые наборы и наборы математических функций;
3. Scikit – learn — это Python – библиотека, в которой имеется огромное количество алгоритмов для решения задач, связанных с машинным обучением и классификации данных;
4. Matploitlib – это основная библиотека, которая используется для построения графиков в Python.

В таблице 8 указаны описания функций, которые используются в каждой python – библиотекой в программе:

Таблица 8. Описание функций

|  |  |  |
| --- | --- | --- |
| Библиотека | Функция | Описание |
| *Pandas* |  |  |
|  | *Чтение данных* | Чтение данных в формате csv |
|  | *Вывод данных* | Вывод информации о данных (тип, количество и т.д.) |
|  | *Добавление данных* | Добавление недостающих данных в основной фрейм данных |
|  | *Изменение данных* | Изменение текстовых данных на числовые данные |
| *NumPy* |  |  |
|  | *Вывод матриц неточностей* | Вывод матриц неточностей для каждого алгоритма машинного обучения |
| *Scikit – learn* |  |  |
|  | *Кодировка данных* | Кодирование данных в значения 0 и 1 (Label Encoding и One – Hot Encoding) |
|  | *Масштабирование данных* | Обработка данных для работы алгоритмов машинного обучения |
|  | *Выбор признаков* | Выбор 13 – ти признаков данных для работы алгоритмов машинного обучения |
|  | *Алгоритм «Случайный лес»* | Вызов алгоритма «Случайный лес» для классификации данных |
|  | *Метод «KNN»* | Вызов метода k – ближайших соседей» для классификации данных |
|  | *Метод «SVM»* | Вызов метода опорных векторов для классификации данных |
|  | *Ансамблевое обучение* | Вызов трех алгоритмов для классификации данных путем ансамблевого обучения |
|  | *Валидация данных* | Выполнений k – fold кросс валидации данных для работы алгоритмов |
|  | *Вывод метрик* | Вывод метрик для последующей оценки эффективности алгоритмов машинного обучения |
| *Matploitlib* |  |  |
|  | *Вывод ROC – кривых* | Вывод ROC – кривых для последующей оценки эффективности алгоритмов машинного обучения |

## 

## 4.2. Разработка пользовательского интерфейса программы для автоматизации процедуры анализа сетевого трафика при помощи машинного обучения

В web – приложении Jupiter Notebook вывод происходит после каждого блока (ячейки) программы (рис. 14).

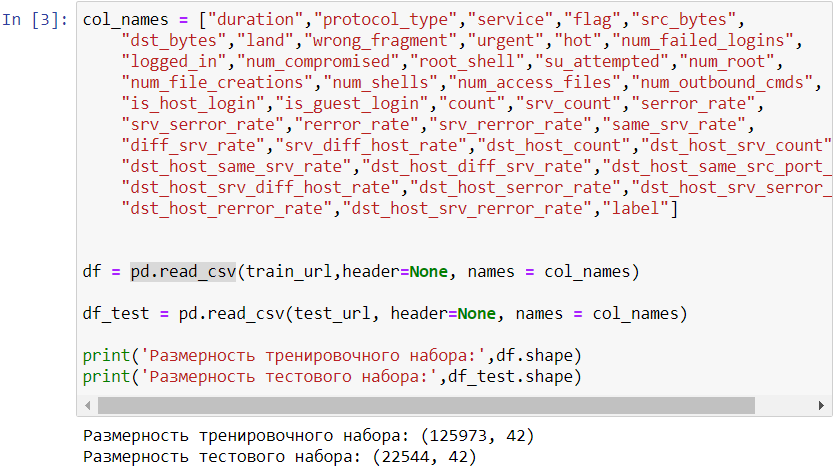


Рис. 14. Вывод данных после некоторого блока программы.

Пользовательский интерфейс программы анализа сетевого трафика при помощи машинного обучения имеет следующие элементы:

1. Данные о размерности наборов. Указывается размерность (количество строк и количество столбцов) данных тренировочного и тестового наборов.
2. Таблица первых пяти строк тестового набора. Показывается визуально информация тестового набора.
3. Данные о метках тестового и тренировочного наборов. Указывается название меток каждой строки и количество строк с данной меткой, также указывается тип данных.
4. Данные о текстовых данных в наборе. Указывается количество значений в каждом столбце тестового и тренировочного наборов, имеющий текстовые данные.
5. Таблицы первых пяти строк текстовых данных в обучающем наборе. Указываются таблица текстовых данных в первых пяти строк обучающего набора и таблица числовых данных после преобразования текстовых данных.
6. Таблица первых пяти строк обучающего набора после кодировки методом One – Hot Encoding. Указываются числовые значения по каждому столбцу.
7. Данные о размерности данных по каждому виду атаки в тренировочном и тестовом наборах. Указывается количество строк и столбцов данных по каждому виду атаки.
8. Данные о выбранных 13 – ти функциях для работы алгоритмов. Указываются наименования столбцов для каждого типа атаки для работы алгоритмов.
9. Матрица неточностей. Указываются матрицы неточностей каждого алгоритма после проведения классификации по каждому типу атаки.
10. Результаты метрик. Указываются значения метрик каждого алгоритма по каждому виду атаки.
11. ROC – кривые. Указываются графики ROC – кривых и значение AUC каждого алгоритма по каждому виду атаки.

Экранные формы элементов пользовательского интерфейса представлены на рис. 15 – 25:



Рис. 15. Данные о размерности наборов.

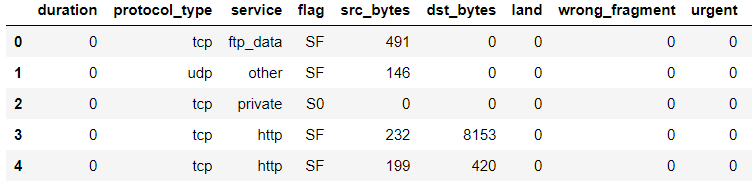


Рис. 16. Таблица первых пяти строк тестового набора.

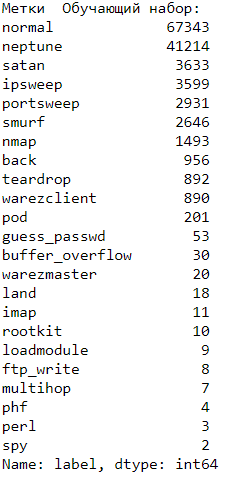
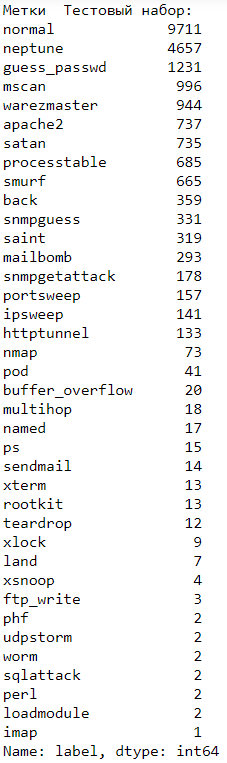
 

Рис. 17. Данные о метках в наборах.

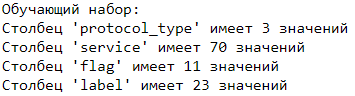
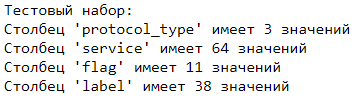
 

Рис. 18. Данные о текстовых данных в наборах.

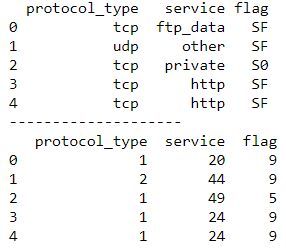


Рис. 19. Таблицы первых пяти строк текстовых данных в обучающем наборе.

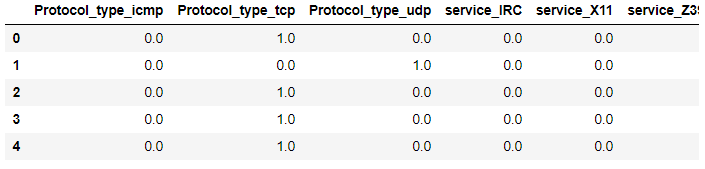


Рис. 20. Таблица первых пяти строк обучающего набора после кодировки методом One – Hot Encoding.

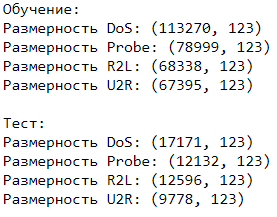


Рис. 21. Данные о размерности данных по каждому виду атаки в тренировочном и тестовом наборах.

