ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

Кафедра «Криптология и кибербезопасность»

Отчет

о практике

«Метод непрерывной аутентификации пользователей мобильных устройств на основе анализа нескольких поведенческих характеристик»

Исполнитель:
студент гр. Б17-505 Казьмин С.К.
подпись, дата

Научный руководитель: Когос К.Г.
подпись, дата

Зам. зав. каф. № 42: Когос К.Г.
подпись, дата

Москва – 2021

РЕФЕРАТ

Отчёт 81 с., 18 рис., 6 табл., 76 источн., 5 прил.

АУТЕНТИФИКАЦИЯ, ПОВЕДЕНЧЕСКАЯ БИОМЕТРИЯ, НЕПРЕРЫВНАЯ АУТЕНТИФИКАЦИЯ, МОБИЛЬНОЕ УСТРОЙСТВО, ВЗАИМОДЕЙСТВИЕ, ПРИЛОЖЕНИЯ, ПРОФИЛЬ, МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ, АНАЛИЗ ПОВЕДЕНИЯ

Объектом исследования является непрерывная аутентификация пользователей мобильных устройств по поведенческой биометрии.

Цель работы — исследовать и оценить эффективность метода непрерывной аутентификации пользователей мобильных устройств на основе анализа нескольких поведенческих характеристик.

В работе рассмотрены различные подходы к аутентификации и предложен метод непрерывной аутентификации на основе нескольких поведенческих характеристик; разработано мобильное приложение под операционную систему Android для составления поведенческого профиля пользователей. С помощью разработанного приложения проведён сбор данных у 8 добровольцев.

На основе собранных данных проведено формирование признаков для тестирования алгоритмов машинного обучения в задаче аутентификации; проведено тестирование алгоритмов на полученных выборках.

Были получены значения метрик качества для нескольких алгоритмов классификации. Лучшие значения показал алгоритм случайного леса, для него значение EER изменялось в пределах от 0.157 до 0.029. Также было проведено сравнение метрик, полученных для разных модулей при разных способах формирования признаков, и выбран способ для дальнейшей оценки качества работы предложенного метода аутентификации.

Предложено использовать алгоритм логистической регрессии для комбинирования вердиктов, полученных на основе анализа нескольких поведенческих характеристик.

По итогам тестирования алгоритмов машинного обучения выбран алгоритм случайного леса для дальнейшего изучения предложенного метода аутентификации. Разработана схема тестирования метода, основанная на формировании потока событий, имитирующего вторжение злоумышленника в защищаемую систему.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящем отчете применяются следующие термины с соответствующими определениями, обозначениями и сокращениями.

ANGA — среднее число подлинных действий (average number of genuine actions).

ANIA — среднее число действий нарушителя (average number of impostor actions).

API — программный интерфейс приложения (application programming interface).

AUC-ROC — площадь под ROC-кривой.

CSV — формат файла, состоящего из значений, разделённых запятыми (comma-separated values).

EER — уровень ошибки, при которой равны FAR и FRR (equal error rate).

FAR — уровень ошибок второго рода (false acceptance rate).

FPR — доля ложноположительных предсказаний модели (false positive rate).

FRR — уровень ошибок первого рода (false rejection rate).

ROC — кривая рабочей характеристики приёмника (receiver operating characteristic).

TPR — доля истинно-положительных предсказаний модели (true positive rate).

ВВЕДЕНИЕ

В 2020 году смартфоны плотно интегрированы в повседневную жизнь человека. Мобильные гаджеты используются как для повседневного общения, так и для деловых коммуникаций, в которых фигурирует конфиденциальная информация. В мобильных телефонах собираются и обрабатываются личные переписки и фотографии, приватные сведения о пользователе и другие данные, утечка которых в свободный доступ нежелательна. Поэтому важно снижать вероятность несанкционированного доступа к информации на устройстве. Из этого следует, что аутентификация — ключевой элемент в процессе обеспечения информационной безопасности пользователей мобильных устройств.

Аутентификация пользователей смартфонов осуществляется с помощью нескольких методик. При проведении аутентификации используются пароли, USB-токены, смарт-карты, биометрические характеристики. За последнее время получила распространение аутентификация на основе сканирования отпечатка пальца, набирает популярность технология распознавания лица пользователя. Кроме того, особняком стоят методы, которые используют поведенческую биометрию. Для распознавания пользователя используются закономерности в его поведении, например, его местоположение, скорость перемещения, запущенные приложения на устройстве, манера ходьбы.

Аутентификация на основе поведения несёт ряд преимуществ. Непрерывное распознавание пользователя можно проводить без ущерба для удобства использования гаджета. В качестве поведенческих характеристик используются параметры и события, которые регистрирует мобильное устройство. Их необходимо выбирать, исходя из того, какой уровень защищённости необходим и какие сценарии использования устройства предполагаются.

В представленной работе проводилось исследование метода непрерывной аутентификации на основе поведенческих характеристик пользователя мобильного устройства.

В первом разделе рассмотрены актуальные способы аутентификации, проведено их сравнение и акцентировано внимание на достоинствах непрерывной аутентификации по поведенческой биометрии. Отмечены недостатки классической биометрии по физическим характеристикам человека.

Во втором разделе проанализированы ошибки, возникающие в системах аутентификации и метрики, использующиеся для их описания. Проведён обзор актуальных и распространённых методов поведенческой биометрии, таких как аутентификация по клавиатурному почерку, по походке, по поведенческому профилю. Проведено сравнение таких подходов.

В третьем разделе предложен метод непрерывной аутентификации на основе биометрического профиля пользователя, основанного на данных о сетях WiFi и устройствах Bluetooth, находящихся вокруг устройства и на данных о геопозиции пользователя.

В четвёртом разделе приведено описание разработанного для сбора данных поведения пользователей приложения под операционную систему Android.

В пятом разделе описан процесс обработки собранных данных и формирования признаков для последующего обучения моделей машинного обучения для различных модулей системы.

В шестом разделе описана схема и приведены результаты тестирования нескольких алгоритмов машинного обучения, использованных для решения задачи классификации пользователей на легальных и несанкционированных.

В седьмом разделе описан способ тестирования предложенного метода аутентификации на основе формирования выборок, симулирующих проникновение нарушителя в защищаемую систему. Также представлен способ смешивания вердиктов нескольких классификаторов.

1 Анализ актуальных методов аутентификации пользователей

Для защиты мобильных устройств от несанкционированного доступа используются идентификация и аутентификация. Идентификация представляет собой механизм сопоставления идентификатора пользователя с уже зафиксированными в базе устройства. Аутентификация же предполагает проведение процедуры подтверждения пользователем того, что он является тем самым субъектом, идентификатор которого использует [1]. Как правило, для проведения аутентификации необходимо, чтобы пользователь предоставил данные, которыми может обладать исключительно он в силу тех или иных причин.

1.1 Факторы аутентификации

Всего рассматривают три типа факторов аутентификации [1], [2], [3]:

- пользователь что-то знает (например, пароль, кодовое слово);
- пользователь чем-то обладает (USB-токен, смарт-карта);
- на основе биометрических характеристик пользователя (отпечаток пальца, радужная оболочка глаза, голос).

Биометрия в свою очередь может быть двух типов:

- биологическая (используются физические особенности человека);
- поведенческая (распознавание человека проводится при помощи привычек пользователя, характерных особенностей использования мобильного устройства).

Аутентификация бывает однофакторной и многофакторной [1]. Однофакторная аутентификация реализуется на основе одного фактора аутентификации, например, пользователь для входа в систему сканирует лицо. При многофакторной аутентификации используются несколько факторов. К примеру, сначала пользователь сканирует отпечаток пальца, после чего вводит пин-код. Стоит отметить, что многофакторная аутентификация обеспечивает более высокий уровень безопасности [4].

На рисунке 1 изображена схема процесса биометрической аутентификации пользователя мобильного устройства. При первичной регистрации в системе у пользователя запрашиваются аутентификационные сведения, на основе которых строится эталонная модель пользователя. При следующей попытке входа у пользователя снова запрашиваются данные, на основе которых строится новая модель, которая сравнивается с эталонной. На основе сравнения принимается решение о выдаче доступа пользователю или блокировке устройства.

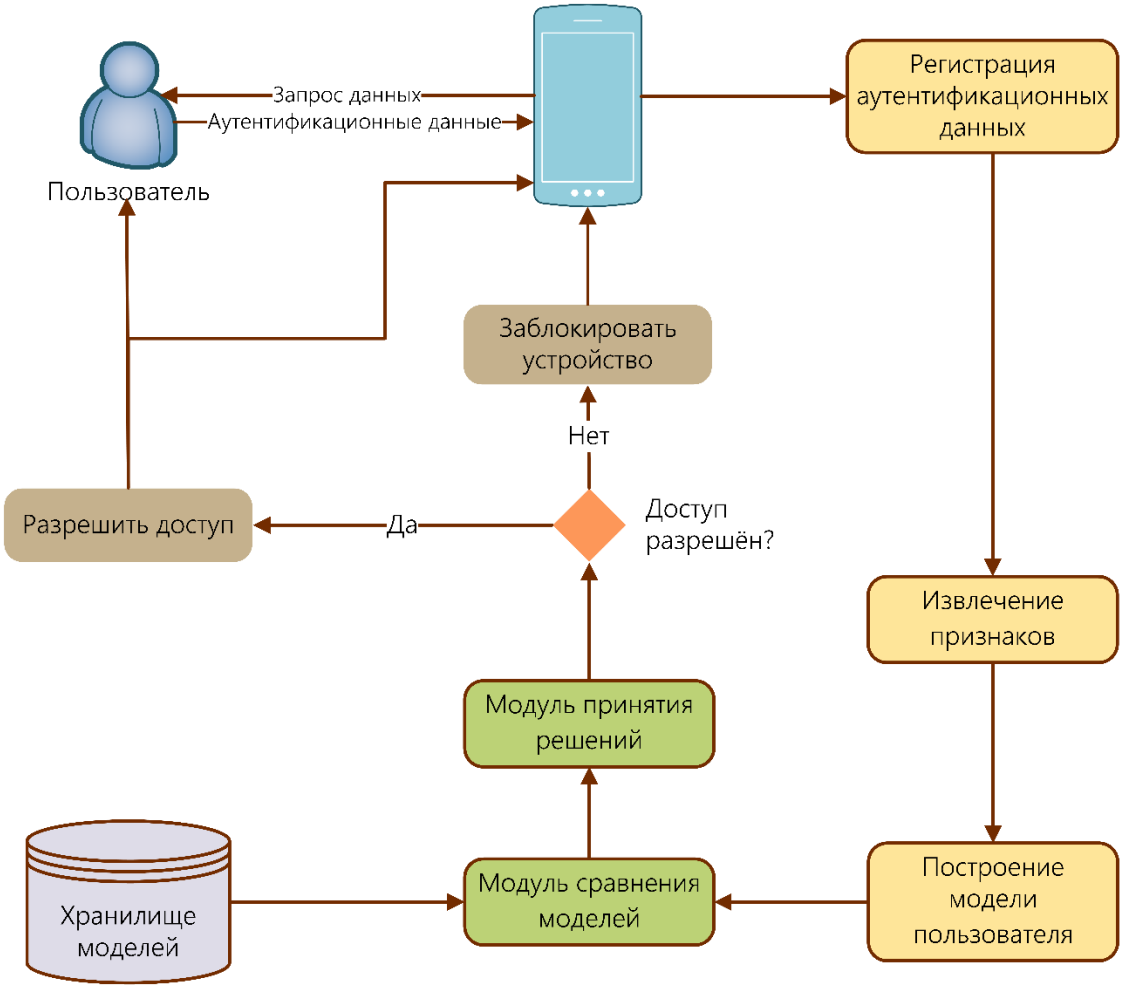

Рисунок 1 — Схема проведения биометрической аутентификации пользователя мобильного устройства

1.2 Поведенческая биометрия

Отдельный интерес представляет поведенческая биометрия — автоматизированный процесс распознавания личности по особенностям поведения [5]. Особенность поведенческих биометрических характеристик

заключается в их протяжённости во времени [1]. Фиксируемые устройством действия имеют начало, середину и конец. Некоторые параметры и вовсе непрерывны. Поведенческая биометрия основана на активности человека и паттернах его поведения [5]. Походка, клавиатурный почерк, голос, местоположение, динамика движения пальца по экрану — всё это используется для распознавания человека [6]. Данный способ аутентификации является нестабильным во времени. Это связано с распорядком дня человека в течение суток и с изменениями в привычках человека. Поэтому необходимо обновлять данные о пользователе. Достоинства поведенческой биометрии — рентабельность и незаметность для пользователя, что добавляет положительных впечатлений от повседневного использования аппарата.

Преимущество поведенческой биометрии по сравнению с классической физиологической биометрией заключается в том, что подделать данные о поведении человека гораздо сложнее, чем биометрические характеристики [7]. Например, существуют способы обходить дактилоскопическую защиту в смартфонах [8], [9]. Maro и Kovalchuk в своей работе [9] исследовали возможность обхода системы биометрической защиты смартфонов и других портативных устройств с помощью желатинового слепка отпечатка пальца. Для разных устройств в 44-75 процентах случаев попытки обойти защиту увенчались успехом. Таким образом, парольная защита с длинным паролем может быть более надёжна [9]. Ввиду этого, системы аутентификации, обладающие более высокой устойчивостью к взлому, кажутся хорошей альтернативной для внедрения в смартфоны.

Для сбора сведений о поведении пользователя не требуется специальных программных и аппаратных решений, так как у большинства мобильных устройств присутствуют необходимые датчики и сенсоры, а операционная система предоставляет к ним доступ стандартными средствами. Сбор сведений можно осуществлять с помощью следующих встроенных датчиков и модулей: GPS, микрофон, датчик освещённости, акселерометр, магнитометр, гироскоп,

камера, сенсорный экран, Bluetooth, WiFi и другие. При этом со стороны пользователя не требуется дополнительных действий, информация собирается автономно.

1.3 Непрерывная аутентификация

В большинстве компьютерных устройств используется одноразовая аутентификация [2]. Пользователь вводит пароль или сканирует отпечаток пальца, после чего ему предоставляется доступ к ресурсам системы до тех пор, пока сессия использования не будет завершена. Это приемлемо для устройств, которым не требуется высокий уровень безопасности. Но такой недостаток приводит к повышению риска несанкционированного доступа к ресурсам системы. К примеру, при дистанционной сдаче экзамена пользователь будет использовать систему длительное время, в течение которого к устройству может быть получен доступ другим человеком.

Решить описанную проблему помогают методы непрерывной аутентификации. При непрерывной аутентификации устройство постоянно отслеживает характеристики пользователя и аутентифицирует по одному или нескольким факторам. Простой способ реализовать такой подход — запрашивать через дискретные интервалы времени у пользователя сведения, подтверждающие личность. Однако, решить проблему в полной мере таким образом не удастся, ведь в промежутки времени между аутентификацией возрастает вероятность получения доступа к информации злоумышленником. На рисунке 2 изображена схема процесса непрерывной аутентификации. При реализации системы непрерывной аутентификации необходимо как можно меньшую продолжительность этапа ожидания новых аутентификационных данных, так как большой промежуток времени является уязвимостью.

Поведенческая биометрия позволяет добиться существенного уменьшения интервала времени между двумя проверками подлинности пользователя. Используя встроенные датчики и сенсоры, можно непрерывно отслеживать действия пользователя и постоянно (через малые промежутки времени) сверять

их с моделью поведения, необходимой для распознавания. После успешного входа в систему устройство непрерывно отслеживает действия человека. На их основе строится поведенческая модель, которая сравнивается с эталонной. Если в какой-то момент времени деятельность пользователя меняется и становится подозрительной, устройство принимает решение отказать в доступе и блокируется.

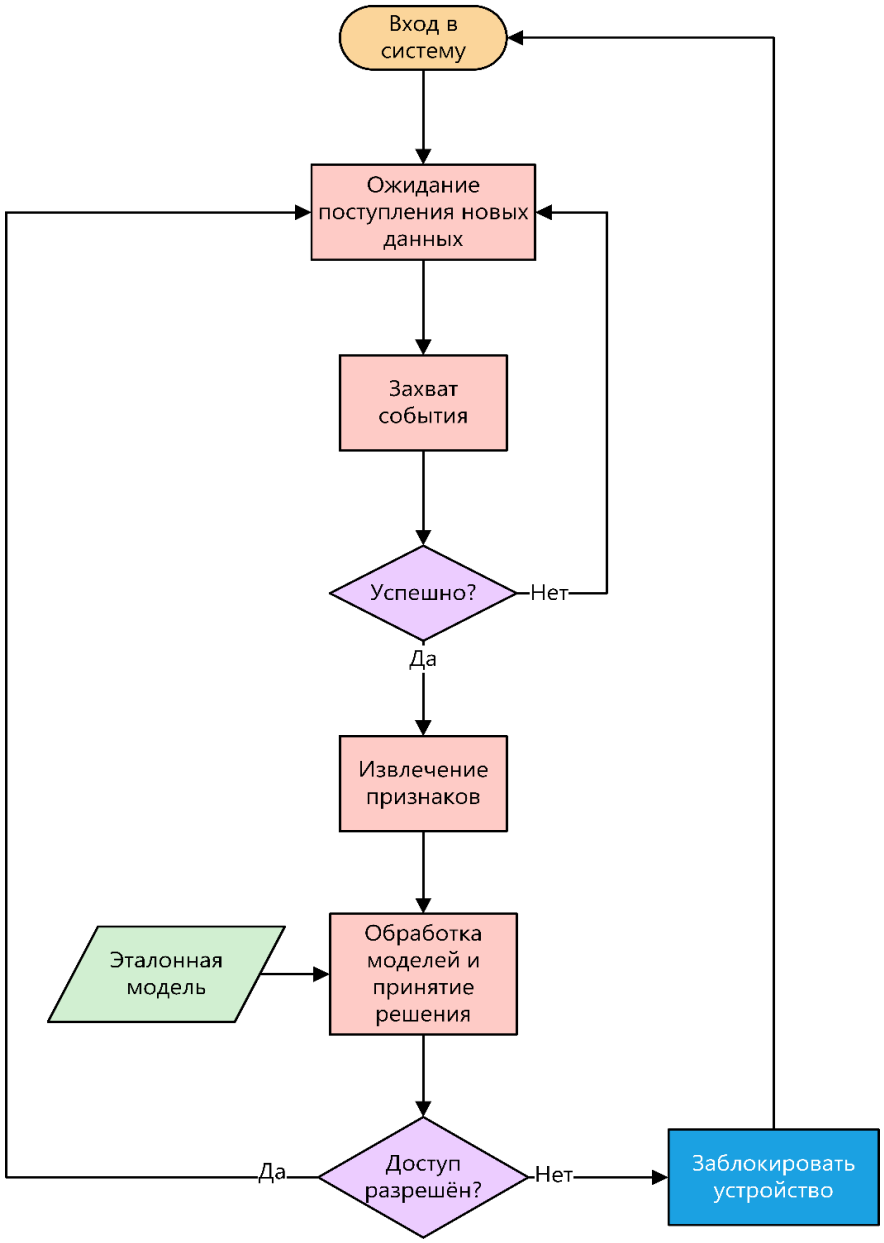

Рисунок 2 — Схема процесса непрерывной аутентификации по поведенческой биометрии на устройстве

Согласно различным исследованиями [10] от 33 до 57 процентов пользователей не блокируют свои телефоны. Следовательно, обеспечивать

безопасность таких пользователей можно именно с помощью непрерывной аутентификации по поведенческой биометрии. Такое решение не повлияет на их привычный опыт взаимодействия со своим устройством.

1.4 Сравнение способов аутентификации

Основываясь на [4], рассмотрим способы аутентификации, которые применяются на мобильных устройствах.

Один из самых распространённых способов — парольная защита, которая основывается на факторе знания секретной фразы или последовательности символов. Реализовать метод можно с помощью простого устройства ввода.

Иногда для аутентификации используются USB-токены или смарт-карты. Они дополняют парольную защиту, добавляя проверку фактора обладания.

Следующий способ — распознавание по голосу. Для аутентификации голос оцифровывается и сравнивается с ранее полученным шаблоном. В связи с тем, что почти все мобильные устройства имеют встроенный микрофон, этот метод достаточно легко внедрить. Однако, голос человека легко имитировать, поэтому аутентификацию по голосу можно использовать в качестве вторичного способа аутентификации.

Также для подтверждения личности используется распознавание лица, получившее распространение среди мобильных устройств после анонса компанией Apple технологии FaceID в 2017 году. Надёжнее всего использовать трёхмерные сканеры, которые реализуются при помощи фронтальной камеры и дополнительного датчика. Распознавание лица является простым в реализации, если используется только камера.

Сканирование отпечатка пальца — распространённый метод подтверждения личности пользователя, который реализован в большинстве смартфонов, выпущенных за последние пять лет. Стоит отметить, что при использовании сканеров отпечатка пальца случаются ошибки. К тому же, отпечаток пальца можно подделать [8], [9]. Поэтому производители стараются

повысить надёжность технологии, используя, например, мониторы сердечного пульса, встроенные в сканер отпечатка пальца.

Пользователя можно аутентифицировать, используя географическое местоположение пользователя и устройства, которые определяются при помощи GPS. Стоит отметить, что GPS сигналы легко могут быть заглушены и искажены. Поэтому применяется несколько источников местоположения, например, GPS и ID устройства в беспроводной сети.

Поведенческая биометрия представляет собой новую технологию относительно существующих способов подтверждения личности. Для создания поведенческого профиля пользователя может использоваться сразу несколько датчиков. Можно собирать данные о подключенных сетях, анализировать клавиатурный почерк, исследовать походку человека. Для этого используются акселерометр, гироскоп, сенсорный дисплей и другие датчики. Подделать поведенческий профиль человека сложно ввиду индивидуальности стиля пользователя. Такой способ аутентификации удобен для человека, так как процесс проходит в фоновом режиме. При этом распознавание пользователя может выполняться непрерывно, что повышает уровень защиты от несанкционированного доступа.

Для сравнения перечисленных способов построена таблица 1. Сравнение проведено по следующим критериям [11]:

- универсальность (отражает возможность применения к любому человеку вне зависимости от его индивидуальных особенностей);
- уникальность (насколько хорошо можно однозначно отличить двух людей по признакам, использующимся в подходе);
- собираемость (как легко собирать данные для построения модели);
- задержка (много ли времени проходит от момента начала распознавания до получения вердикта);
- надёжность (насколько сложно подделать аутентификационные данные и получить несанкционированный доступ к системе);

- непрерывность (можно ли использовать данный метод аутентификации непрерывно без навязывания пользователю дополнительных действий);
- удобство (насколько быстро и удобно пользователь может зайти в систему);
- распространённость (как сильно распространена технология в современных мобильных устройствах).

В таблице высокий уровень обозначается буквой H (high), средний — M (medium), низкий — L (low).

Таблица 1 — Сравнение способов аутентификации

	Универсальность	Уникальность	Собираемость	Задержка	Надёжность	Непрерывность	Удобство	Распространённость
Парольная защита	M	L	H	M	L	L	L	H
USB-токен, смарт-карта	L	H	H	M	M	M	L	L
Отпечаток пальца	M	H	M	L	H	L	H	H
Распознавание лица	H	M	M	L	M	M	H	H
Распознавание голоса	L	M	M	M	M	H	H	L
Местоположение	M	L	H	M	L	M	H	M
Поведенческая биометрия	H	H	M	L	M	H	H	M

На основе сравнения (табл. 1), можно отметить, что поведенческая биометрия обладает преимуществами перед остальными способами подтверждения личности. Поведенческие характеристики сложно подделать, аутентификационные данные можно собирать достаточно легко. Также с

помощью поведенческой биометрии можно обеспечить непрерывную аутентификацию с малым интервалом времени между двумя процедурами распознавания пользователя. При этом данный метод пока не получил широкое распространение. Поведенческая биометрия может применяться наравне с остальными методами аутентификации пользователей или в комбинации с ними при реализации многофакторной аутентификации.

2 Обзор способов аутентификации в мобильных устройствах, основанных на поведенческой биометрии

Есть много методов аутентификации, основанных на поведении пользователя. Для выбора способов аутентификации для будущего исследования следует рассмотреть уже хорошо изученные методы.

Для сравнения методов аутентификации важно обратить внимание на метрики ошибок — уровни ошибок первого и второго рода, FRR и FAR соответственно. Одна из проблем, возникающих при использовании биометрии — это ошибка первого рода [12]. Под ошибкой первого рода при аутентификации подразумевается событие, когда зарегистрированному пользователю ошибочно отказано в доступе. В противоположном случае возникает ошибка второго рода, когда система ошибочно пропускает незарегистрированного пользователя. Также в качестве метрики используется величина EER, которая равна значению первых двух метрик FAR и FRR, когда они равны между собой. В случае мобильных устройств значение FRR должно быть как можно ниже, потому что никто не захочет себе устройство, которое не будет аутентифицировать своего владельца несколько раз подряд. При этом необходимо предотвращать попытки несанкционированного доступа к смартфону. Поэтому для системы, которая требует высокий уровень безопасности, значение FAR также должно быть как можно меньше [12].

Параметры FAR и FRR взаимосвязаны таким образом, что при уменьшении одного увеличивается другой [13]. Поэтому приходится находить компромисс в зависимости от требований, предъявляемых к системе. На рисунке 3 можно видеть, как FAR и FRR зависят от порога доверия, установленного в системе. При увеличении порога растёт уровень неверно отклонённых пользователей, что может быть приемлемо в случае необходимости высокого уровня защищённости. При уменьшении же порога FRR пользователь не так часто будет сталкиваться с ложным отказом в доступе, но при этом возрастёт вероятность несанкционированного доступа. Иногда в качестве порога

эффективно выбрать значение, соответствующее равенству FAR и FRR на графике, то есть EER (зелёная пунктирная линия на рис. 3).

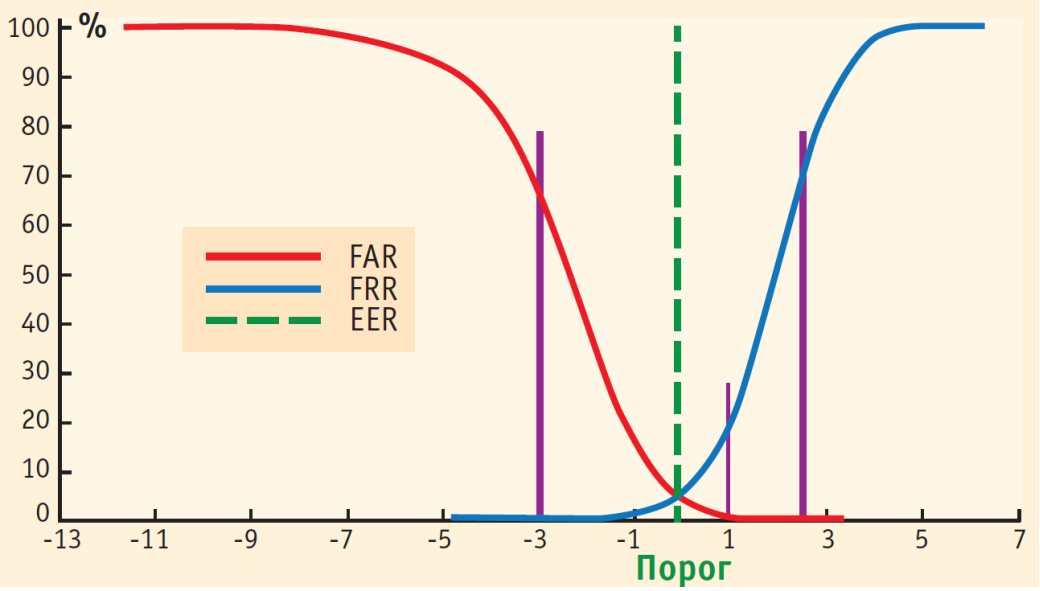

Рисунок 3 — Графики FAR и FRR [14]

Для решения задачи аутентификации по поведенческой биометрии используют методы машинного обучения, так как это эффективный способ обработки больших массивов признаков, которые формируются из данных, собираемых в ходе наблюдения за поведением пользователя.