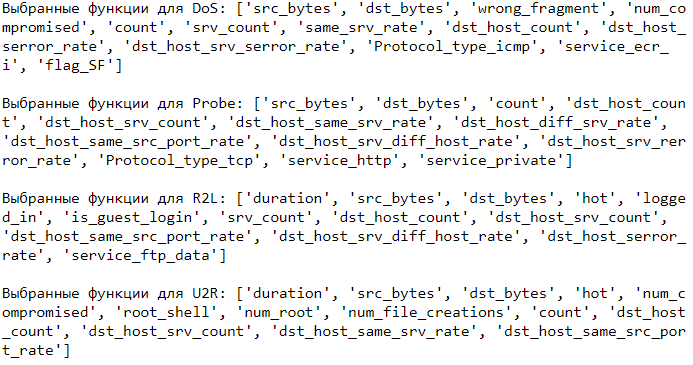


Рис. 22. Данные о выбранных 13 – ти функциях для работы алгоритмов.

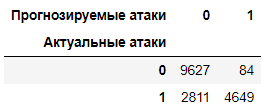


Рис. 23. Пример матрицы неточностей.

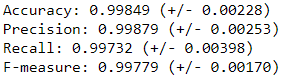


Рис. 24. Пример результатов метрик.

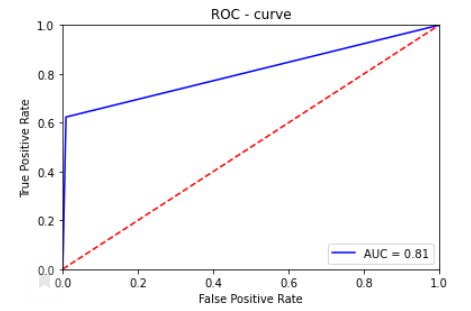
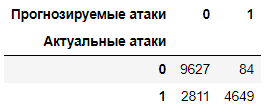
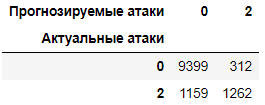


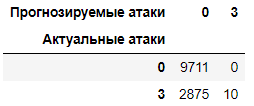
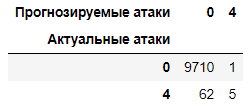
Рис. 25. Пример ROC – кривой.

## Примеры решения задачи анализа сетевого трафика при помощи машинного обучения

Матрицы неточностей после выполнения классификации алгоритмом «Random Forest» («Случайный лес») представлены на рис. 26:

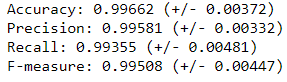
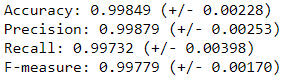
а) б)

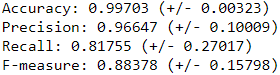
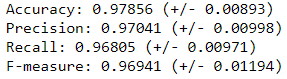
в) г)

Рис. 26. Матрицы неточностей алгоритма «Случайный лес»: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

Результаты метрик алгоритма «Random Forest» («Случайный лес») представлены на рис. 27:



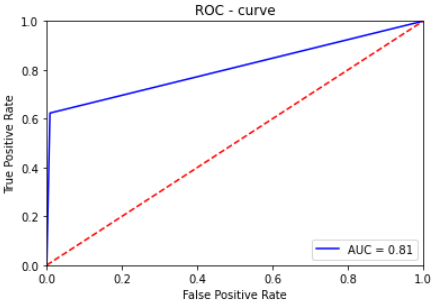
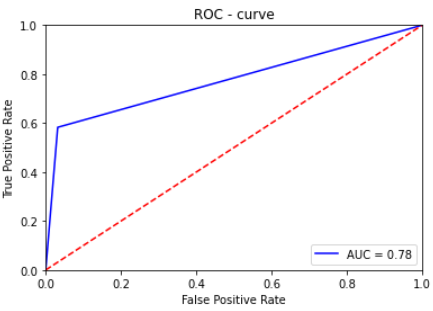
а) б)



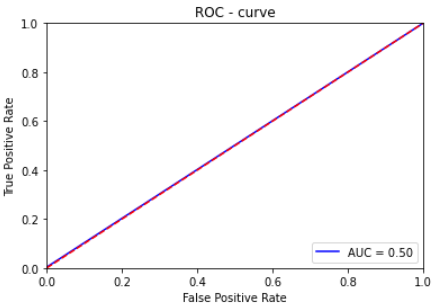
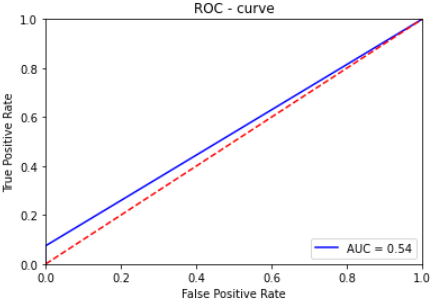
в) г)

Рис. 27. Результаты метрик алгоритма «Случайный лес»: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

ROC – кривые алгоритма «Random Forest» («Случайный лес») представлены на рис. 28:

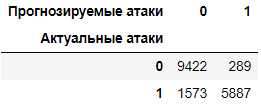
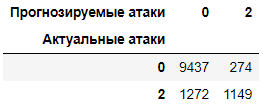
а) б)

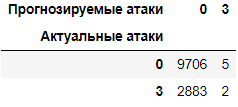
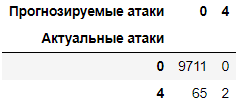
в) г)

Рис. 28. ROC – кривые алгоритма «Случайный лес»: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

Матрицы неточностей после выполнения классификации методом «KNN» (метод k – ближайших соседей) представлены на рис. 29:

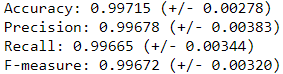
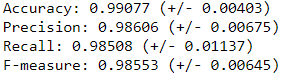
а) б)

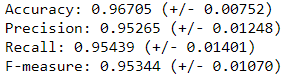
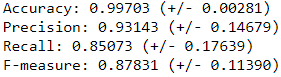
в) г)

Рис. 29. Матрицы неточностей метода k – ближайших соседей: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

Результаты метрик метода «KNN» (метод k – ближайших соседей) представлены на рис. 30:

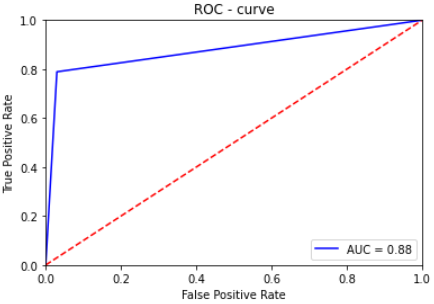
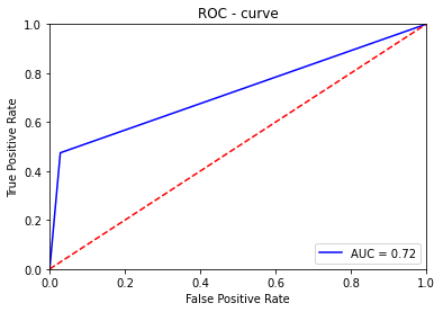
а) б)

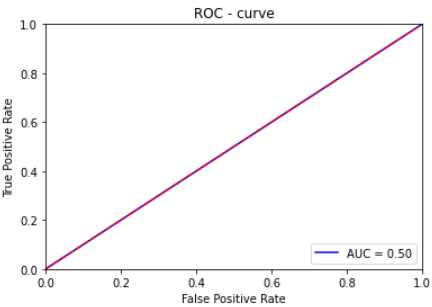
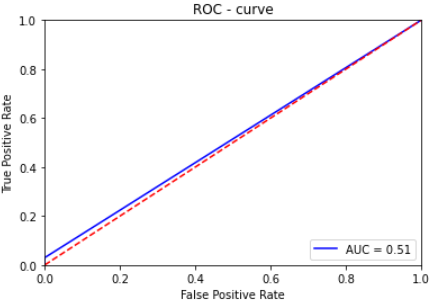
в) г)

Рис. 30. Результаты метрик метода k – ближайших соседей: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

ROC – кривые метода «KNN» (метод k – ближайших соседей) представлены на рис. 31:

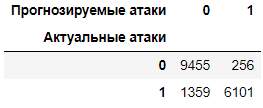
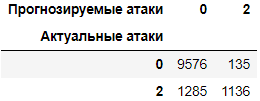
а) б)