Признаки, использующиеся в поведенческой биометрии, можно разделить на два класса [5]: устройство-независимые и устройство-зависимые. Устройство-независимые признаки — те признаки, которые давно изучаются и формируются независимо от того, использует ли человек смартфон. Например, походка, подпись. На устройство-зависимые же признаки влияет модель устройства и программные решения. Поэтому в этом случае имеет значение модель устройства и версия операционной системы на нём.

Устройство-зависимые признаки в свою очередь можно разделить на три категории [5]:

- базирующиеся на физическом взаимодействии пользователя с устройством (клавиатурный почерк, движение пальца при пролистывании);
- стилевые (написание текстовых сообщений, действия в браузере, стиль кода для программистов);

- связанные с знаниями и навыками человека, которые проявляются при его взаимодействии с программным обеспечением (стратегия в игре, навыки вождения автомобиля).

Согласно работе Sultana [5] в эру Интернета спектр наших привычек и активностей расширился, включив поведение в виртуальном мире. Поэтому вводят [5] новую категорию устройство-зависимых признаков — социально-поведенческую биометрию на основе активности пользователя в сети [5]. Эта категория включает действия пользователя в социальных сетях. Например, ретвиты, репосты, лайки, публикации, фотографии. Аутентификация по паттернам поведения в сети — новое предложение и представляет интерес. Ежедневно пользователи социальных сетей заходят в онлайн и совершает множество действий. Выгрузить накопленную информацию можно при помощи API, которые предоставляют социальные сети. Однако, стоит отметить, что подобный способ биометрии можно назвать отложенным, так как с момента начала отслеживания действий пользователя должно пройти достаточное время, чтобы накопилось минимальное количество данных для анализа системой. В зависимости от конкретного человека требуемый интервал времени может сильно варьироваться из-за того, что каждый человек ведёт себя в социальных сетях отлично от других.

В данной работе рассмотрим методы аутентификации, использующие как устройство-зависимые, так и устройство-независимые признаки. Остановимся на аутентификации по особенностям использования сенсорного экрана, биометрии по лингвистическому и поведенческому профилю, а также аутентификации по характерным перемещениям человека.

2.1 Аутентификация на основе клавиатурного почерка

К 2020 году сенсорные экраны стали распространённым и привычным устройством ввода в мобильных устройствах. Благодаря этому можно собирать и анализировать данные об индивидуальных особенностях взаимодействия

пользователя с экраном. Можно выделить два направления в способах аутентификации по поведенческим особенностям работы с сенсорным дисплеем:

- аутентификация на основе клавиатурного почерка;
- аутентификация по динамике взаимодействия с экраном.

Первое использует более специфические признаки, которые характерны только при наборе текста. Второе же включает в себя все остальные взаимодействия с сенсорным экраном, включающие в себя, например, пролистывания меню, нажатия для открытия приложений.

Распознавание личности по манере набора текста — один из старейших методов аутентификации пользователей. В качестве характерных признаков для построения модели клавиатурного почерка могут быть использованы [6], [12], [15], [16]:

- время между освобождением одной клавиши и нажатием на следующую;
- интервал между двумя последовательными нажатиями клавиш;
- время удерживания пальца на одной клавише;
- количество нажатий на клавишу возврата;
- расстояние в пикселях между двумя нажатыми клавишами;
- скорость пальца — вычисляется на основе расстояния и времени в движении;
- число нажатий клавиши Shift;
- сила нажатия на клавиши (поддерживается рядом устройств);
- площадь нажатия на клавишу.

Trojahn и Ortmeier исследовали смешанную схему аутентификации [12], основанную на почерке человека и манере набора текста на клавиатуре. Они собрали данные в ходе двух экспериментов. В первом 18 участников исследования вводили предложение, содержащее два и более слов десять раз при помощи виртуальной клавиатуры на мобильном телефоне HTC Desire с ОС

Android 2.2. Во втором эксперименте 16 персонам было предложено ввести пароль восемь раз на смартфоне HTC Desire HD с ОС Android 2.3.5.

При обработке полученных данных исследователи использовали программное обеспечение WEKA [17]. Trojahn и Ortmeier выбрали несколько моделей классификации: решающие деревья (J48) [18], K* [19], многослойный персептрон (MLP) [20], нейронная сеть на основе радиально-базисной функции (RBF) [21], BayesNet [22], наивный байесовский классификатор (Naive Bayes) [23]. Значения метрик были усреднены по всем участникам исследования, вводившим текст. В первом эксперименте достигнуто среднее значение EER — 2.0%, во втором — 13.5%. Модели J48, K*, MLP, RBF, BayesNet и Naive Bayes дали следующие средние значения FAR — 2.03%, 3.49%, 2.52%, 1.13%, 5.56% и 19.29%, соответственно. FRR — 2.67%, 2.21%, 3.0%, 4.47%, 2.59% и 1.8%, соответственно.

Kambourakis и другие авторы предложили метод, основанный на скорости и расстоянии движений пальца во время набора текста [15]. Такой способ нацелен на улучшение качества классической методики анализа клавиатурного почерка. Для сбора данных исследователи воспользовались помощью 20 человек, которым было предложено вводить парольные и другие фразы по 12 раз. Авторы работы разработали два сценария. В первом субъекты исследования вводили фиксированный пароль, состоящий из цифр и букв. Во втором сценарии необходимо было ввести фиксированную осмысленную фразу. Для построения модели использовались три классификатора: случайный лес [24], метод k ближайших соседей (k-NN) [25] и MLP [20]. При первом сценарии лучший результат получен при использовании случайного леса: FRR — 39.4%, FAR — 12.5%, EER — 26.0%. При втором сценарии лучший результат показал k-NN: FRR — 3.5%, FAR — 23.7%, EER — 13.6%.

Draffin, Zhu, Zhang предложили новую систему аутентификации KeySens [16], базирующуюся на микро-поведенческих особенностях пользователя при его взаимодействии с клавиатурой смартфона. В исследовании участвовали 13

человек, благодаря которым были собраны данные о 86000 нажатиях на клавиши и 430000 точках касания экрана. Для классификации использовалась обученная нейронная сеть [26]. После пяти нажатий на клавиши система определяла незарегистрированного пользователя в 67.7% случаев и FAR составил 32.3%, FRR — 4.6%. В сессии ввода, состоящей из 15 нажатий, злоумышленник был детектирован в 86.0% случаев, FAR — 14.0%, FRR — 2.2%.

2.2 Аутентификация по двигательной активности

Проводить аутентификацию пользователей мобильных устройств можно и на основе анализа походки. Существует несколько подходов к биометрии по походке [6], осуществляемых:

- на базе компьютерного зрения;
- при помощи напольных датчиков;
- с помощью носимых гаджетов.

В контексте нашей работы интересен только последний подход, так как первые два не применимы в мобильных устройствах.

В качестве носимого устройства могут выступать, например, мобильные телефоны, фитнес-браслеты, спортивные трекеры. Необходимо, чтобы в устройстве присутствовали встроенные сенсоры, которые помогают извлекать признаки для построения модели походки человека. Обычно, используются следующие датчики:

- акселерометр;
- гироскоп;
- гравиметр.

Обработка признаков может быть проведена с использованием циклов или без них [27]. Циклу соответствует два полных шага. Циклические признаки формируются из сопоставления циклов походки человека во время прогулки. Без использования циклов просто выделяется отрезок времени, в котором фиксируются данные с датчиков без анализа циклов.

Derawi и другие исследовали поведенческую биометрию на основе распознавания походки с помощью мобильного телефона [28]. Данные собирались с акселерометра, записи координат по трём осям добавлялись с частотой около 40-50 в секунду в текстовый файл. На рисунке 4 изображены оси акселерометра, для каждой из которых снимаются значения ускорения. Телефон находился у участвовавших в исследовании добровольцев в кармане на правой стороне бедра. В ходе испытаний участникам было предложено пройти туда и обратно вдоль прямого коридора, длиной около 37 метров, пол которого был устлан ковром. Испытуемые должны были идти в привычной манере. В исследовании участвовал 51 человек. Каждый из них повторял испытание два раза в два различных дня.

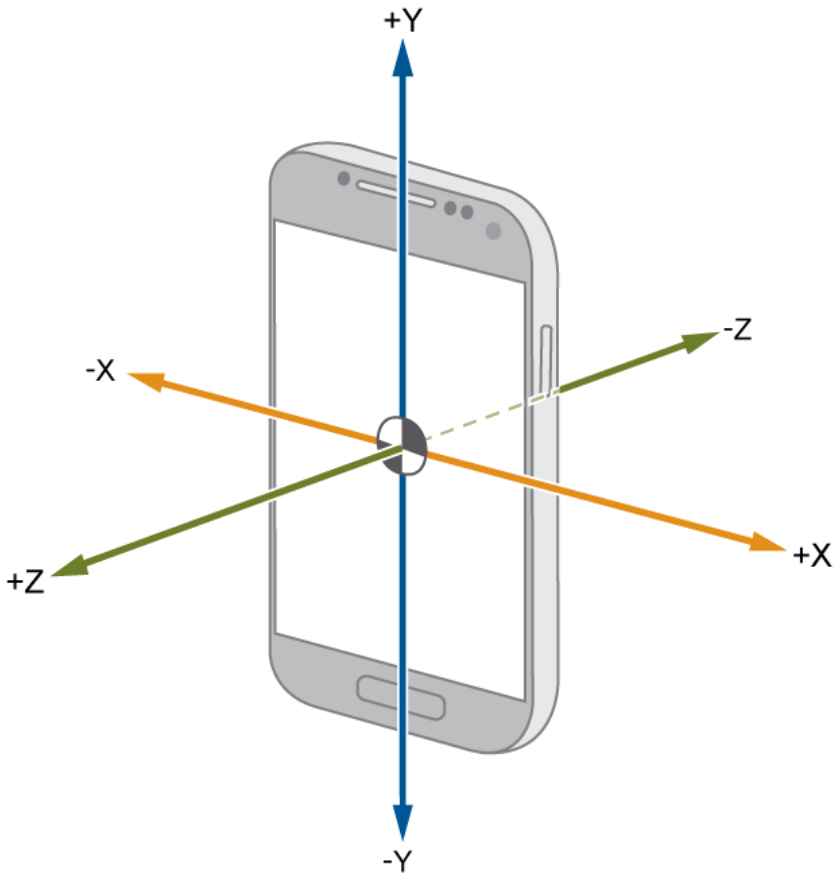

Рисунок 4 — Оси X, Y и Z относительно мобильного телефона [29]

Телефон выводит значения только тогда, когда происходит изменения в датчике. Таким образом, временные интервалы между двумя точками отсчета (значениями ускорения) не всегда равны. Поэтому к данным была применена

интерполяция по времени. Обработка шума была выполнена методом взвешенного скользящего среднего. Также была вычислена средняя продолжительность цикла и найдены все циклы для каждой прогулки испытуемого вдоль коридора.

На рисунке 5 визуализированы данные, полученные с акселерометра. Для каждой из осей (x, y и z) представлена зависимость значения ускорения от времени. Пунктирными линиями обозначены характерные промежутки двигательной активности.

Для сравнения векторов признаков использовался алгоритм динамической трансформации временной шкалы (DTW) [30]. Было достигнуто значение ERR 20.1%.

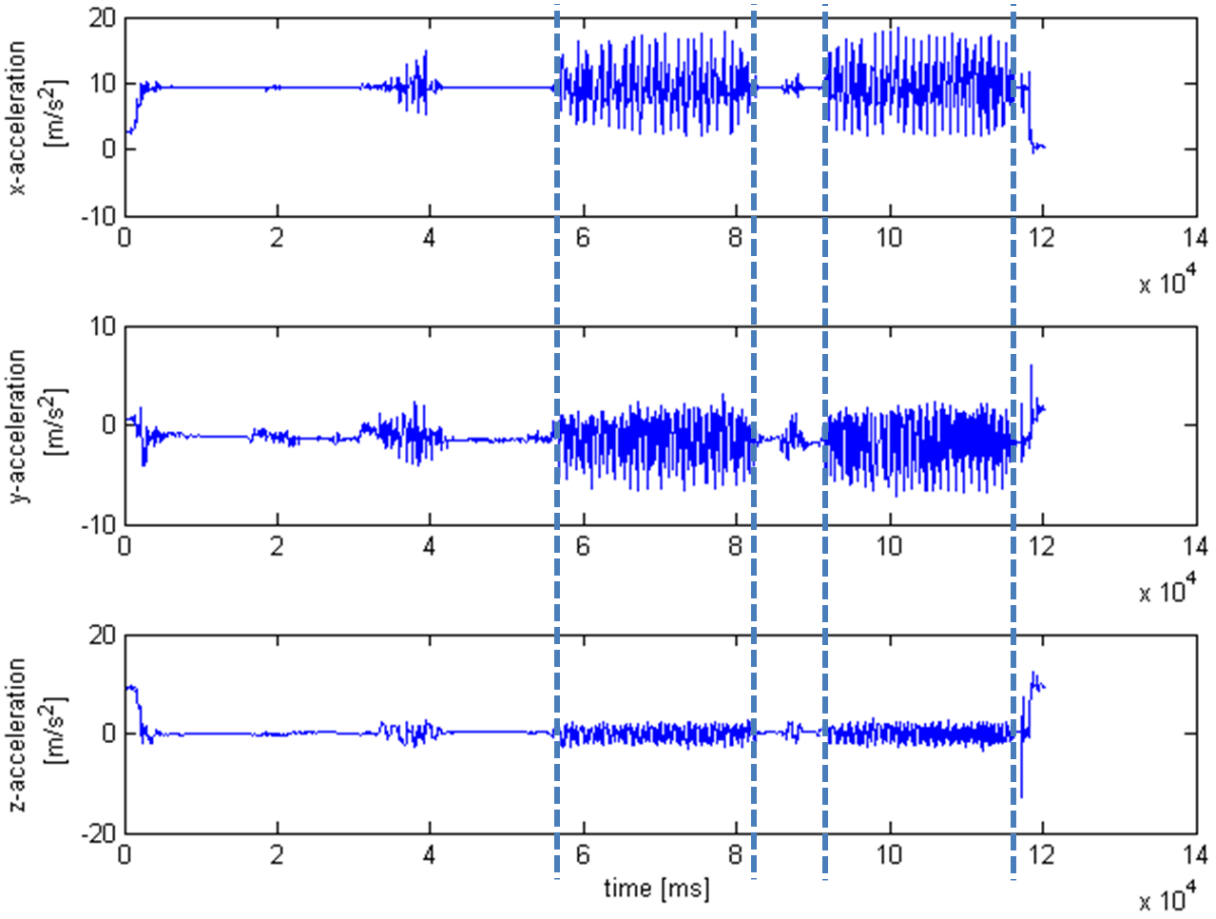

Рисунок 5 — Показания акселерометра для каждой оси [28]

Nickel, Wirtl, Busch применили алгоритм k ближайших соседей (k-NN) [25] к классификации походок [31]. Информация о походке собиралась с встроенного в мобильный телефон под управлением ОС Android акселерометра. Данные о

походке были получены от 36 участников исследования. В секунду датчиком фиксировалось в среднем по 127 значений ускорения по трём осям. Полученные данные были разбиты на сегменты одинаковой временной продолжительности (от 3 до 7,5 секунд) с перекрытием в 50%. Для вычисления расстояния между векторами признаков было использовано евклидово расстояние. Для k-NN была выбрана библиотека WEKA [17]. EER составил от 8.24% до 8.85%.

Также Nickel и Busch в следующей работе применили скрытые марковские модели (HMM) [32] для аутентификации по манере ходьбы [33]. В эксперименте были использованы данные, собранные в предыдущем исследовании [31]. Для тестового вектора признаков вычислялись вероятности совпадения с векторами других моделей. Соответственно, классификация осуществлялась на основе этих вероятностей путём выставления порога для принятия решения к какому классу относить вектор. С использованием принципа голосования [34] был достигнут EER — 5.81%, без использования голосования — 7.45%.

2.3 Аутентификация на основе поведенческого профиля

Среди подходов к поведенческой биометрии отдельно стоит выделить аутентификацию по поведенческому профилю человека. Поведенческий профиль может состоять из нескольких независимых сведений о пользователе, связанных с сервисами, которые он использует на своём устройстве [35]. Для построения профиля используются:

- статистика использования приложений;
- статистика использования смартфона;
- статистика энергопотребления устройства;
- подключение к WiFi сетям;
- географическое местоположение устройства;
- лингвистический профиль (стилометрия);
- статистика запросов в браузере и т.п.

Таким образом, поведенческий профиль представляет совокупность закономерностей в активности человека внутри мобильного устройства.

Одна из особенностей аутентификации по поведенческому профилю — это задержка. В течение некоторого времени фиксируются пользовательские действия и затем выносится вердикт. При этом такой временной интервал может доходить до нескольких десятков минут. Ввиду задержки вынесения вердикта такую систему аутентификации стоит применять как дополнение к основной.

Fridman и другие исследовали непрерывную аутентификацию, основанную на текстовой стилометрии, статистике использования приложений и браузера и местоположении пользователя [10]. Предложенная система аутентификации многомодульная, так как данные приходят от нескольких источников (текст, приложения, браузер, GPS). В зависимости от того, как пользователь использует свой смартфон, информации от одного модуля может быть больше, чем от остальных. Поэтому исследователи решили использовать метод комбинирования вердиктов, приходящих от разных модулей [36].

Данные были собраны у 200 добровольцев, каждый из которых пользовался устройством как минимум 30 дней. С частотой в одну секунду отслеживалась информация о тексте, который вводил пользователь, о посещённых приложениях и сайтах, а также фиксировалось местоположение посредством GPS или WiFi.

Для каждого из 200 пользователей и для каждого из 4 модулей (текст, приложения, браузер, локация) был обучен бинарный классификатор, который выдавал вердикт. В массиве данных для обучения классификаторов производилось разбиение на два класса. Классу записей, соответствующих валидному пользователю, выставлялась метка 1, а всем остальным записям от 199 пользователей присваивалась метка 0.

Для лингвистического анализа использовался метод n-грамм [37]. Для данных браузера и приложений был использован метод максимального правдоподобия (ММП) [38]. Классификация местоположений проводилась с помощью метода опорных векторов (SVM) [39]. Для принятия окончательного вердикта использовалась модель решающего смешивания [40].

При построении такой системы классификации краеугольным камнем является фиксированный временной интервал, в течение которого собираются данные. Схема того, как происходит сбор и обработка событий, представлена на рисунке 6. В течение определённого промежутка времени собирается какое-то количество информации от различных источников, после чего классификаторы выносят свои решения, которые обрабатываются специальным модулем компоновки решений (DFC — Data Fusion Center). Чем больше промежуток времени, в течение которого происходит сбор данных, тем меньше вероятность ошибки классификатора. При этом нельзя выставлять слишком большую задержку, так как в продолжительное временное окно злоумышленник может легко получить доступ к устройству. Зависимость FAR и FRR для каждого модуля в зависимости от временного окна представлена на рисунке 7.

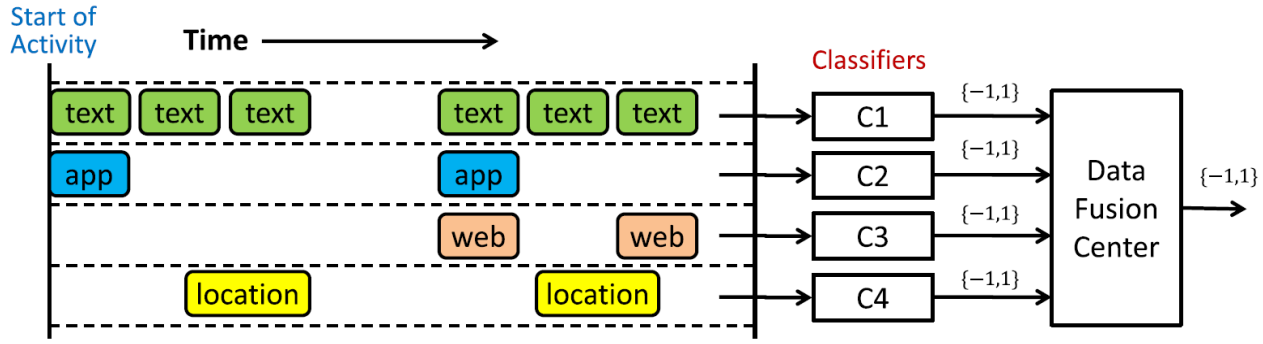

Рисунок 6 — Схема обработки событий классификаторами и принятия решения DFC [10]

По графикам (рис. 7) видно, что лучшее качество показала классификация по местоположению с помощью GPS. FAR составил менее 10% и FRR — менее 5%. Стилометрический подход же показал худший результат. Авторы отмечают, что убывание уровней ошибок с увеличением временного окна не такое сильное, как можно было бы ожидать. Это происходит из-за того, что большинство событий, исходящих от пользователя, приходят вспышками на фоне общей неактивности. При смешивании вердиктов авторам удалось добиться значения EER — 5% при размере окна в одну минуту и 1% при размере окна в 30 минут.

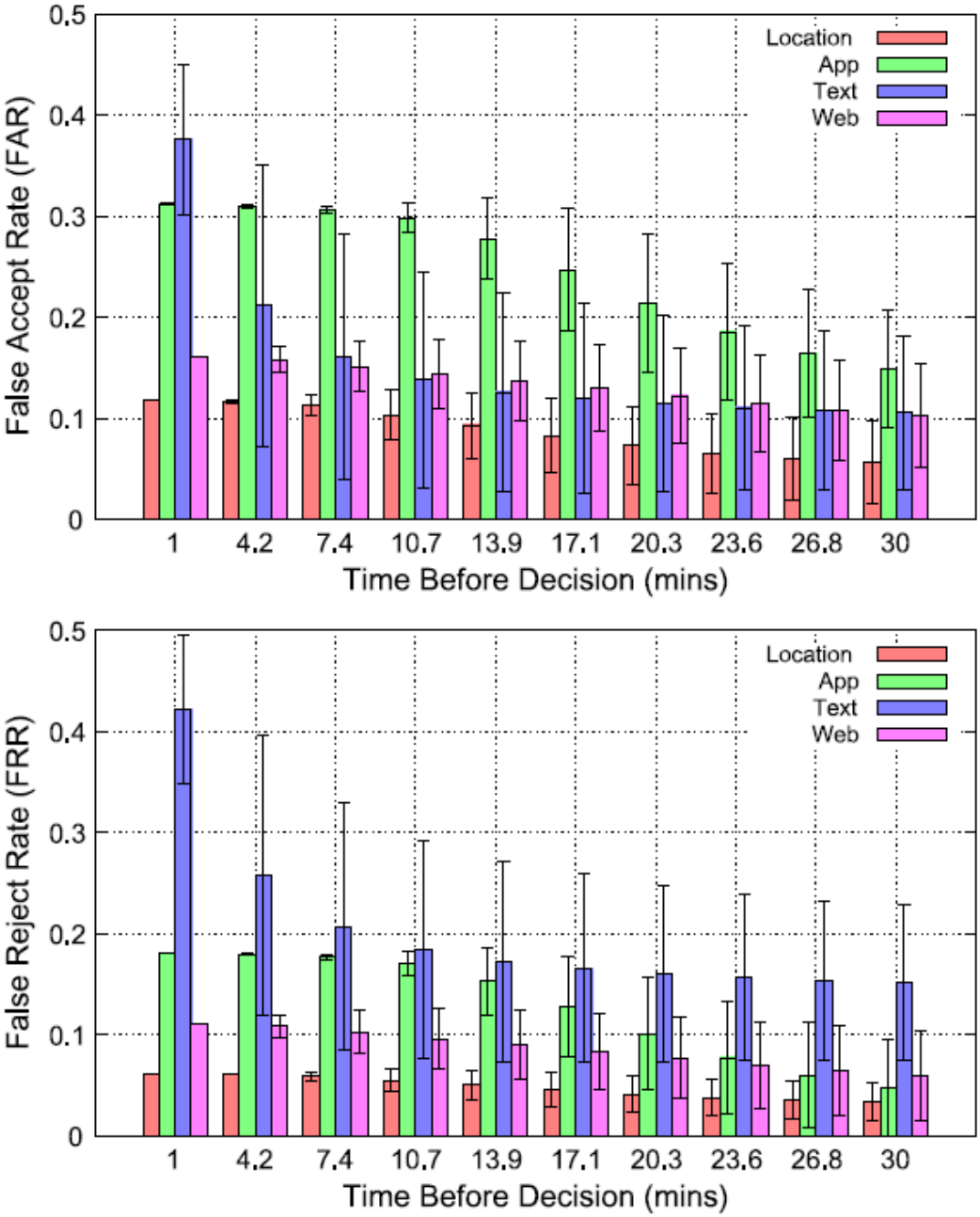

Рисунок 7 — Зависимость FAR и FRR для каждого модуля от временного интервала, отведенного на сбор данных [10]

Saevanee и другие изучили аутентификацию на основе клавиатурного почерка в совокупности с лингвистическим профилем [41]. Использовались данные 30 пользователей, содержащие текстовые сообщения и события с клавиатуры. В эксперименте сравнивались показатели качества методов как по отдельности, так и в комбинации. Рассмотрим результаты применения лингвистического профиля. Признаки для текстовой стилометрии были разбиты

на четыре класса: словарный профиль, лексические, синтаксические и структурные. Словарный профиль пользователя был составлен с помощью частотного распределения 133 аббревиатур и эмоциональных слов. Для создания лингвистического профиля пользователя был использован t-критерий Стьюдента, с помощью которого признаки сортировались по их индивидуальности для человека. Для классификации использовались k-NN [25], радиально-базисная функция (RBF) [21] и многослойный персептрон прямого распространения (MLP) [20]. Средний EER для лингвистического профиля составил 12.8%.

Li и другие провели исследование непрерывной аутентификации на основе поведенческого профиля пользователя [42]. В эксперименте использовался общедоступный набор данных, предоставленный Массачусетским технологическим институтом [43]. Данные включают в себя записи действий 106 пользователей. Авторы разделили приложения, используемые на мобильных устройствах, на две категории: информация о стандартных приложениях включает в себя время использования приложения и местоположение устройства, а приложения с расширенным профилем позволяют получить дополнительную информацию, например, телефонный номер или запрос в браузере.

Для стандартных приложений авторы исследования использовали название приложения, время и место доступа. Местоположение определялось по идентификатору сотовой ячейки, в радиусе действия которой находилось устройство. В качестве расширенных признаков выступали номер телефона и продолжительность звонка. Для классификации были выбраны три метода: нейронная сеть на основе радиально-базисной функции (RBF) [21], многослойный персептрон прямого распространения (MLP) [20] и специально разработанный алгоритм на основе статистического критерия.

В первом эксперименте модели обучались на информации о телефонных звонках. Лучший результат RBF был получен при использовании телефонного

номера и местоположения устройства во время звонка. FAR составил 10.7%, FRR — 10.2%, EER — 10.5%. Признаки продолжительность звонка и время начала звонка ухудшали результат. С помощью MLP получили следующие результаты: FAR — 14.9%, FRR — 20.1%, EER — 17.5%. Также авторы применили разработанный алгоритм, основанный на статистическом критерии. Основой для такого подхода послужила статистика, полученная при анализе данных, а также большие дисперсии, наблюдаемые в обработанной информации. Такой метод основан на предположении, что составленный по известным событиям профиль пользователя может быть использован для предсказания вероятности нового события. Применяя разработанный подход, исследователям удалось добиться FAR — 7.1%, FRR — 14.8%, EER — 11%. Лучшие показатели также были получены при использовании только телефонного номера и локации звонка.

Во втором эксперименте множество признаков в профиле пользователя было расширено путём добавления дополнительной информации приложений. Таким образом полноценное поведенческое профилирование было использовано во втором испытании. Li и другие авторы решили применить во втором опыте только разработанный алгоритм, так как в предыдущем эксперименте он показал результаты, немногим уступающие RBF, при этом RBF требует в два раза большей вычислительной мощности. Для улучшения качества системы аутентификации авторы использовали динамический профиль пользователя, который обновлялся каждые 7, 10 и 14 дней. Это было сделано для того, чтобы минимизировать эффект, вызванный нерегулярными привычками и особенностями поведения пользователя. Для приложений с функцией звонка был получен лучший EER — 5.4%, для SMS-приложений EER — 6.4%. В обоих случаях лучше всего показал себя динамический профиль с интервалом в 14 дней. При использовании всех установленных на устройстве приложений удалось добиться EER — 9.8% с динамическим профилем, рассчитываемым за 10 дней.