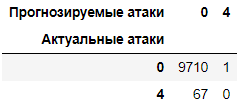
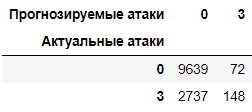
в) г)

Рис. 31. ROC – кривые метода k – ближайших соседей: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

Матрицы неточностей после выполнения классификации методом «SVM» (метод опорных векторов) представлены на рис. 32:

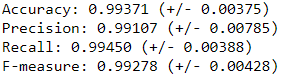
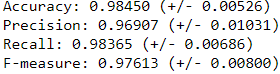
а) б)



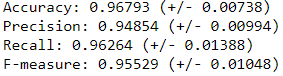
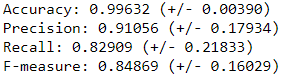
в) г)

Рис. 32. Матрицы неточностей метода опорных векторов: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

Результаты метрик метода «SVM» (метод опорных векторов) представлены на рис. 33:

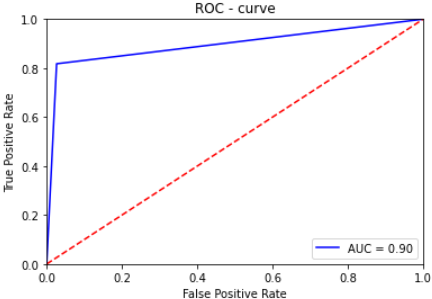
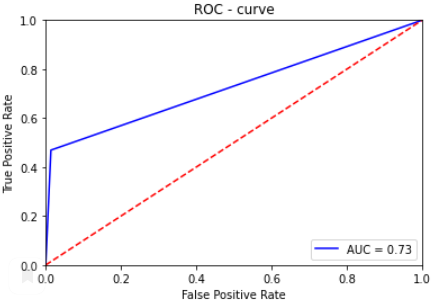
а) б)

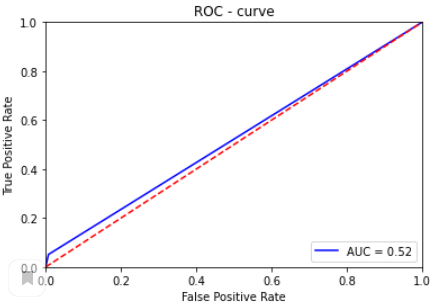
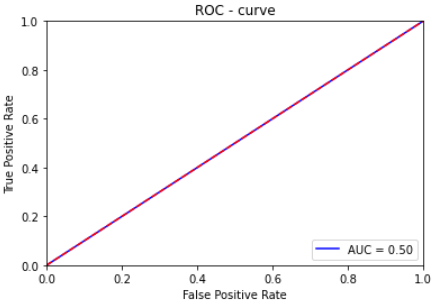
в) г)

Рис. 33. Результаты метрик метода опорных векторов: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

ROC – кривые метода «SVM» (метод опорных векторов) представлены на рис. 34:

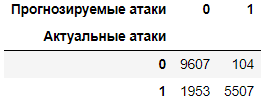
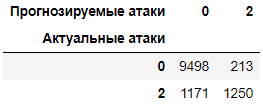
а) б)

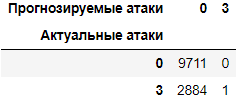
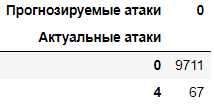
в) г)

Рис. 34. ROC – кривые метода опорных векторов: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

Матрицы неточностей после выполнения классификации ансамблевым обучением представлены на рис. 35:

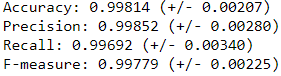
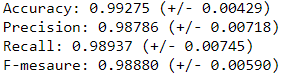
а) б)

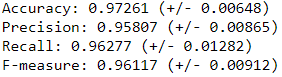
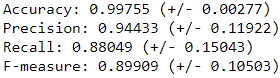
в) г)

Рис. 35. Матрицы неточностей ансамблевого обучения: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

Результаты метрик ансамблевого обучения представлены на рис. 36:

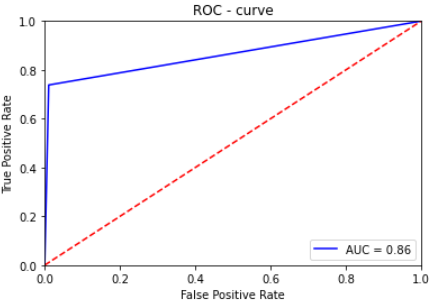
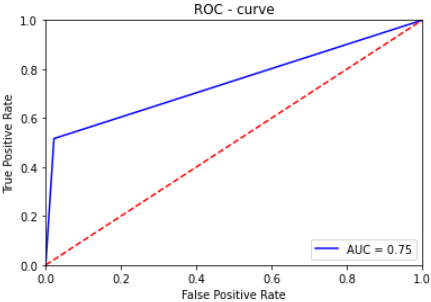
а) б)

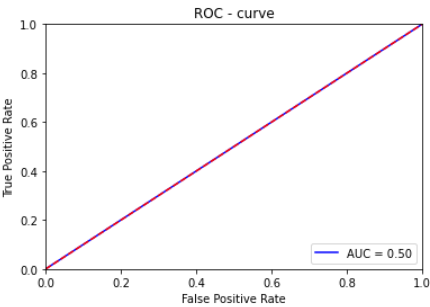
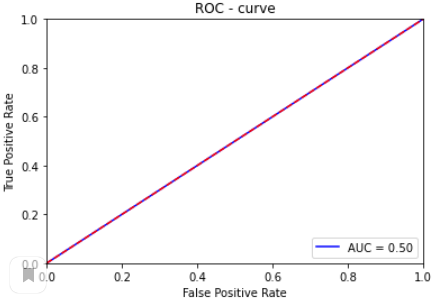
в) г)

Рис. 36. Результаты метрик ансамблевого обучения: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

ROC – кривые ансамблевого обучения представлены на рис. 37:

а) б)

в) г)

Рис. 37. ROC – кривые ансамблевого обучения: а) DoS – атака; б) Probe – атака; в) R2L – атака; г) U2R – атака.

## Исследование эффективности алгоритмов для решения задачи анализа сетевого трафика

Для исследования эффективности алгоритмов машинного обучения для решения задачи анализа сетевого трафика результаты метрик и значений ROC – кривых алгоритмов сгруппируем в таблицы 9 – 13.

Таблица 9. Значения метрики accuracy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритмы/Классы** | **DoS** | **Probe** | **R2L** | **U2R** |
| **Random Forest** | 0.99 | 0.99 | 0.97 | 0.99 |
| **KNN** | 0.99 | 0.99 | 0.96 | 0.99 |
| **SVM** | 0.99 | 0.98 | 0.96 | 0.99 |
| **Ensemble Learning** | 0.99 | 0.99 | 0.97 | 0.99 |

Таблица 10. Значения метрики precision

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритмы/Классы** | **DoS** | **Probe** | **R2L** | **U2R** |
| **Random Forest** | 0.99 | 0.99 | 0.97 | 0.96 |
| **KNN** | 0.99 | 0.98 | 0.95 | 0.93 |
| **SVM** | 0.99 | 0.96 | 0.94 | 0.91 |
| **Ensemble Learning** | 0.99 | 0.98 | 0.95 | 0.94 |

Таблица 11. Значения метрики recall

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритмы/Классы** | **DoS** | **Probe** | **R2L** | **U2R** |
| **Random Forest** | 0.99 | 0.99 | 0.96 | 0.82 |
| **KNN** | 0.99 | 0.98 | 0.95 | 0.85 |
| **SVM** | 0.99 | 0.98 | 0.96 | 0.82 |
| **Ensemble Learning** | 0.99 | 0.98 | 0.96 | 0.88 |

Таблица 12. Значения метрики F – measure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритмы/Классы** | **DoS** | **Probe** | **R2L** | **U2R** |
| **Random Forest** | 0.99 | 0.99 | 0.96 | 0.88 |
| **KNN** | 0.99 | 0.98 | 0.95 | 0.87 |
| **SVM** | 0.99 | 0.97 | 0.95 | 0.84 |
| **Ensemble Learning** | 0.99 | 0.98 | 0.96 | 0.89 |

Таблица 13. Значения метрики AUC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритмы/Классы** | **DoS** | **Probe** | **R2L** | **U2R** |
| **Random Forest** | 0.81 | 0.78 | 0.50 | 0.54 |
| **KNN** | 0.88 | 0.72 | 0.50 | 0.51 |
| **SVM** | 0.90 | 0.73 | 0.52 | 0.50 |
| **Ensemble Learning** | 0.86 | 0.75 | 0.50 | 0.50 |

Рассмотрев таблицы и матрицы неточностей можно сделать вывод о том, что алгоритмы хорошо определяют DoS и Probe атаки, а R2L и U2R атаки определяют плохо. Из – за несбалансированных (недостаточных) обучающих данных все алгоритмы классифицируют данные с атаками U2R и R2L как нормальные. Для решения проблемы несбалансированности данных необходимо дублирование данных или переход к бинарной классификации, например, «normal» и «not normal».