Отдельно стоит рассмотреть подход, основанный на распознавании лингвистического профиля пользователя (текстовая стилометрия).

Brocardo и другие авторы рассмотрели способ проверки подлинности автора коротких сообщений при использовании стилометрии [37]. Среди различных стилистических признаков в тексте использовать лучше те, которые сильнее всего отличают пользователей друг от друга. Исследователи решили использовать методику n-грамм для анализа текстов, предложив новый подход, который заключается в анализе исключительно присутствия или отсутствия n-граммы в образце текста.

Эксперименты проводились на наборе данных Enron [44], который состоит более чем из 200 000 электронных писем от приблизительно 150 пользователей. Среднее количество слов в одном письме составило 200. Для классификации авторы разработали собственный алгоритм, который для каждого пользователя рассчитывал порог. После чего были подсчитаны метрики FAR и FRR для каждого пользователя. Лучшие результаты алгоритм показал с использованием 5-грамм: FRR — 14.71%, FAR — 13.93%. EER достиг 14.35%.

2.4 Сравнение и оценка методов аутентификации на основе поведенческой биометрии

В таблице 2 представлено сравнение характеристик методов поведенческой биометрии из проанализированных работ.

Лучше всего исследованы методы поведенческой биометрии, которые использовались до массового вхождения в обиход мобильных устройств. Соответственно, методы аутентификации, реализованные с помощью новых возможностей, появившихся с распространением мобильных устройств, меньше изучались.

При этом такие методы имеют потенциал к развитию. Биометрия по поведенческому профилю, например, может быть более энергоэффективной, нежели непрерывная аутентификация по походке. Среди рассмотренных способов реализации поведенческой биометрии биометрия по поведенческому

Таблица 2 — Сравнение подходов поведенческой биометрии

Публикация	Метод	Количество испытуемых	Классификаторы	EER (%)	FAR (%)	FRR (%)
[12]	Клавиатурный и классический почерк	16, 18	J48, K*, MLP, RBF, BayesNet, Naive Bayes	2.0-13.5	1.13-19.29	1.8-4.47
[15]	Клавиатурный почерк: динамика движений пальца	20	Случайный лес, k-NN, MLP	13.6-26.0	12.5-23.7	3.5-39.4
[16]	KeySens: микро-поведенческий клавиатурный почерк	13	—	—	14.0-32.3	2.2-4.6
[28]	Распознавание походки с использованием циклов	51	DTW	20.1	—	—
[31]	Распознавание походки	36	k-NN	8.24-8.85	—	—
[33]	Распознавание походки	36	HMM	5.81-7.45	—	—
[10]	Поведенческий профиль: приложения, браузер, геолокация, текст	200	Метод n-грамм, SVM, DFC, ММП	1-5	—	—
[41]	Лингвистический профиль	30	k-NN, RBF, MLP	12.8	—	—
[42]	Поведенческий профиль: приложения, звонки, местоположение	106	RBF, MLP, статистический критерий	5.4-17.5	7.1-14.9	10.2-20.1
[37]	Стилометрия по коротким сообщениям	150	Алгоритм на основе метода n-грамм	14.35	13.93	14.71

профилю представляет интерес, так как является относительно новым и малоизученным подходом. Во всех приведённых работах были достигнуты сходные уровни ошибок. Это означает, что не существует лучшего подхода к поведенческой биометрии и рассмотрения и исследования заслуживаю все.

3 Метод аутентификации по поведенческой биометрии для мобильного устройства

В данной работе предлагается метод непрерывной аутентификации, основанный на данных с нескольких модулей:

- LOCATION — модуль регистрации местоположения пользователя;
- WIFI — модуль слежения за WiFi-сетями;
- BT — модуль слежения за Bluetooth-устройствами.

Каждый модуль позволяет на основе соответствующих данных вынести вердикт о подлинности пользователя.

LOCATION отслеживает геолокацию пользователя с использованием системы глобального позиционирования GPS.

WIFI отслеживает окружение из сетей, к которым устройство пользователя не подключено и фиксирует подключенную сеть.

BT фиксирует окружение из устройств Bluetooth.

С помощью данных, собираемых модулями, формируются признаки, которые будут использоваться классификаторами, реализованными с помощью алгоритмов машинного обучения. При этом решение о допуске пользователя в систему может приниматься на основе вычисления композиции вердиктов модулей, подобно системе DFC в [10].

На рисунке 8 представлено схематическое изображение предлагаемой системы. Модуль DECIDER отвечает за итоговый вердикт. От каждого модуля в решающий модуль поступают готовые решения от соответствующих классификаторов. Затем модуль комбинирует решения и выдаёт конечный итоговый результат. Также модуль DECIDER должен реализовывать функцию ожидания новых событий, так как все данные собираются модулями асинхронно и могут приходить в различные временные моменты.

Выбор модулей для системы исходил из соображений эффективности, энергосбережения и новизны. Методы аутентификации, основанные на одном

или нескольких модулях из выбранных [10], [37], [41], [42] показали хорошие результаты (таблица 2).

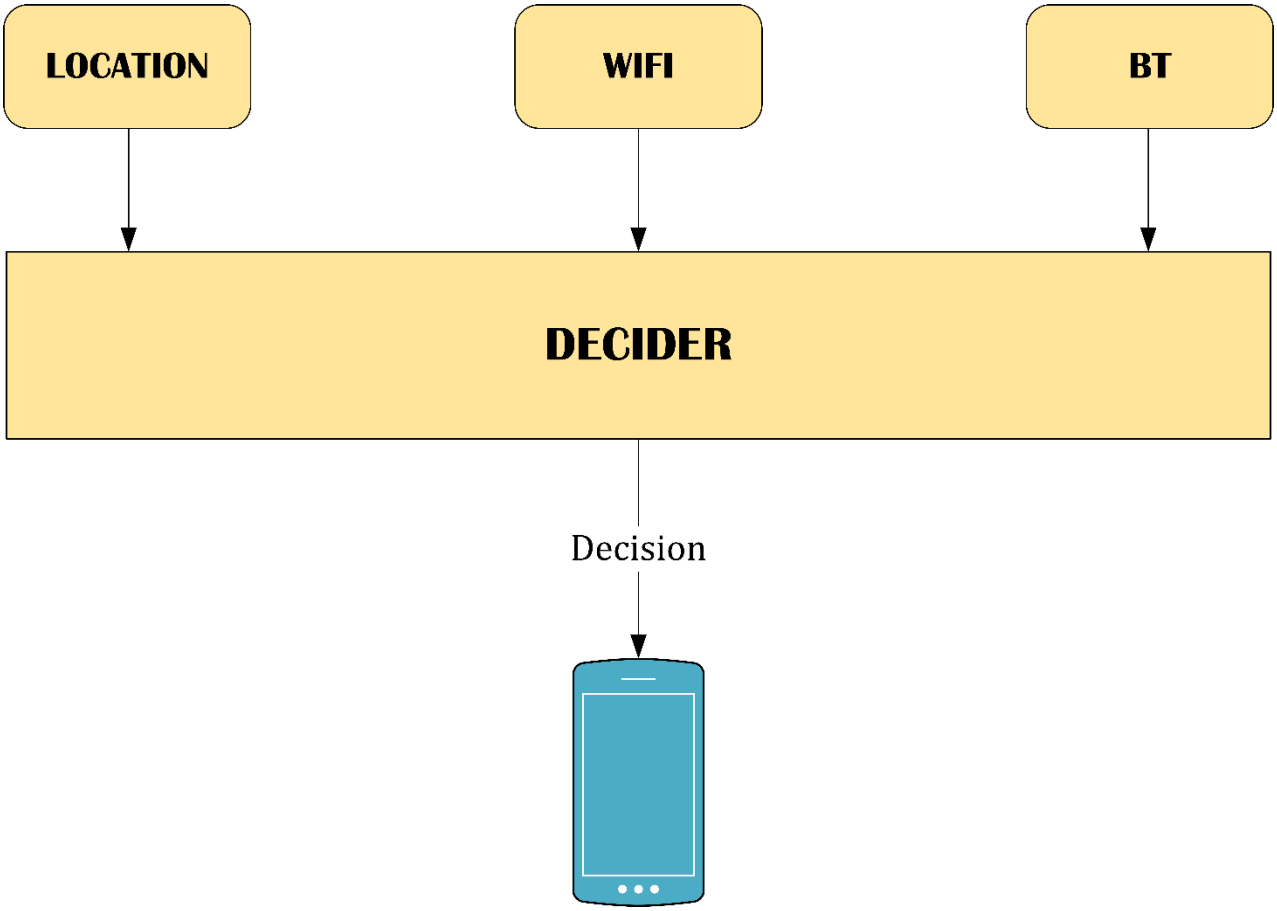

Рисунок 8 — Схема многомодульной системы аутентификации

Большую роль в построении подобной системы играет интервал времени, за который агрегируются приходящие события [10]. Система должна предоставлять возможность менять размер этого временного окна в зависимости от требуемого уровня безопасности.

3.1 Особенности внедрения и эксплуатации метода аутентификации

Рассматриваемый метод может эксплуатироваться в двух различных сценариях.

В первом случае система аутентификации, основанная на предложенном методе, может быть интегрирована в мобильное приложение, которое является посредником между пользователем и компанией, которая предоставляет некий пакет услуг, например, банк. При этом пользователь при входе в приложение

проходит процедуру идентификации и аутентификации, а его поведенческий профиль формируется и хранится удалённо компанией — поставщиком услуг.

Есть вероятность, что у человека будут похищены данные, используемые для аутентификации, например, логин и пароль. Тогда потенциальный злоумышленник сможет войти в приложение на своём устройстве под именем жертвы. В таком случае, система, которая встроена в исходное приложение должна будет зафиксировать активность преступника и отправить её по защищённому каналу на сервер компании для проверки. И в случае, если будет обнаружена подозрительная активность, приложение будет заблокировано, а злоумышленник не сможет осуществить доступ к предоставляемым услугам.

Во втором случае система может функционировать непосредственно на устройстве пользователя с целью предотвращения несанкционированного доступа к информации на телефоне в целом. Этот вариант предполагает непрерывную работу системы и отслеживание подозрительной активности на устройстве в целом, а не только в отдельном приложении.

Эти два подхода имеют одно существенное отличие — для второго сценария недопустимо основывать функционирование системы на данных, специфичных конкретному местоположению пользователя, тогда как для первого такой подход приемлем. Первый вариант предполагает защищённый доступ к услугам посредством устройства. Предполагается, что злоумышленник будет пытаться получить доступ со *своего* мобильного телефона, тогда как второй вариант реализует защиту *устройства пользователя* в реальном времени. Если же построить систему на основе знания о место-специфичных особенностях поведения пользователя, то второй возможный вариант внедрения системы становится нереализуем, так как нет гарантий, что попытка несанкционированного доступа не будет произведена в той же локации, которая типична для легального пользователя. Если же построить систему на основе независимых от конкретного местоположения характеристик, то она потенциально сможет функционировать как в первом, так и во втором случае.

4 Мобильное приложение для сбора данных

Наборы данных поведенческого профилирования пользователей, находящиеся в открытом доступе [44] сформированы специфическим образом. Во-первых, как правило, такие данные были собраны ещё до резкого роста популярности мобильных устройств с сенсорным экраном. Во-вторых, они не обладают всеми признаками, которые были бы необходимы для полноценной проверки работоспособности предложенного метода аутентификации. При нехватке данных для одного из модулей становится невозможным проводить исследование метода. Поэтому было принято решение реализовать мобильное приложение, которое собирает данные пользователей для исследования.

Было разработано мобильное приложение под операционную систему Android, реализующее возможность сбора данных для дальнейшего исследования метода аутентификации. Приложение было реализовано на языке программирования Java, с использованием Android API 29 [45]. Исходный код компонентов приложения представлен в приложении А. Также исходный код и прочие файлы, используемые для сборки приложения представлены на портале GitHub под свободной лицензией [46].

На рисунке 9 представлен главный экран приложения. Пользователь может управлять приложением, используя три кнопки. Кнопки «Start» и «Stop» позволяют включать и выключать процесс сбора данных поведения, а круглая кнопка с пиктограммой конверта позволяет отправлять собранные данные в формате ZIP.

4.1 Служба сбора и записи данных

Основным компонентом приложения является служба (service) [45], исполняющаяся в отдельном процессе операционной системы. Служба запускается пользователем посредством кнопки «Start» и работает в режиме переднего плана (foreground) [45] до её остановки кнопкой «Stop».

Служба управляет модулями WIFI, BT и LOCATION. Каждый из модулей по расписанию запускает сканирование или запрашивает необходимую

информацию, после чего принимает результаты и записывает их в соответствующий файл формата CSV, хранящийся в системном каталоге операционной системы. Запись происходит с помощью специального реализованного модуля записи в файл. Расписанием предусмотрен сбор информации каждые 5 секунд. Таким образом, 5 секунд — минимальный интервал времени, за который можно анализировать полученную информацию.