При рассмотрении только сбалансированных классов (DoS и Probe) лучшее определение DoS – атак показал метод опорных векторов («SVM»), лучшее определение Probe – атак показал алгоритм «случайный лес» («Random forest»).

Однако при использовании метода опорных векторов необходимо совершить подготовку (нормализацию) данных, что при использовании алгоритма «случайный лес» данную подготовку можно исключить. Также алгоритм «случайный лес» намного быстрее производит работу, чем метод опорных векторов, т.к. в основе первого алгоритма имеется возможность работы сразу нескольких параллельных вычислительных процессов и в основе алгоритма используются вычислительно лёгкие деревья решений, а в основе второго метода используется вычислительно сложный алгоритм бинарной классификации, где происходит сравнение каждого класса с каждым, что, впоследствии, увеличивает время работы анализа сетевого трафика.

Таким образом, можно сделать вывод о том, что для анализа сетевого трафика при отсутствии больших вычислительных мощностей и уменьшении времени работы лучше всего использовать алгоритм «случайный лес».

### Выводы по главе 4

В главе 4 было проведено описание архитектуры программы для автоматизации анализа сетевого трафика при помощи машинного обучения. В рамках данной главы были описаны используемые библиотеки, инструменты, функции обнаружения анализа сетевого трафика при помощи машинного обучения. Также был описан пользовательский интерфейс программы, представлены элементы пользовательского интерфейса. В главе были рассмотрены примеры работы программы, на основании которых можно сделать выводы об эффективности использования алгоритмов машинного обучения для анализа сетевого трафика и, что программа для анализа сетевого трафика является работоспособной и выполняет указанные требования.

# ЗАКЛЮЧЕНИЕ

Основные результаты выполнения ВКР заключаются в следующем:

1. Проведен обзор предметной области с помощью диаграмм IDEF0 и а IDEF3, рассмотрены имеющиеся на рынке программные обеспечения для анализа сетевого трафика, указана актуальность задачи анализа сетевого трафика при помощи машинного обучения.
2. Сформулирована математическая и содержательная постановка задачи анализа сетевого трафика.
3. Проведен обзор используемых методов и алгоритмов решения задачи анализа сетевого трафика, разработан алгоритм решения задачи анализа сетевого трафика при помощи машинного обучения, алгоритм кодировки данных для работы алгоритмов машинного обучения, описаны и рассмотрены алгоритмы машинного обучения для анализа сетевого трафика.
4. Описана информационная база данных для решения задачи анализа сетевого трафика при помощи машинного обучения.
5. Описана архитектура программа для автоматизации решения задачи анализа сетевого трафика и разработана программа на языке Python на web – приложении Jupiter Notebook.
6. Выполнен обзор результатов работы алгоритмов машинного обучения и сформулирован вывод об использовании алгоритмов машинного обучения для анализа сетевого трафика.

**CONCLUSION**

The main results of the WRC implementation are as follows:

1. An overview of the subject area with the help of IDEF0 and a IDEF3 diagrams was carried out, the software available on the market for the analysis of network traffic was considered, the relevance of the problem of analyzing network traffic using machine learning was indicated.

2. A mathematical and meaningful statement of the problem of network traffic analysis has been formulated.

3. A review of the methods and algorithms used for solving the problem of network traffic analysis is carried out, an algorithm for solving the problem of network traffic analysis using machine learning, a data encoding algorithm for the operation of machine learning algorithms is developed, machine learning algorithms for analyzing network traffic are described and considered.

4. An information database for solving the problem of analyzing network traffic using machine learning is described.

5. The architecture of a program for automating the solution of the problem of analyzing network traffic is described and a program in Python on the Jupiter Notebook web application is developed.

6. A review of the results of the work of machine learning algorithms is carried out and a conclusion is formulated about the use of machine learning algorithms for analyzing network traffic.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Пирогова Е. В. Проектирование и технология печатных плат:

Учебник. — М.: ФОРУМ: ИНФРА-М, 2005. — 560 с. — (Высшее образование). — [ISBN 5-16-001999-5](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5160019995). — [ISBN 5-8199-0138-X](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/581990138X).<http://echemistry.ru/literatura/literatura/pechatnye-platy1/avtomatizaciya-v-proektirovanii-i-proizvodstve-pechatnyh-plat-radioelektronnoj-apparatury.html>

1. Бахтин Б.И. Автоматизация в проектировании и производстве печатных

плат радиоэлектронной аппаратуры, 1979

1. Internet. Aplite – система автоматизации оптического контроля

[Электронный ресурс] (<https://www.tech-e.ru/2005_4_28.php>)

1. Internet. Phiplastic – семейство программ для обработки сканированных

изображений фотошаблонов и печатных плат [Электронный ресурс] (<http://www.eurointech.ru/eda/other/phiplastic/Phiplastic.phtm>)

1. Жигалов А.Т. Конструирование и технология печатных плат. Учебное

пособие для радиотехнических специальностей вузов. – М: Высшая школа, 1973, -216 с.

1. Вьюгин В. Математические основы теории машинного обучения и

прогнозирования. – МЦМНО, 2013. – 390 с.

1. Медведев А. Мир Электроники. Технология производства печатных

КМплат.– Техносфера, 2005.

1. ГОСТ Р 56251-2014 Платы печатные. Классификация дефектов
2. Галушкин А. И. Синтез многослойных систем распознавания образов. –

М.: Энергия, 1974.

1. Калацкая Л. В., Новиков В. А., Садов В. С. Организация и обучение

искусственных нейронных сетей: Экспериментальное учеб. пособие. — Минск: Изд-во БГУ, 2003. — 72 c.

1. Бонгард М. М.  Проблемы узнавания. — М.: Физматгиз, 1967.
2. Internet. Сверточные нейронные сети: структура, топология

[Электронный ресурс] (<https://habr.com/ru/post/348000/>)

1. Internet. Сверточные нейронные сети [Электронный ресурс]

(<https://programforyou.ru/poleznoe/convolutional-network-from-scratch-part-zero-introduction>)

1. Internet. Методы обратного распространения ошибки: математика,

примеры, код [Электронный ресурс] (<https://neurohive.io/ru/osnovy-data-science/obratnoe-rasprostranenie/>)

1. Internet. Обратное распространение ошибки [Электронный ресурс]

(<http://neerc.ifmo.ru/wiki/index.php?title=Обратное_распространение_ошибки>)

1. Internet. Алгоритм обратного распространения ошибки [Электронный

ресурс] (<http://www.aiportal.ru/articles/neural-networks/back-propagation.html>)

1. Internet. Нейронная сеть – обучение ИНС с помощью алгоритма

обратного распространения [Электронный ресурс] (<http://robocraft.ru/blog/algorithm/560.html>)

1. Internet. Метод Оцу [Электронный ресурс]

(<https://ru.wikipedia.org/wiki/Метод_Оцу>)

1. В.Т. Фисенко, Т.Ю. Фисенко, Компьютерная обработка и распознавание

изображений: учеб. пособие. - СПб: СПбГУ ИТМО, 2008. – 192 с.

1. Федоров, А. Бинаризация черно-белых изображений: состояния и

перспективы развития [Электронный ресурс] (<http://iu5.bmstu.ru/~philippovicha/ITS/IST4b/ITS4/Fyodorov.htm>)

1. Бенсон, Дж. Стратегии обнаружения дефектов. Рентгеновский контроль/

Дж. Бенсон // Печатный монтаж. - 2007. - № 3. - С. 34 - 36.

1. Данилова, Е.А. Классификация дефектов печатных плат / Е.А. Данилова

// Труды международного симпозиума «Надежность и качество». - Пенза: Изд-во Пенз. гос. ун-та. - 2013. - Т. 1. - С. 325-328.

1. Данилова, Е.А. Обзор методов обнаружения опасных технологических

дефектов в электронных платах /Е.А. Данилова // Труды международного симпозиума «Надежность и качество». - Пенза: Изд-во Пенз. гос. ун-та. - 2013. -Т. 1. - С. 331-335.

1. Машиностроение. Энциклопедия. Т. III-7. Измерения, контроль,

испытания и диагностика / Ред. совет: К.В. Фролов (пред) / В.В. Клюев, Ф.Р. Соснин, В.Н. Филинов и др; Под общ. ред. В.В. Клюева. - М.: Машиностроение, 1996. - 464 с.

1. Медведев, А.М. Технология производства печатных плат /A.М.Медведев.- М.: Техносфера, 2005. - 360 с.
2. Фисенко, В.Т., Фисенко, Т.Ю. Компьютерная обработка и

распознавание изображений: учеб. пособие / В.Т. Фисенко, Т.Ю. Фисенко. - СПб: СПбГУ ИТМО, 2008. - 192 с.

1. Шухард, П. Особенности выбора системы оптического контроля / П.

Шухард // Производство электроники: технологии, оборудование, материалы. -2011. - № 6. - С. 54 - 56.

1. Григорьев, А.В., Юрков, Н.К., Затылкин, А.В., Данилова, Е.А.,

Држевецкий, А.Л. Структурное обнаружение и различие вырывов проводящего рисунка печатных плат / А.В. Григорьев, Н.К. Юрков, А.В. Затылкин, Е.А. Данилова, А.Л. Држевецкий // Известия высших учебных заведений. Поволжский регион. Технические науки. - 2013. - № 4. - С. 97-108.

1. Черемисин, И., Задорин, А. Создание электронного представления

печатной платы по ее сканированному изображению / И. Черемисин, А. Задорин // Технологии в электронной промышленности. - 2005. - № 2. - С. 31 - 34.

1. Данилова, Е.А. Построение информационно-измерительной системы

обнаружения дефектов проводящего рисунка печатных плат / Данилова Е.А. // Измерение. Мониторинг. Управление. Контроль. - 2016. - № 4 (18. - С. 69 - 78.)

1. Григорьев, А.В., Кочегаров, И.И., Юрков, Н.К. Автоматизированная

система для подсчета заужений проводящего рисунка печатной платы, обусловленных и не обусловленных наличием раковин / Григорьев А.В., Кочегаров И.И., Юрков Н.Н. //Надежность и качество сложных систем. — 2015. -№ 2 (10) - С. 27 - 33.

1. Данилова, Е.А., Кочегаров, И.И., Емашкина, Т.С. Построение

информационно-измерительной системы обнаружения дефектов печатных плат с осуществлением анализа причин дефектов / Е.А. Данилова, И.И. Кочегаров, Т.С. Емашкина // Современные информационные технологии. - 2016. - 24. - С.65

1. Атре Ш. Структурный подход к организации баз данных. / Пер. с англ.

A.A. Александрова, В.И. Будзко; Под ред. Будзко. М.: Финансы и статистика, 1983. - 317 с.

34. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. М., Финансы и статистика, 1989.

35. Борн Г. Форматы данных. Киев: Торгово-издательское бюро BHV, 1995.

36. И.Бусленко Н.П. Моделирование сложных систем. М.: Наука,1978.-340 с.

37. Васильев Г.П. Программное обеспечение неоднородных распределенных систем: Анализ и реализация. / Г.П. Васильев, В.Е.

38. Вескес А.Дж., Гандерлоу Майк. Access и SQL Server. Руководство разработчика: Пер. с англ. М.: Лори, 1997. - 362 с.

39. Глушков В.М. Введение в АСУ. Киев: Техника, 1974. - 319 с.

40. Коберн Алистер. Быстрая разработка программного обеспечения. Издательство: Лори, 2002 г. 336 с. ISBN 5-85582-182-Х. 2002 г.

41. С. Фалчер. Программирование на Microsoft Visual Studio .NET Серия: Фундаментальные знания Издательство: Русская Редакция, 2002 г. 800 стр. ISBN 5-7502-0224-0

42. Брайан Джонсон, Крэйг Скибо, Марк Янг Основы Microsoft Visual Studio.NET 2003 Inside Microsoft Visual Studio.NET Серия: Фундаментальные знания Издательство: Русская Редакция, 2003. 464 с. ISBN 0-7356-1874-7

43. JIunaeo В. В. Программная инженерия. Методологические основы. — Москва: Издательство «ТЕИС», 2006. — 609 с.

44. Gao Q., Li M., Vitanyi P.M.B. Applying MDL to learning best model granularity//Artificial Intelligence. 2000. V. 121. P. 1-29.

45. Solomonoff R.J. The Universal Distribution and Machine Learning // The Computer Journal. 2003. V. 46. P. 598-601.

46. Перегудов Ф.И., Тарасенко В.П. Введение в системный анализ. М: Высшая школа, 1989. 367 с.

47. Ту Дж., Гонсалес P. Принципы распознавания образов: М., Мир. 1978. 412 с.

48. Quinlan J.R. С4.5: Programs for Machine Learning: San Mateo, CA, Morgan Kaufmann. 1993

ПРИЛОЖЕНИЕ 1. Глоссарий (словарь терминов предметной области).

* + - 1. ГОСТ – государственный стандарт, одна из основных категорий стандартов в Российской Федерации.
      2. Фреймданные – измененные форматом данных, которые нужны для выполнения работы алгоритмов.
      3. Сетевой трафик — какой – либо объем информации, которая передается по компьютерной сети за определенный период времени.
      4. Точность – характеристика, которая показывает долю объектов, названных классификатором положительными и при этом действительно являющимися положительными
      5. Полнота – характеристика, которая показывает долю правильно помеченных положительных объектов среди всех объектов положительного класса.
      6. NSL – KDD – база данных, являющаяся модернизацией базы данных KDD Cup ’99.
      7. Кросс – валидация – метод, благодаря которому алгоритм будет способен работать на наборе данных, который не используется при обучении.
      8. Python – это высокоуровневый язык программирования, который используется во многих сферах IT, например, машинное обучение, нейронные сети, разработка web – приложений, приложений и т.д.
      9. Jupiter Notebook – инструмент для разработки и представления проектов в области наук о данных. Данный инструмент объединяет код и вывод в виде одного документа, который содержит текст, математические уравнения и изображения.
      10. Pandas – программная библиотека на языке Python, которая используется для обработки и анализа данных;
      11. NumPy – программная библиотека на языке Python, которая используется для операций с большими массивами данных и матриц, также в данной библиотеке имеются числовые наборы и наборы математических функций;
      12. Matploitlib – это основная библиотека, которая используется для построения графиков в Python.

# ПРИЛОЖЕНИЕ 2. ЛИСТИНГ ПРОГРАММЫ.

import pandas as pd

import numpy as np

import sys

import sklearn

import io

import random

train\_url = 'https://raw.githubusercontent.com/merteroglu/NSL-KDD-Network-Instrusion-Detection/master/NSL\_KDD\_Train.csv'

test\_url = 'https://raw.githubusercontent.com/merteroglu/NSL-KDD-Network-Instrusion-Detection/master/NSL\_KDD\_Test.csv'

col\_names = ["duration","protocol\_type","service","flag","src\_bytes",

"dst\_bytes","land","wrong\_fragment","urgent","hot","num\_failed\_logins",

"logged\_in","num\_compromised","root\_shell","su\_attempted","num\_root",

"num\_file\_creations","num\_shells","num\_access\_files","num\_outbound\_cmds",

"is\_host\_login","is\_guest\_login","count","srv\_count","serror\_rate",

"srv\_serror\_rate","rerror\_rate","srv\_rerror\_rate","same\_srv\_rate",

"diff\_srv\_rate","srv\_diff\_host\_rate","dst\_host\_count","dst\_host\_srv\_count",

"dst\_host\_same\_srv\_rate","dst\_host\_diff\_srv\_rate","dst\_host\_same\_src\_port\_rate",

"dst\_host\_srv\_diff\_host\_rate","dst\_host\_serror\_rate","dst\_host\_srv\_serror\_rate",