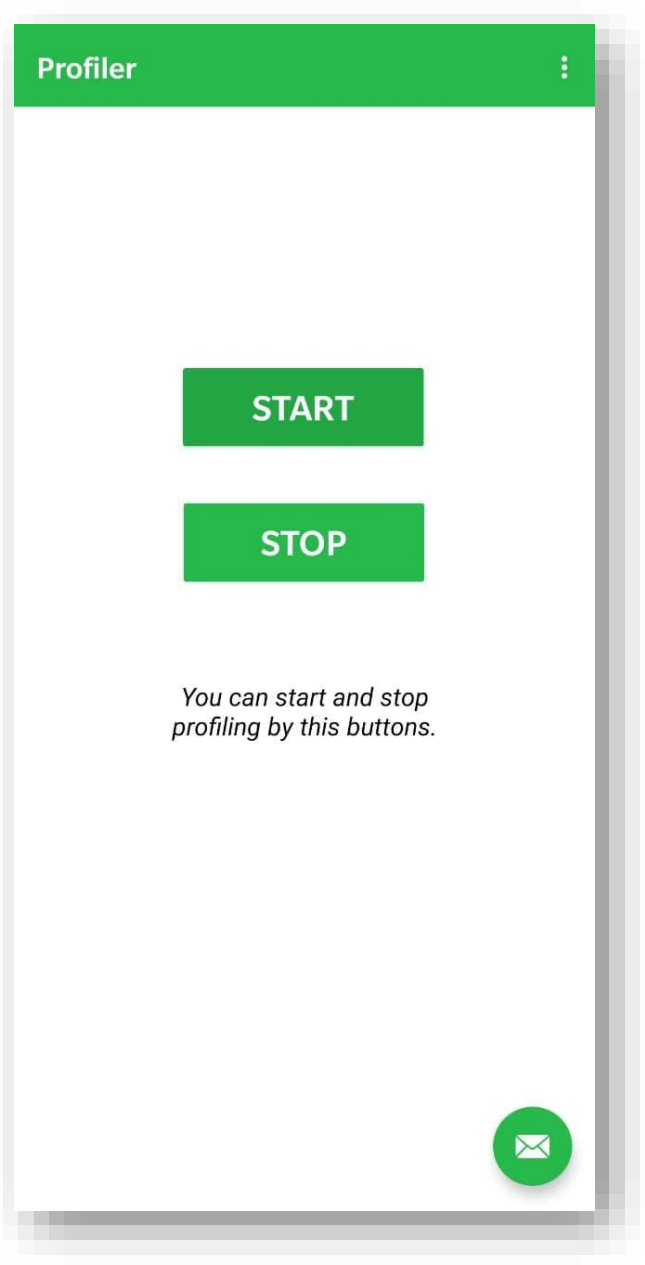

Рисунок 9 — Главный экран приложения

4.2 Собираемые данные

Каждый модуль приложения записывает получаемую информацию в отдельный файл CSV. Всего формируется 4 файла: wifi.data, bt.data и location.data.

Модуль WIFI сохраняет в файл результаты сканирования сетей, находящихся поблизости. Для каждого сканирования записываются все найденные сети и их характеристики. Отдельно отмечается сеть, к которой пользователь подключен в текущий момент времени. Для этого используются возможности класса WifiManager [45].

Модуль BT собирает информацию о Bluetooth-устройствах, которые расположены поблизости, и их параметрах. Задействованы методы, предоставляемые классами BluetoothDevice, BluetoothLeScanner, BluetoothManager [45].

Модуль LOCATION получает информацию о геопозиции пользователя с помощью класса FusedLocationProviderClient из Google API.

4.3 Проведение сбора данных

В ходе исследования были задействованы 8 добровольцев, каждому из которых было предложено установить приложение к себе на устройство и активировать службу сбора информации. В течение 5 дней каждый участник отправлял собранные данные, после чего выключал службу и удалял приложение со своего устройства. Все устройства, используемые участниками в ходе исследования, работали на операционной системе Android 10.

5 Обработка данных поведенческого профиля

Обработка полученных данных была осуществлена в несколько этапов:

- подготовка данных к формированию признаков;
- формирование признаков для каждого модуля;
- подготовка выборок для обучения моделей.

Для обработки данных и формирования признаков были разработаны сценарии на языке программирования Python. Для различных задач были использованы следующие подключаемые библиотеки: Pandas [47], NumPy [48], GeoPy [49], SciPy [50], Matplotlib [51]. Исходный код сценариев приведён в приложении Б.

Перед формированием признаков данные от каждого пользователя были подготовлены к обработке, а именно:

- были распакованы архивы формата ZIP, содержащие CSV-файлы;
- для каждого модуля был сформирован единый CSV-файл, содержащий всю собранную информацию;
- данные были отфильтрованы, чтобы исключить записи, зарегистрированные в то время, когда пользователь не использовал устройство (устройство было заблокировано).

5.1 Формирование признаков

Полученные данные представляют собой упорядоченный во времени массив событий. Поэтому было решено для модулей WIFI, BT и LOCATION агрегировать такие события по временным интервалам разной длины, создавая признаковое описание окружения пользователя за определённый интервал времени.

Учитывая необходимость создания признаков, которые бы не зависели от конкретных особенностей, присущих местоположению или устройству пользователя, решено было отказаться от использования таких признаков как координаты пользователя, уникальные WiFi-сети и Bluetooth-устройства. В то время как в [10] авторы классифицировали пользователей на основе их

местоположения по данным с GPS и WiFi, в данной работе планируется не использовать подобные признаки (точные координаты), а рассчитывать другие величины, которые описывали бы окружение пользователя, но при этом не основывались на место-специфичных параметрах. Так, например, вместо того чтобы использовать координаты устройства, предполагается рассчитывать скорость по этим координатам. Такой признак более общий и не несёт в себе информацию о месте, где пользователь находится.

Формирование одних и тех же признаков проводилось для каждого пользователя отдельно. Затем полученные выборки векторов признаков сливались в одну.

Для каждого модуля можно ввести понятие *элементарного события* — одной записи в CSV-файле, которая содержит значения ряда параметров. Для WIFI — это одна запись о сети, для BT — об одном устройстве, для LOCATION — запись с координатами. Так как каждое такое элементарное событие малоинформативно отдельно от других, решено было сгруппировать элементарные события в группы для того, чтобы на их основе формировать вектора признакового пространства.

5.1.1 Формирование признаков для модуля WIFI

События от модуля WIFI приходили в сформированных операционной системой результатах сканирования, поэтому решено было не менять такую группировку.

Агрегирование элементарных событий по группам проводилось следующим образом.

В каждой группе было подсчитано общее количество событий (количество сетей в результате сканирования).

Был выделен параметр — показатель уровня принимаемого сигнала (RSSI) [52] сети, к которой был подключен смартфон во время сканирования, соответствующего группе. В случае, когда устройство не было подключено к WiFi, RSSI подключенной сети приравнивался к 0.

Для некоторых характеристик, присущих элементарному событию, было рассчитано среднее арифметическое по всей группе:

- частота, на которой функционирует сеть;
- RSSI сети.

Таким образом был сформирован один вектор из четырёх признаков $F_1 = [\, feature_0, ..., feature_3]$, где $feature_i$ — значение i-ого признака.

Также были составлены ещё два вектора, которые описывают присутствие сетей и их RSSI в данной группе элементарных событий по следующему правилу. Допустим, что все сети WiFi, представляющие уникальные элементарные события, пронумерованы от 0 до $n-1$, где n — общее число уникальных сетей в данных одного пользователя. Для выделения уникальных сетей был использован их MAC-адрес (BSSID) [52]. Тогда получим следующие вектора для каждой группы элементарных событий (1):

$$
\begin{aligned}
&F_2 = [net_0, net_1, \ldots, net_n], \\
&F_3 = [rssi_0, rssi_1, \ldots, rssi_n], \\
&net_i = \begin{cases} 1, \textit{ если } i \in G_j \\ 0, \textit{ если } i \notin G_j \end{cases}, \\
&rssi_i = \begin{cases} rssi_i^j, \textit{ если } i \in G_j \\ 0, \textit{ если } i \notin G_j \end{cases},
\end{aligned}
\tag{1}
$$

где $rssi_i^j$ — значение RSSI i-ой сети, когда она находится в j-ой группе, G_j — множество номеров сетей WiFi, присутствующих в j-ой группе элементарных событий.

Полученные вектора можно интерпретировать как координаты устройства пользователя в пространстве сетей WiFi.

В результате первого этапа агрегирования для каждой группы элементарных событий был получен набор векторов (F_1, F_2, F_3). При этом наборы

векторов упорядочены по времени. На основе сформированного набора векторов для каждой группы были вычислены признаки, которые отражают характер перемещений устройства среди сетей WiFi с течением времени. Для расчёта новых признаков выбран размер окна m, равный количеству групп, по которым рассчитанные признаки усреднялись по формуле среднего арифметического (1). В случае, если между соседними наборами векторов был промежуток времени более 10 минут, значение нового признака принималось равным 0.

5.1.1.1 Число возникших сетей

Усреднённое по окну количество сетей, которое появилось в радиусе обнаружения за время между двумя сканированиями. Нормировано относительно общего количества сетей в текущей и предыдущей группах. Рассчитывалось по формуле (2):

$$appeared\_nets\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m}\frac{|G_j \setminus (G_j \bigcap G_{j-k})|}{|G_j \bigcup G_{j-k}|}, \quad (2)$$

где m — размер выбранного окна (количество групп), G_j — множество номеров сетей WiFi, присутствующих в *j*-ой группе элементарных событий.

5.1.1.2 Число исчезнувших сетей

Усреднённое по окну количество сетей, которое пропало за время между двумя сканированиями. Рассчитывалось по формуле (3):

$$disappeared\_nets\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m}\frac{|G_{j-k} \setminus (G_j \bigcap G_{j-k})|}{|G_j \bigcup G_{j-k}|}, \quad (3)$$

5.1.1.3 Расстояние Жаккара

Усреднённое по окну расстояние Жаккара (дополнение индекса Жаккара) [53] характеризует схожесть двух результатов сканирования по сетям,

присутствующим в текущей и предшествующей группах (результатах сканирования). Для вычисления использовалась формула (4):

$$J\_dist\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m}\left(1-\frac{|G_j \bigcap G_{j-k}|}{|G_j \bigcup G_{j-k}|}\right), \quad (4)$$

5.1.1.4 Изменение вектора сетей между сканированиями

Евклидово расстояние между двумя векторами F_2 групп, усреднённое по окну. Использовалась формула (5):

$$net\_dist\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m}\sqrt{\sum_{i=0}^{n-1}(x_j^i - x_{j-k}^i)^2}, \quad (5)$$

где x_j^i — i-ый элемент вектора F_2 из набора, сформированного для j-ой группы элементарных событий.

5.1.1.5 Изменения вектора RSSI между сканированиями

Евклидово расстояние между двумя векторами F_3 групп, усреднённое по окну. Использовалась формула (6):

$$rssi\_dist\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m}\sqrt{\sum_{i=0}^{n-1}(x_j^i - x_{j-k}^i)^2}, \quad (6)$$

где x_j^i — i-ый элемент вектора F_3 из набора, сформированного для j-ой группы элементарных событий.

5.1.1.6 Изменение значения RSSI подключенной сети

Изменение значения RSSI подключенной сети, усреднённое по окну. Рассчитывалось по формуле (7):

$$conn\_rssi\_chg\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m} x_k - x_{j-k}, \quad (7)$$

где x_j — значение RSSI подключенной сети для *j*-ой группы, взятое из вектора F_1.

5.1.1.7 Изменения общего количества сетей

Изменение значения общего количества сетей в группе, усреднённое по окну. Рассчитывалось по формуле (8):

$$count\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m} count_k - count_{j-k}, \quad (8)$$

где $count_j$ — количество сетей в j-ой группы, взятое из вектора F_1.

После проведения вычислений для каждой группы элементарных событий были получены вектора, состоящие из всех элементов вектора F_1 из набора (F_1, F_2, F_3) и новых вычисленных признаков. Новые вектора признаков будем называть *элементарными векторами признаков*, так как они были получены в результате агрегирования элементарных событий.

Для составления окончательного набора векторов, который будет использован для обучения моделей, был проведён ещё один этап агрегации с целью снижения количества векторов и выделения дополнительных признаков, характеризующих динамику поведения пользователя. Агрегация была проведена двумя различными способами:

- путём разбиения всех векторов на непересекающиеся группы по стационарным временным окнам (пример такого разбиения изображён в левой части рисунка 10, цветными рамками изображены группы элементарных векторов при размере окна в 20 секунд);

- путём разбиения с помощью скользящего окна (изображено в правой части рисунка 10, ширина окна — 20 секунд).

timestamp	freq	level	count
2020-12-06 17:56:05.536	3303.470588	-57.235294	17
2020-12-06 17:56:08.521	3184.000000	-46.916667	12
2020-12-06 17:56:12.036	3128.461538	-50.769231	13
2020-12-06 17:56:19.939	3188.166667	-49.583333	12
2020-12-06 17:56:26.836	3356.368421	-59.842105	19
2020-12-06 17:56:30.335	3720.428571	-33.714286	7
2020-12-06 17:56:34.951	3101.555556	-58.111111	18
2020-12-06 17:56:39.942	3176.500000	-61.050000	20
2020-12-06 17:56:44.326	3304.941176	-58.764706	17
2020-12-06 17:56:49.361	3314.400000	-60.550000	20
2020-12-06 17:56:54.354	3483.071429	-54.285714	14
2020-12-06 17:56:59.445	3271.428571	-61.571429	21
2020-12-06 17:57:04.416	3345.437500	-57.750000	16
2020-12-06 17:57:09.427	3247.277778	-59.555556	18

timestamp	freq	level	count
2020-12-06 17:56:05.536	3303.470588	-57.235294	17
2020-12-06 17:56:08.521	3184.000000	-46.916667	12
2020-12-06 17:56:12.036	3128.461538	-50.769231	13
2020-12-06 17:56:19.939	3188.166667	-49.583333	12
2020-12-06 17:56:26.836	3356.368421	-59.842105	19
2020-12-06 17:56:30.335	3720.428571	-33.714286	7
2020-12-06 17:56:34.951	3101.555556	-58.111111	18
2020-12-06 17:56:39.942	3176.500000	-61.050000	20
2020-12-06 17:56:44.326	3304.941176	-58.764706	17
2020-12-06 17:56:49.361	3314.400000	-60.550000	20
2020-12-06 17:56:54.354	3483.071429	-54.285714	14
2020-12-06 17:56:59.445	3271.428571	-61.571429	21
2020-12-06 17:57:04.416	3345.437500	-57.750000	16
2020-12-06 17:57:09.427	3247.277778	-59.555556	18

Рисунок 10 — Схематичное изображение разбиения данных модуля WIFI с помощью стационарного (слева) и скользящего (справа) окна

В группах элементарных векторов для каждого окна и для всех признаков из вектора F_1 были вычислены:

- выборочное среднее;
- несмещённая выборочная дисперсия по формуле (9):

$$S^2 = \frac{1}{n-1}\sum_{i=0}^{n-1}(x_i - \overline{x})^2, \qquad (9)$$

где n — количество элементарных векторов в группе, сформированной окном, $\bar{x}$ — выборочное среднее;

- медиана;
- несмещённый выборочный коэффициент асимметрии [54] по формуле (10):

$$skew = \frac{\sqrt{n(n-1)}}{n-2} \frac{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^3}{\left(\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^2\right)^{\frac{3}{2}}}; \quad (10)$$

- несмещённый выборочный коэффициент эксцесса по формуле (11):

$$kurtosis = \frac{n^2 - 1}{(n-2)(n-3)}\left(\frac{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^4}{\left(\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^2\right)^2} - 3 + \frac{6}{n+1}\right). \quad (11)$$

Для всех признаков, рассчитанных на основе векторов F_2 и F_3, вычислены:

- выборочное среднее;
- несмещённый выборочный коэффициент асимметрии (11);
- среднее абсолютное отклонение по формуле (12):

$$MAD = \frac{1}{n}\sum_{i=0}^{n-1}|x_i - \bar{x}|. \quad (12)$$

На этом формирование признаков для данных, собранных модулем WIFI, было завершено. Всего был сформирован 41 признак.