"dst\_host\_rerror\_rate","dst\_host\_srv\_rerror\_rate","label"]

df = pd.read\_csv(train\_url,header=None, names = col\_names)

df\_test = pd.read\_csv(test\_url, header=None, names = col\_names)

print('Размерность тренировочного набора:',df.shape)

print('Размерность тестового набора:',df\_test.shape)

df.head(5)

print('Метки Обучающий набор:')

print(df['label'].value\_counts())

print()

print('Метки Тестовый набор:')

print(df\_test['label'].value\_counts())

print('Обучающий набор:')

for col\_name in df.columns:

if df[col\_name].dtypes == 'object' :

unique\_cat = len(df[col\_name].unique())

print("Столбец '{col\_name}' имеет {unique\_cat} значений".format(col\_name=col\_name, unique\_cat=unique\_cat))

print()

print('Распределение значений в сервисах:')

print(df['service'].value\_counts().sort\_values(ascending=False).head())

print('Тестовый набор:')

for col\_name in df\_test.columns:

if df\_test[col\_name].dtypes == 'object' :

unique\_cat = len(df\_test[col\_name].unique())

print("Столбец '{col\_name}' имеет {unique\_cat} значений".format(col\_name=col\_name, unique\_cat=unique\_cat))

from sklearn.preprocessing import LabelEncoder,OneHotEncoder

categorical\_columns=['protocol\_type', 'service', 'flag']

df\_categorical\_values = df[categorical\_columns]

testdf\_categorical\_values = df\_test[categorical\_columns]

df\_categorical\_values.head()

# protocol type

unique\_protocol=sorted(df.protocol\_type.unique())

string1 = 'Protocol\_type\_'

unique\_protocol2=[string1 + x for x in unique\_protocol]

print(unique\_protocol2)

# service

unique\_service=sorted(df.service.unique())

string2 = 'service\_'

unique\_service2=[string2 + x for x in unique\_service]

print(unique\_service2)

# flag

unique\_flag=sorted(df.flag.unique())

string3 = 'flag\_'

unique\_flag2=[string3 + x for x in unique\_flag]

print(unique\_flag2)

# put together

dumcols=unique\_protocol2 + unique\_service2 + unique\_flag2

#do it for test set

unique\_service\_test=sorted(df\_test.service.unique())

unique\_service2\_test=[string2 + x for x in unique\_service\_test]

testdumcols=unique\_protocol2 + unique\_service2\_test + unique\_flag2

df\_categorical\_values\_enc=df\_categorical\_values.apply(LabelEncoder().fit\_transform)

print(df\_categorical\_values.head())

print('--------------------')

print(df\_categorical\_values\_enc.head())

# test set

testdf\_categorical\_values\_enc=testdf\_categorical\_values.apply(LabelEncoder().fit\_transform)

enc = OneHotEncoder(categories='auto')

df\_categorical\_values\_encenc = enc.fit\_transform(df\_categorical\_values\_enc)

df\_cat\_data = pd.DataFrame(df\_categorical\_values\_encenc.toarray(),columns=dumcols)

# test set

testdf\_categorical\_values\_encenc = enc.fit\_transform(testdf\_categorical\_values\_enc)

testdf\_cat\_data = pd.DataFrame(testdf\_categorical\_values\_encenc.toarray(),columns=testdumcols)

df\_cat\_data.head()

trainservice=df['service'].tolist()

testservice= df\_test['service'].tolist()

difference=list(set(trainservice) - set(testservice))

string = 'service\_'

difference=[string + x for x in difference]

difference

for col in difference:

testdf\_cat\_data[col] = 0

print(df\_cat\_data.shape)

print(testdf\_cat\_data.shape)

newdf=df.join(df\_cat\_data)

newdf.drop('flag', axis=1, inplace=True)

newdf.drop('protocol\_type', axis=1, inplace=True)

newdf.drop('service', axis=1, inplace=True)

# test data

newdf\_test=df\_test.join(testdf\_cat\_data)

newdf\_test.drop('flag', axis=1, inplace=True)

newdf\_test.drop('protocol\_type', axis=1, inplace=True)

newdf\_test.drop('service', axis=1, inplace=True)

print(newdf.shape)

print(newdf\_test.shape)

labeldf=newdf['label']

labeldf\_test=newdf\_test['label']

# изменение столбцов

newlabeldf=labeldf.replace({ 'normal' : 0, 'neptune' : 1 ,'back': 1, 'land': 1, 'pod': 1, 'smurf': 1, 'teardrop': 1,'mailbomb': 1, 'apache2': 1, 'processtable': 1, 'udpstorm': 1, 'worm': 1,

'ipsweep' : 2,'nmap' : 2,'portsweep' : 2,'satan' : 2,'mscan' : 2,'saint' : 2

,'ftp\_write': 3,'guess\_passwd': 3,'imap': 3,'multihop': 3,'phf': 3,'spy': 3,'warezclient': 3,'warezmaster': 3,'sendmail': 3,'named': 3,'snmpgetattack': 3,'snmpguess': 3,'xlock': 3,'xsnoop': 3,'httptunnel': 3,

'buffer\_overflow': 4,'loadmodule': 4,'perl': 4,'rootkit': 4,'ps': 4,'sqlattack': 4,'xterm': 4})

newlabeldf\_test=labeldf\_test.replace({ 'normal' : 0, 'neptune' : 1 ,'back': 1, 'land': 1, 'pod': 1, 'smurf': 1, 'teardrop': 1,'mailbomb': 1, 'apache2': 1, 'processtable': 1, 'udpstorm': 1, 'worm': 1,

'ipsweep' : 2,'nmap' : 2,'portsweep' : 2,'satan' : 2,'mscan' : 2,'saint' : 2

,'ftp\_write': 3,'guess\_passwd': 3,'imap': 3,'multihop': 3,'phf': 3,'spy': 3,'warezclient': 3,'warezmaster': 3,'sendmail': 3,'named': 3,'snmpgetattack': 3,'snmpguess': 3,'xlock': 3,'xsnoop': 3,'httptunnel': 3,

'buffer\_overflow': 4,'loadmodule': 4,'perl': 4,'rootkit': 4,'ps': 4,'sqlattack': 4,'xterm': 4})

newdf['label'] = newlabeldf

newdf\_test['label'] = newlabeldf\_test

to\_drop\_DoS = [0,1]

to\_drop\_Probe = [0,2]

to\_drop\_R2L = [0,3]

to\_drop\_U2R = [0,4]

# Фильтрация значений

DoS\_df=newdf[newdf['label'].isin(to\_drop\_DoS)];

Probe\_df=newdf[newdf['label'].isin(to\_drop\_Probe)];

R2L\_df=newdf[newdf['label'].isin(to\_drop\_R2L)];

U2R\_df=newdf[newdf['label'].isin(to\_drop\_U2R)];

#test

DoS\_df\_test=newdf\_test[newdf\_test['label'].isin(to\_drop\_DoS)];

Probe\_df\_test=newdf\_test[newdf\_test['label'].isin(to\_drop\_Probe)];

R2L\_df\_test=newdf\_test[newdf\_test['label'].isin(to\_drop\_R2L)];

U2R\_df\_test=newdf\_test[newdf\_test['label'].isin(to\_drop\_U2R)];

print('Обучение:')

print('Размерность DoS:' ,DoS\_df.shape)

print('Размерность Probe:' ,Probe\_df.shape)

print('Размерность R2L:' ,R2L\_df.shape)

print('Размерность U2R:' ,U2R\_df.shape)

print()

print('Тест:')

print('Размерность DoS:' ,DoS\_df\_test.shape)

print('Размерность Probe:' ,Probe\_df\_test.shape)

print('Размерность R2L:' ,R2L\_df\_test.shape)

print('Размерность U2R:' ,U2R\_df\_test.shape)

X\_DoS = DoS\_df.drop('label',1)

Y\_DoS = DoS\_df.label

X\_Probe = Probe\_df.drop('label',1)

Y\_Probe = Probe\_df.label

X\_R2L = R2L\_df.drop('label',1)

Y\_R2L = R2L\_df.label

X\_U2R = U2R\_df.drop('label',1)

Y\_U2R = U2R\_df.label

# Тестовая выборка

X\_DoS\_test = DoS\_df\_test.drop('label',1)

Y\_DoS\_test = DoS\_df\_test.label

X\_Probe\_test = Probe\_df\_test.drop('label',1)

Y\_Probe\_test = Probe\_df\_test.label

X\_R2L\_test = R2L\_df\_test.drop('label',1)

Y\_R2L\_test = R2L\_df\_test.label

X\_U2R\_test = U2R\_df\_test.drop('label',1)

Y\_U2R\_test = U2R\_df\_test.label

colNames=list(X\_DoS)

colNames\_test=list(X\_DoS\_test)

from sklearn import preprocessing

scaler1 = preprocessing.StandardScaler().fit(X\_DoS)

X\_DoS=scaler1.transform(X\_DoS)

scaler2 = preprocessing.StandardScaler().fit(X\_Probe)

X\_Probe=scaler2.transform(X\_Probe)

scaler3 = preprocessing.StandardScaler().fit(X\_R2L)

X\_R2L=scaler3.transform(X\_R2L)

scaler4 = preprocessing.StandardScaler().fit(X\_U2R)

X\_U2R=scaler4.transform(X\_U2R)

# test data

scaler5 = preprocessing.StandardScaler().fit(X\_DoS\_test)

X\_DoS\_test=scaler5.transform(X\_DoS\_test)

scaler6 = preprocessing.StandardScaler().fit(X\_Probe\_test)

X\_Probe\_test=scaler6.transform(X\_Probe\_test)

scaler7 = preprocessing.StandardScaler().fit(X\_R2L\_test)

X\_R2L\_test=scaler7.transform(X\_R2L\_test)

scaler8 = preprocessing.StandardScaler().fit(X\_U2R\_test)

X\_U2R\_test=scaler8.transform(X\_U2R\_test)

from sklearn.feature\_selection import RFE

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n\_estimators=10,n\_jobs=2)

rfe = RFE(estimator=clf, n\_features\_to\_select=13, step=1)

rfe.fit(X\_DoS, Y\_DoS.astype(int))

X\_rfeDoS=rfe.transform(X\_DoS)

true=rfe.support\_

rfecolindex\_DoS=[i for i, x in enumerate(true) if x]

rfecolname\_DoS=list(colNames[i] for i in rfecolindex\_DoS)

rfe.fit(X\_Probe, Y\_Probe.astype(int))

X\_rfeProbe=rfe.transform(X\_Probe)

true=rfe.support\_

rfecolindex\_Probe=[i for i, x in enumerate(true) if x]

rfecolname\_Probe=list(colNames[i] for i in rfecolindex\_Probe)

rfe.fit(X\_R2L, Y\_R2L.astype(int))

X\_rfeR2L=rfe.transform(X\_R2L)

true=rfe.support\_

rfecolindex\_R2L=[i for i, x in enumerate(true) if x]

rfecolname\_R2L=list(colNames[i] for i in rfecolindex\_R2L)

rfe.fit(X\_U2R, Y\_U2R.astype(int))

X\_rfeU2R=rfe.transform(X\_U2R)

true=rfe.support\_

rfecolindex\_U2R=[i for i, x in enumerate(true) if x]

rfecolname\_U2R=list(colNames[i] for i in rfecolindex\_U2R)

print('Выбранные функции для DoS:',rfecolname\_DoS)

print()

print('Выбранные функции для Probe:',rfecolname\_Probe)

print()

print('Выбранные функции для R2L:',rfecolname\_R2L)

print()

print('Выбранные функции для U2R:',rfecolname\_U2R)

# все функции

clf\_DoS=RandomForestClassifier(n\_estimators=10,n\_jobs=2)

clf\_Probe=RandomForestClassifier(n\_estimators=10,n\_jobs=2)

clf\_R2L=RandomForestClassifier(n\_estimators=10,n\_jobs=2)

clf\_U2R=RandomForestClassifier(n\_estimators=10,n\_jobs=2)

clf\_DoS.fit(X\_DoS, Y\_DoS.astype(int))

clf\_Probe.fit(X\_Probe, Y\_Probe.astype(int))

clf\_R2L.fit(X\_R2L, Y\_R2L.astype(int))

clf\_U2R.fit(X\_U2R, Y\_U2R.astype(int))

clf\_rfeDoS=RandomForestClassifier(n\_estimators=10,n\_jobs=2)

clf\_rfeProbe=RandomForestClassifier(n\_estimators=10,n\_jobs=2)

clf\_rfeR2L=RandomForestClassifier(n\_estimators=10,n\_jobs=2)

clf\_rfeU2R=RandomForestClassifier(n\_estimators=10,n\_jobs=2)

clf\_rfeDoS.fit(X\_rfeDoS, Y\_DoS.astype(int))

clf\_rfeProbe.fit(X\_rfeProbe, Y\_Probe.astype(int))

clf\_rfeR2L.fit(X\_rfeR2L, Y\_R2L.astype(int))

clf\_rfeU2R.fit(X\_rfeU2R, Y\_U2R.astype(int))

clf\_DoS.predict(X\_DoS\_test)

clf\_DoS.predict\_proba(X\_DoS\_test)[0:10]

Y\_DoS\_pred=clf\_DoS.predict(X\_DoS\_test)

# Создание матрицы неточностей

pd.crosstab(Y\_DoS\_test, Y\_DoS\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_Probe\_pred=clf\_Probe.predict(X\_Probe\_test)

# Создание матрицы неточностей

pd.crosstab(Y\_Probe\_test, Y\_Probe\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_R2L\_pred=clf\_R2L.predict(X\_R2L\_test)

# Создание матрицы неточностей

pd.crosstab(Y\_R2L\_test, Y\_R2L\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_U2R\_pred=clf\_U2R.predict(X\_U2R\_test)

# Создание матрицы неточностей

pd.crosstab(Y\_U2R\_test, Y\_U2R\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

from sklearn.model\_selection import cross\_val\_score

from sklearn import metrics

accuracy = cross\_val\_score(clf\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='precision')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='recall')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='f1')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_DoS.predict(X\_DoS\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_DoS\_test, prediction, pos\_label = 1)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_Probe.predict(X\_Probe\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_Probe\_test, prediction, pos\_label=2)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_U2R.predict(X\_U2R\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_U2R\_test, prediction, pos\_label=4)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_R2L.predict(X\_R2L\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_R2L\_test, prediction, pos\_label = 3)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

X\_DoS\_test2=X\_DoS\_test[:,rfecolindex\_DoS]

X\_Probe\_test2=X\_Probe\_test[:,rfecolindex\_Probe]

X\_R2L\_test2=X\_R2L\_test[:,rfecolindex\_R2L]

X\_U2R\_test2=X\_U2R\_test[:,rfecolindex\_U2R]

X\_U2R\_test2.shape

Y\_DoS\_pred2=clf\_rfeDoS.predict(X\_DoS\_test2)

# Создание матрицы неточностей

pd.crosstab(Y\_DoS\_test, Y\_DoS\_pred2, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_Probe\_pred2=clf\_rfeProbe.predict(X\_Probe\_test2)

# Создание матрицы неточностей

pd.crosstab(Y\_Probe\_test, Y\_Probe\_pred2, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_R2L\_pred2=clf\_rfeR2L.predict(X\_R2L\_test2)

# Создание матрицы неточностей

pd.crosstab(Y\_R2L\_test, Y\_R2L\_pred2, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_U2R\_pred2=clf\_rfeU2R.predict(X\_U2R\_test2)

# Создание матрицы неточностей

pd.crosstab(Y\_U2R\_test, Y\_U2R\_pred2, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

accuracy = cross\_val\_score(clf\_rfeDoS, X\_DoS\_test2, Y\_DoS\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_rfeDoS, X\_DoS\_test2, Y\_DoS\_test, cv=10, scoring='precision')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_rfeDoS, X\_DoS\_test2, Y\_DoS\_test, cv=10, scoring='recall')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_rfeDoS, X\_DoS\_test2, Y\_DoS\_test, cv=10, scoring='f1')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

prediction = clf\_rfeDoS.predict(X\_DoS\_test2)

fpr, tpr, threshold = metrics.roc\_curve(Y\_DoS\_test, prediction, pos\_label = 1)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_rfeProbe, X\_Probe\_test2, Y\_Probe\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_rfeProbe, X\_Probe\_test2, Y\_Probe\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_rfeProbe, X\_Probe\_test2, Y\_Probe\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_rfeProbe, X\_Probe\_test2, Y\_Probe\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_rfeProbe.predict(X\_Probe\_test2)

fpr, tpr, threshold = metrics.roc\_curve(Y\_Probe\_test, prediction, pos\_label = 2)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_rfeR2L, X\_R2L\_test2, Y\_R2L\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_rfeR2L, X\_R2L\_test2, Y\_R2L\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_rfeR2L, X\_R2L\_test2, Y\_R2L\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_rfeR2L, X\_R2L\_test2, Y\_R2L\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_rfeR2L.predict(X\_R2L\_test2)

fpr, tpr, threshold = metrics.roc\_curve(Y\_R2L\_test, prediction, pos\_label = 3)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_rfeU2R, X\_U2R\_test2, Y\_U2R\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_rfeU2R, X\_U2R\_test2, Y\_U2R\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_rfeU2R, X\_U2R\_test2, Y\_U2R\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_rfeU2R, X\_U2R\_test2, Y\_U2R\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

prediction = clf\_rfeU2R.predict(X\_U2R\_test2)

fpr, tpr, threshold = metrics.roc\_curve(Y\_U2R\_test, prediction, pos\_label = 4)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

from sklearn.neighbors import KNeighborsClassifier

clf\_KNN\_DoS=KNeighborsClassifier()

clf\_KNN\_Probe=KNeighborsClassifier()

clf\_KNN\_R2L=KNeighborsClassifier()

clf\_KNN\_U2R=KNeighborsClassifier()

clf\_KNN\_DoS.fit(X\_DoS, Y\_DoS.astype(int))

clf\_KNN\_Probe.fit(X\_Probe, Y\_Probe.astype(int))

clf\_KNN\_R2L.fit(X\_R2L, Y\_R2L.astype(int))

clf\_KNN\_U2R.fit(X\_U2R, Y\_U2R.astype(int))