5.1.2 Формирование признаков для модуля BT

Для BT агрегирование событий также было проведено в два этапа. В отличие от WIFI, элементарные события, поступающие от модуля BT, не были сгруппированы по результатам сканирования. Поэтому группировка элементарных событий была произведена по времени — одна группа соответствовала результатам, полученным за 5 секунд. Такой временной промежуток был выбран исходя из того, что задачи сканирования формировались приложением один раз в 5 секунд.

Для каждой группы было подсчитано:

- количество Bluetooth-устройств;
- среднее значение RSSI сигнала от устройств Bluetooth Low Energy (Bluetooth LE);
- количество устройств Bluetooth LE, которые доступны для подключения.

Таким образом был сформирован один вектор признаков: $F_1 = [\, feature_0, ..., feature_2 \,]$.

Также как и для модуля WIFI, был сформирован вектор, который описывает присутствие устройств в группе элементарных событий, $F_2 = [device_0, device_1, \ldots, device_{n-1}]$. Каждый элемент вектора равняется 1 или 0 в зависимости от того, присутствовало ли устройство в группе. Уникальные устройства были определены по их MAC-адресу.

После первого этапа агрегирования получен набор векторов $\left(F_1^{BT}, F_2^{BT}\right)$ для каждой группы элементарных событий.

Аналогично тому как вычислялись признаки на основе векторов F_1 и F_2 для WIFI были вычислены и признаки для BT на основе вектора F_2^{BT}:

- число возникших устройств (2);
- число исчезнувших устройств (3);
- расстояние Жаккара (4);

- изменение вектора устройств между группами (5);
- изменение количества устройств в группе (8).

С помощью стационарного и скользящего окна был осуществлён второй этап агрегирования. Для признаков из вектора F_1^{BT} были рассчитаны:

- выборочное среднее;
- несмещённая выборочная дисперсия (9);
- медиана;
- несмещённый выборочный коэффициент асимметрии (10);
- несмещённый выборочный коэффициент эксцесса (11).

Для признаков, рассчитанных на основе вектора F_2^{BT}, были вычислены:

- выборочное среднее;
- несмещённый выборочный коэффициент асимметрии (10);
- среднее абсолютное отклонение (12).

Всего было сформировано 30 признаков.

5.1.3 Формирование признаков для модуля LOCATION

Для данных, полученных с модуля LOCATION, отсутствовала необходимость для двухэтапного агрегирования событий. Элементарные события LOCATION представляют собой вектора, содержащие несколько параметров, описывающих локацию пользователя:

- широту (latitude);
- долготу (longitude);
- высоту над уровнем моря (altitude).

К тому же, каждое событие содержит точность, с которой были получены координаты со спутников.

Для элементарных событий были вычислены несколько признаков. Затем, используя стационарное и скользящее окна, была проведена агрегация. Для каждого признака вычислялись:

- выборочное среднее;
- несмещённая выборочная дисперсия (9);

- медиана;
- несмещённый выборочный коэффициент асимметрии (10);
- несмещённый выборочный коэффициент эксцесса (11);
- стандартное отклонение по формуле (13):

$$S = \sqrt{\frac{\sum_{i=0}^{n-1}(x_i - \overline{x})^2}{n-1}}, \tag{13}$$

где $\overline{x}$ — выборочное среднее.

Всего было сформировано 30 признаков.

5.1.3.1 Скорость

Скорость перемещения устройства на основе геодезического расстояния между координатами текущего элементарного события по сравнению с предыдущим была рассчитана по формуле (14):

$$speed = \frac{dist}{time}, \tag{14}$$

где *dist* — расстояние между точками, координаты которых были зафиксированы в событиях (рассчитывалось с помощью подключаемой библиотеки GeoPy, которая реализует алгоритм Karney [55]), *time* — время прошедшее между двумя событиями.

5.1.3.2 Скорость изменения высоты

Скорость изменения высоты устройства над уровнем моря была вычислена по формуле (15):

$$alt\_speed = \frac{altitude_{curr} - altitude_{prev}}{time}, \tag{15}$$

где $altitude_{curr}$ и $altitude_{prev}$ — высота устройства для текущего обрабатываемого события и для предыдущего по времени события соответственно.

5.2 Подготовка выборок для обучения алгоритмов

После формирования векторов признаков необходимо было подготовить выборки для обучения различных моделей машинного обучения.

5.2.1 Масштабирование вектора признаков

Масштабирование признаков имеет большое значение при обучении моделей [56]. Многие алгоритмы машинного обучения чувствительны к масштабу признаков и плохо работают данных разного масштаба. К тому же, признаки, которые были сформированы на предыдущем этапе работы, могут сильно отличаться по масштабу ввиду того, что они характеризуют поведенческие особенности и окружение разных пользователей.

В данной работе решено было прибегнуть к методу минимаксной нормализации [57], [58], при которой все значения признаков приводятся к интервалу $[0,1]$. Сначала вычисляются минимальное и максимальное значения *j*-ого признака на выборке (18, 19):

$$x_{\min} = \min(x_0^j, ..., x_{n-1}^j), \tag{18}$$

$$x_{\max} = \max(x_0^j, ..., x_{n-1}^j) \tag{19}$$

После чего, каждое значение *j*-ого признака на *i*-ом объекте выборки вычисляется следующим образом (20):

$$x_j^i = \frac{x_j^i - x_{\min}}{x_{\max} - x_{\min}}, \tag{20}$$

Масштабирование было осуществлено с помощью встроенных в библиотеку Pandas средств вычисления.

5.2.2 Обработка выбросов

В задачах машинного обучения важной частью подготовки данных является обработка выбросов, которые могут существенно ухудшить качество работы алгоритмов [59].

На рисунке 11 изображена диаграмма рассеяния, на которой по осям расположены значения некоторых признаков, полученных для модуля BT, до устранения выбросов. По горизонтальной оси отмечены значения среднего абсолютного отклонения усреднённого числа исчезнувших устройств между группами элементарных событий, а по вертикальной — коэффициента асимметрии дисперсии расстояния Жаккара между группами. Данные каждого пользователя на рисунке обозначены своим цветом.

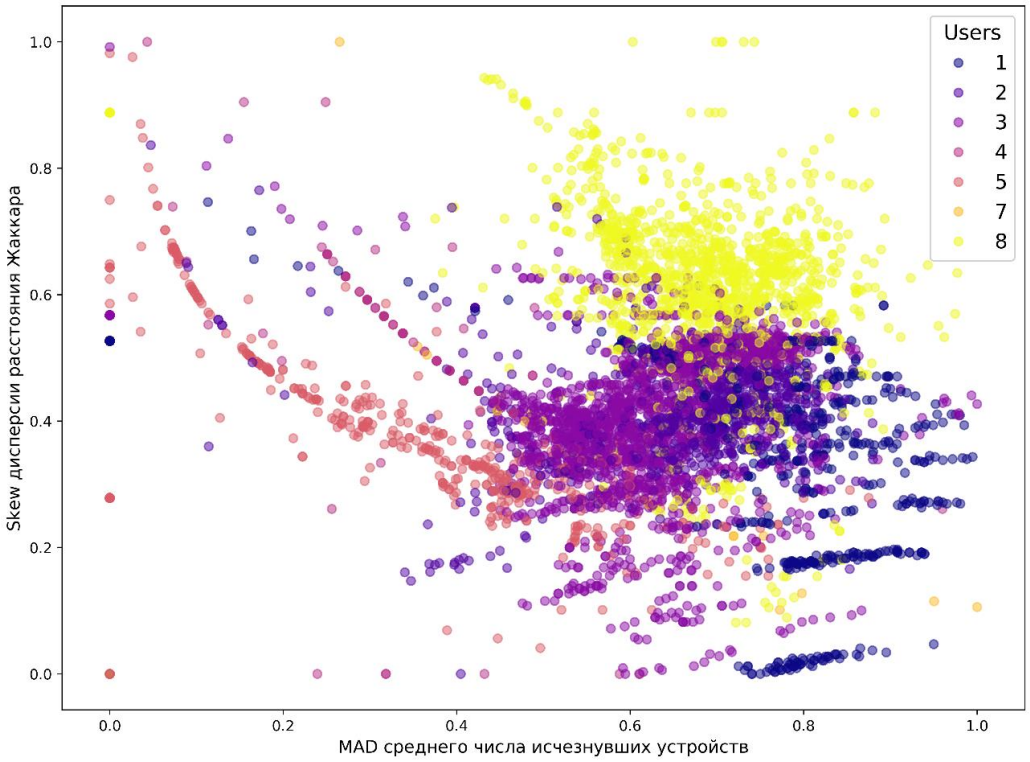

Рисунок 11 — Диаграмма рассеяния для каждого пользователя до устранения выбросов

Один из способов устранения выбросов из выборки — это вычисление z-оценки для каждого признака выборки и удаление векторов признаков, в которых значение z-оценки больше заданного порога [59]. Z-оценка вычисляется по формуле (21):

$$z = \frac{x - \overline{x}}{S}, \tag{21}$$

где $\overline{x}$ — выборочное среднее, S — стандартное отклонение, рассчитывается по формуле (22):

$$S = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \overline{x})^2}. \tag{22}$$

В данной работе z-оценка была вычислена с помощью функции zscore из подключаемой библиотеки scipy.stats. При этом из выборок были удалены все векторы, в которых значение z-оценки принимало значение большее 3 по модулю.

На рисунке 12 изображена диаграмма рассеяния уже после устранения выбросов.

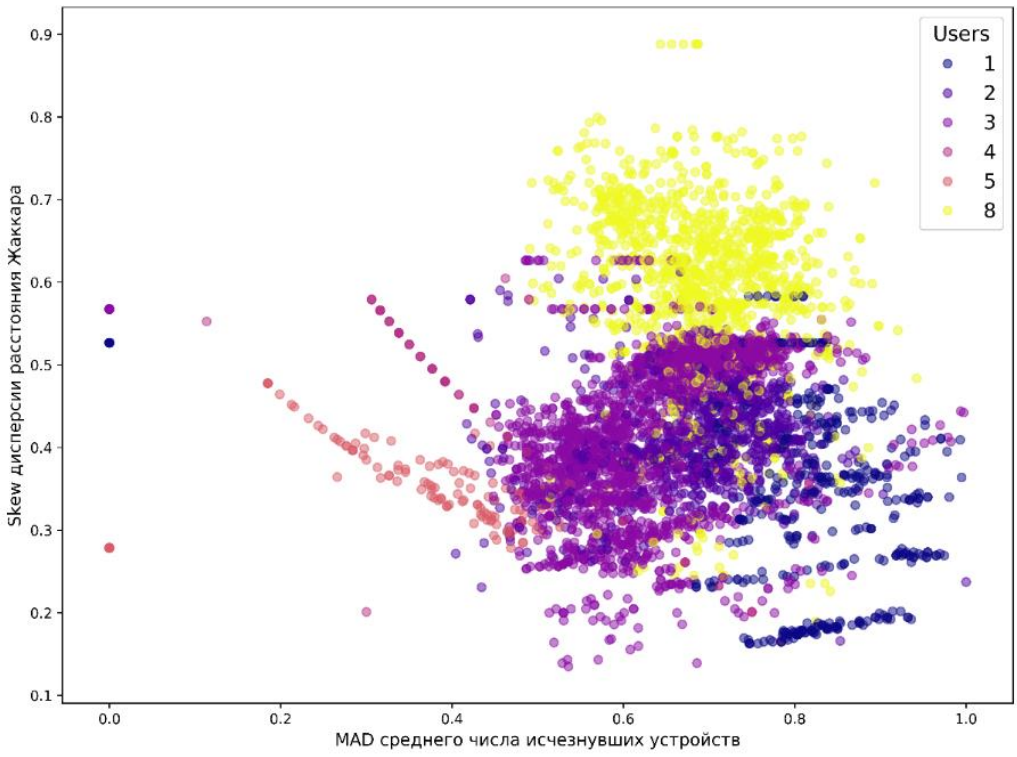

Рисунок 12 — Диаграмма рассеяния для каждого пользователя после масштабирования признаков

Можно видеть, что значения признаков после удаления выбросов лежат в более узком интервал, а также уменьшилось количество удалённых отдельно лежащих точек.

5.2.3 Отбор признаков

Для каждого модуля был сформирован ряд признаков. Для того чтобы понять, стоит ли использовать все эти признаки одновременно, можно оценить значения коэффициента корреляции Пирсона [60] признаков друг с другом. На рисунке 13 представлена матрица коэффициентов корреляции признаков для модуля LOCATION.