Y\_DoS\_pred=clf\_KNN\_DoS.predict(X\_DoS\_test)

pd.crosstab(Y\_DoS\_test, Y\_DoS\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_Probe\_pred=clf\_KNN\_Probe.predict(X\_Probe\_test)

pd.crosstab(Y\_Probe\_test, Y\_Probe\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_R2L\_pred=clf\_KNN\_R2L.predict(X\_R2L\_test)

pd.crosstab(Y\_R2L\_test, Y\_R2L\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_U2R\_pred=clf\_KNN\_U2R.predict(X\_U2R\_test)

pd.crosstab(Y\_U2R\_test, Y\_U2R\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

from sklearn.model\_selection import cross\_val\_score

from sklearn import metrics

accuracy = cross\_val\_score(clf\_KNN\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_KNN\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='precision')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_KNN\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='recall')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_KNN\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='f1')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

prediction = clf\_KNN\_DoS.predict(X\_DoS\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_DoS\_test, prediction, pos\_label = 1)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_KNN\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_KNN\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_KNN\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_KNN\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_KNN\_Probe.predict(X\_Probe\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_Probe\_test, prediction, pos\_label = 2)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_KNN\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_KNN\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_KNN\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_KNN\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_KNN\_R2L.predict(X\_R2L\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_R2L\_test, prediction, pos\_label = 3)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_KNN\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_KNN\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_KNN\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_KNN\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_KNN\_U2R.predict(X\_U2R\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_U2R\_test, prediction, pos\_label = 4)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

from sklearn.svm import SVC

clf\_SVM\_DoS=SVC(kernel='linear', C=1.0, random\_state=0)

clf\_SVM\_Probe=SVC(kernel='linear', C=1.0, random\_state=0)

clf\_SVM\_R2L=SVC(kernel='linear', C=1.0, random\_state=0)

clf\_SVM\_U2R=SVC(kernel='linear', C=1.0, random\_state=0)

clf\_SVM\_DoS.fit(X\_DoS, Y\_DoS.astype(int))

clf\_SVM\_Probe.fit(X\_Probe, Y\_Probe.astype(int))

clf\_SVM\_R2L.fit(X\_R2L, Y\_R2L.astype(int))

clf\_SVM\_U2R.fit(X\_U2R, Y\_U2R.astype(int))

Y\_DoS\_pred=clf\_SVM\_DoS.predict(X\_DoS\_test)

pd.crosstab(Y\_DoS\_test, Y\_DoS\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_Probe\_pred=clf\_SVM\_Probe.predict(X\_Probe\_test)

pd.crosstab(Y\_Probe\_test, Y\_Probe\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_R2L\_pred=clf\_SVM\_R2L.predict(X\_R2L\_test)

pd.crosstab(Y\_R2L\_test, Y\_R2L\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_U2R\_pred=clf\_SVM\_U2R.predict(X\_U2R\_test)

pd.crosstab(Y\_U2R\_test, Y\_U2R\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

from sklearn.model\_selection import cross\_val\_score

from sklearn import metrics

accuracy = cross\_val\_score(clf\_SVM\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_SVM\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='precision')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_SVM\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='recall')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_SVM\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='f1')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

prediction = clf\_SVM\_DoS.predict(X\_DoS\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_DoS\_test, prediction, pos\_label = 1)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_SVM\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_SVM\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_SVM\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_SVM\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_SVM\_Probe.predict(X\_Probe\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_Probe\_test, prediction, pos\_label = 2)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_SVM\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_SVM\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_SVM\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_SVM\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_SVM\_R2L.predict(X\_R2L\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_R2L\_test, prediction, pos\_label = 3)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_SVM\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_SVM\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_SVM\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_SVM\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_SVM\_U2R.predict(X\_U2R\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_U2R\_test, prediction, pos\_label = 4)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

from sklearn.ensemble import VotingClassifier

clf\_voting\_DoS = VotingClassifier(estimators=[('rf', clf\_DoS), ('knn', clf\_KNN\_DoS), ('svm', clf\_SVM\_DoS)], voting='hard')

clf\_voting\_Probe = VotingClassifier(estimators=[('rf', clf\_Probe), ('knn', clf\_KNN\_Probe), ('svm', clf\_SVM\_Probe)], voting='hard')

clf\_voting\_R2L = VotingClassifier(estimators=[('rf', clf\_R2L), ('knn', clf\_KNN\_R2L), ('svm', clf\_SVM\_R2L)], voting='hard')

clf\_voting\_U2R = VotingClassifier(estimators=[('rf', clf\_U2R), ('knn', clf\_KNN\_U2R), ('svm', clf\_SVM\_U2R)], voting='hard')

clf\_voting\_DoS.fit(X\_DoS, Y\_DoS.astype(int))

clf\_voting\_Probe.fit(X\_Probe, Y\_Probe.astype(int))

clf\_voting\_R2L.fit(X\_R2L, Y\_R2L.astype(int))

clf\_voting\_U2R.fit(X\_U2R, Y\_U2R.astype(int))

Y\_DoS\_pred=clf\_voting\_DoS.predict(X\_DoS\_test)

pd.crosstab(Y\_DoS\_test, Y\_DoS\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_Probe\_pred=clf\_voting\_Probe.predict(X\_Probe\_test)

pd.crosstab(Y\_Probe\_test, Y\_Probe\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_R2L\_pred=clf\_voting\_R2L.predict(X\_R2L\_test)

pd.crosstab(Y\_R2L\_test, Y\_R2L\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

Y\_U2R\_pred=clf\_voting\_U2R.predict(X\_U2R\_test)

pd.crosstab(Y\_U2R\_test, Y\_U2R\_pred, rownames=['Актуальные атаки'], colnames=['Прогнозируемые атаки'])

from sklearn.model\_selection import cross\_val\_score

from sklearn import metrics

accuracy = cross\_val\_score(clf\_voting\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_voting\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='precision')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_voting\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='recall')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_voting\_DoS, X\_DoS\_test, Y\_DoS\_test, cv=10, scoring='f1')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_voting\_DoS.predict(X\_DoS\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_DoS\_test, prediction, pos\_label = 1)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_voting\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_voting\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_voting\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_voting\_Probe, X\_Probe\_test, Y\_Probe\_test, cv=10, scoring='f1\_macro')

print("F-mesaure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_voting\_Probe.predict(X\_Probe\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_Probe\_test, prediction, pos\_label = 2)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_voting\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_voting\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_voting\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_voting\_R2L, X\_R2L\_test, Y\_R2L\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

prediction = clf\_voting\_R2L.predict(X\_R2L\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_R2L\_test, prediction, pos\_label = 3)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

accuracy = cross\_val\_score(clf\_voting\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='accuracy')

print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() \* 2))

precision = cross\_val\_score(clf\_voting\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='precision\_macro')

print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() \* 2))

recall = cross\_val\_score(clf\_voting\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='recall\_macro')

print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() \* 2))

f = cross\_val\_score(clf\_voting\_U2R, X\_U2R\_test, Y\_U2R\_test, cv=10, scoring='f1\_macro')

print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() \* 2))

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc

from sklearn import metrics

prediction = clf\_voting\_U2R.predict(X\_U2R\_test)

fpr, tpr, threshold = metrics.roc\_curve(Y\_U2R\_test, prediction, pos\_label = 4)

roc\_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt

plt.title('ROC - curve')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()