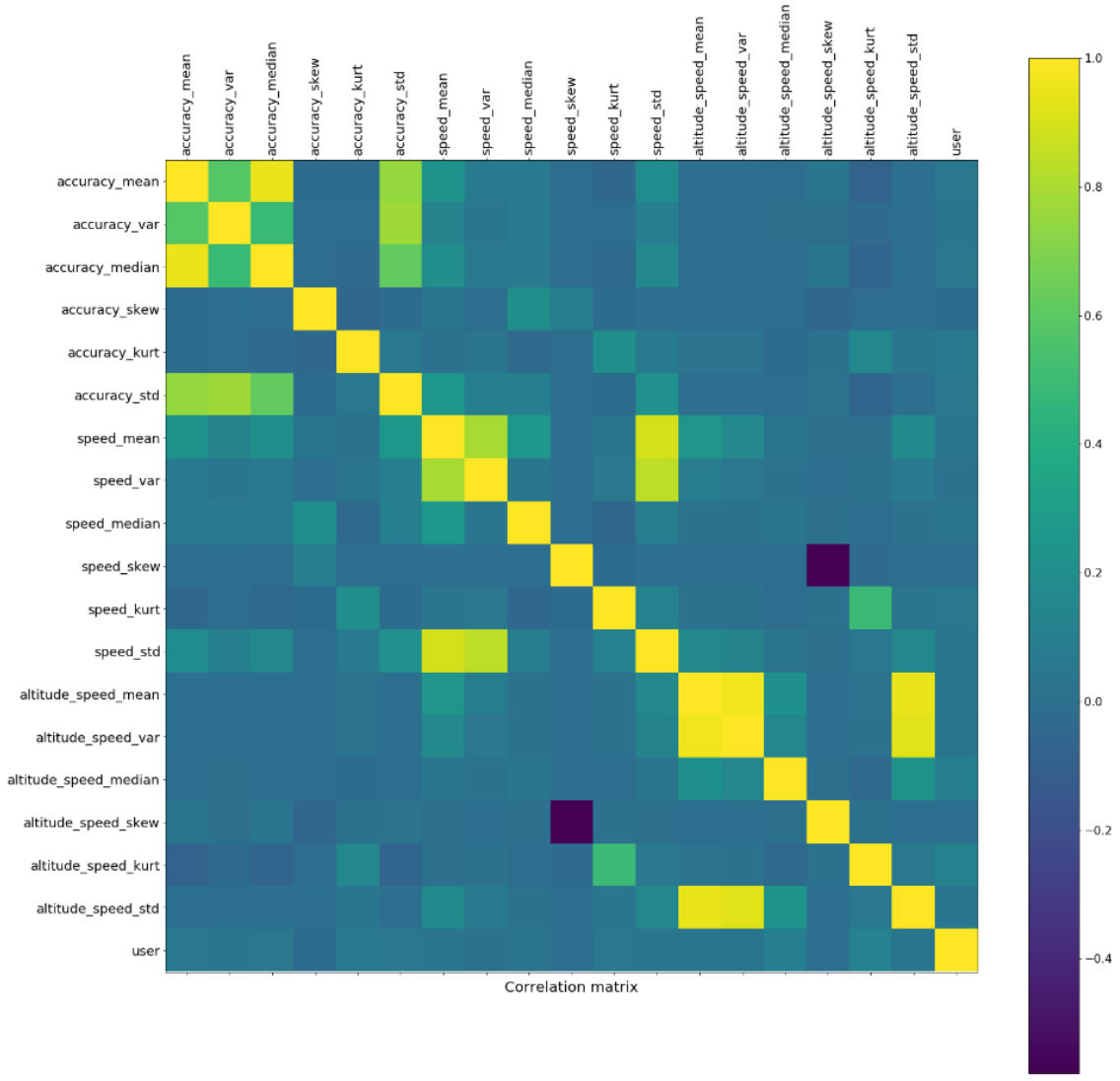

Рисунок 13 — Матрица корреляций признаков между собой до удаления сильно коррелирующих для модуля LOCATION

Коэффициент корреляции рассчитывается следующим образом (23):

$$r_{xy} = \frac{\sum_{i=0}^{n-1}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=0}^{n-1}(x_i - \overline{x})^2 \sum_{i=0}^{n-1}(y_i - \overline{y})^2}}, \quad (23)$$

где x_i и y_i — значения двух признаков на i-ом объекте выборки соответственно, а $\overline{x}$ и $\overline{y}$ — значения выборочного среднего для каждого из признаков соответственно.

В данной работе с помощью функции corr из библиотеки Pandas были посчитаны коэффициенты корреляции всех признаков со всеми и были удалены признаки из пар, в которых коэффициент корреляции составил больше 0.7 по модулю. На рисунке 14 представлена матрица корреляций после удаления. Можно видеть, что после удаления остаются менее коррелирующие признаки.

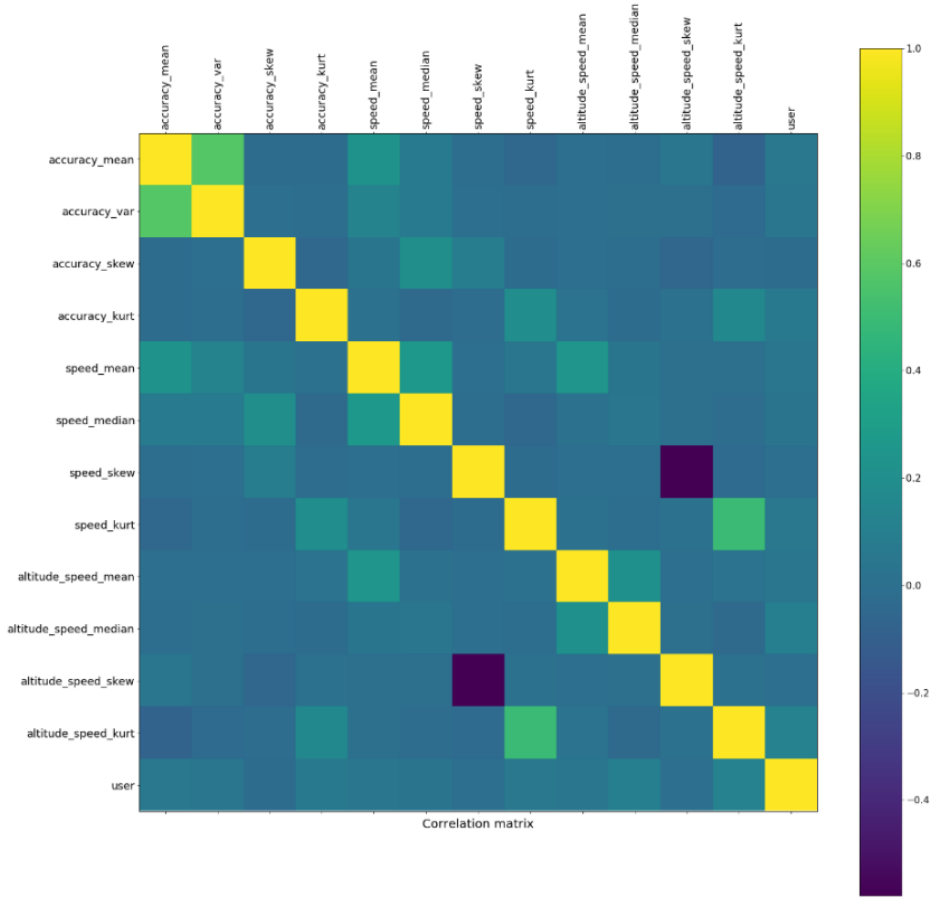

Рисунок 14 — Матрица корреляций признаков между собой после удаления сильно коррелирующих для модуля LOCATION

6 Методы машинного обучения в задаче аутентификации по поведенческой биометрии

Для аутентификации по поведенческой биометрии применяются методы машинного обучения, решающие задачи классификации [61], кластеризации [62] и обнаружения аномалий (одноклассовой классификации) [63]. В данной работе будут рассмотрены методы бинарной классификации. Необходимо, чтобы обученная модель классифицировала пользователей на легальных (зарегистрированных в системе) и несанкционированных (потенциальных злоумышленников).

Будут оценены показатели точности и качества работы нескольких алгоритмов для модулей WIFI, BT и LOCATION.

Для тестирования на данных, полученных с модулей WIFI, BT и LOCATION выбраны 4 алгоритма машинного обучения на основе анализа работ [10], [15], [24], [27], [39]:

- градиентный бустинг [64] (использован CatBoostClassifier из библиотеки CatBoost [65]);
- метод опорных векторов [66] (Support Vector Machine, SVM, использован Support Vector Classifier, SVC, из библиотеки scikit-learn [67]);
- случайный лес [68] (использован RandomForestClassifier из библиотеки scikit-learn);
- логистическая регрессия [69] (LogisticRegression, scikit-learn).

Исходный код сценариев на языке Python, использованных для обучения и тестирования алгоритмов представлен в приложении В.

6.1 Показатели эффективности работы алгоритмов

Работа алгоритмов машинного обучения оценивается с использованием большого количества метрик. Перед тем как остановиться на метриках, используемых в данной работе, рассмотрим какие предсказания может сделать алгоритм [56]:

- истинно-положительные (true positive, TP);

- истинно-отрицательные (true negative, TN);
- ложноположительные (false positive, FP);
- ложноотрицательные (false negative, FN).

Метрики качества, как правило, рассчитываются на основе долей разных типов предсказаний алгоритма на тестовой выборке. В данной работе были использованы следующие метрики [70].

Точность (accuracy) — доля правильных предсказаний алгоритма. Используется формула (24):

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \quad (24)$$

Точность (precision) — доля истинно-положительных предсказаний алгоритма среди всех положительных предсказаний. Вычисляется по формуле (25):

$$precision = \frac{TP}{TP + FP}. \quad (25)$$

Полнота (recall, true positive rate, TPR) — отношение числа истинно-положительных вердиктов к количеству элементов в положительном классе. Рассчитывается по формуле (26):

$$recall = \frac{TP}{TP + FN}. \quad (26)$$

FRR — доля легальных пользователей, отвергнутых системой по ошибке (ошибка первого рода) [71]. Рассчитывается по формуле (27):

$$FRR = FNR = 1 - recall = \frac{FN}{TP + FN}. \quad (27)$$

FAR — доля нелегальных пользователей, пропущенных системой по ошибке (ошибка второго рода) [71]. Рассчитывается по формуле (28):

$$FAR = FPR = \frac{FP}{TN + FP}. \quad (28)$$

EER рассчитывается по формуле (29):

$$EER = \frac{FAR + FRR}{2}. \quad (29)$$

AUC-ROC — площадь под кривой ошибок (ROC-кривой) [72]. ROC-кривая отражает зависимость между FPR и TPR при различных значениях порога принятия решения о принадлежности объекта к классу 1. Площадь под ROC-кривой является показателем качества, инвариантным относительно ошибок первого и второго рода [73]. В случае идеального алгоритма площадь равна 1, в случае худшего — 0.5.

Метрики AUC-ROC и EER обладают большей обобщающей способностью ввиду того, что позволяют оценить качество работы алгоритма в целом, не определяя порог принадлежности объектов к классам. Поэтому их более целесообразно применять при выборе алгоритма и подборе гиперпараметров. Остальные же метрики необходимо применять при более тонком конфигурировании выбранной модели для реализации метода аутентификации.

6.2 Формирование выборок для обучения и тестирование алгоритмов

После обработки признаков были созданы CSV-файлы для каждого модуля, содержащие итоговые наборы векторов для всех пользователей. При этом было сформировано несколько различных наборов. Для каждого способа

формирования окна (стационарное и скользящее) были составлены выборки с размерами окна в 5, 10, 30, 60, 90, 120, 240 и 600 секунд. Это было сделано, чтобы сравнить эффективность алгоритмов для различных размеров окна.

Также решено было тестировать алгоритмы в 2 этапа.

На первом этапе выполнялась кросс-валидация [74] алгоритмов для каждого пользователя следующим образом:

- текущий пользователь был назначен легальным;
- были выбраны данные одного пользователя, отличного от легального, и исключены из обучающей выборки;
- в обучающей выборке была проведена балансировка векторов признаков сначала для различных пользователей, а потом для классов легального пользователя и несанкционированных при помощи метода извлечения случайных векторов (undersampling) [75];
- была сформирована тестовая из данных пользователя, исключенных из обучающей выборки на втором этапе;
- модель была обучена и протестирована.

Заметим, что кросс-валидация для одного легального пользователя проводилась путём исключения на каждом этапе кросс-валидации одного из незарегистрированных пользователей. Система аутентификации, построенная на алгоритмах классификации, обучается на известных данных, после чего на вход модели могут прийти события от неизвестного пользователя. Поэтому был выбрана именно такая схема кросс-валидации, при которой обученная модель тестируется на данных неизвестного ей ранее пользователя.

Для того, чтобы объединить полученные результаты кросс-валидации для каждого алгоритма было рассчитано среднее значение метрики accuracy следующим образом (30):

$$cv\_score = \frac{1}{n(n-1)} \sum_{\substack{1 \le i, j \le n \\ i \ne j}}^{n} score_i^j \,, \tag{30}$$

где n — общее число пользователей (8 человек), $score_i^j$ — значение метрики на j-ой итерации кросс-валидации (модель тестируется на j-ом пользователе) для i-ого пользователя (принятого легальным).

На втором этапе выполнялось финальное тестирование алгоритмов после прохождения каждым алгоритмом кросс-валидации. Для каждого пользователя из выборки выполнялись следующие шаги:

- текущий пользователь был помечен как легальный;
- из выборки полностью были извлечены данные одного из пользователей, отличного от легального;
- из выборки были исключены 25% данных всех пользователей, присутствующих в выборке;
- была проведена балансировка пользователей, а затем классов;
- модель была обучена на подготовленной выборке;
- из данных, извлечённых на предыдущих шагах, была сформирована тестовая выборка, состоящая на треть из данных легального пользователя, на треть из данных несанкционированного пользователя, которые были полностью извлечены из обучающей выборки ранее, оставшуюся треть составляли данные остальных пользователей, сбалансированные между собой;
- модель была протестирована на тестовой выборке.

Такая схема тестирования представляет собой модифицированную кросс-валидацию. Ввиду того, что в тестовую выборку теперь были включены данные легального пользователя, можно рассчитать метрики AUC-ROC и EER.

Для вычисления метрик использовались функции из библиотеки sklearn.metrics [67], а также собственные реализации, представленные в приложении В.

6.3 Результаты тестирования алгоритмов

Каждый выбранный алгоритм был протестирован по описанным выше схемам. Результаты представлены в приложении Г в соответствующих таблицах. На данном этапе работы тестирование моделей проводилось для модулей WIFI, BT и LOCATION по отдельности с целью сравнить их между собой.

Для того, чтобы оценить какие из алгоритмов показали лучшие результаты, были построены таблицы 3 и 4. В таблице 3 отражено количество раз, когда каждый алгоритм давал значение метрики больше, чем остальные. А в таблице 4 — наоборот, количество худших результатов для каждого алгоритма.

Таблица 3 — Количество итераций кросс-валидации и финального тестирования, на которых алгоритмы показали лучший результат

Модуль	CatBoostClassifier	RandomForestClassifier	SVM-SVC	LogisticRegression
BT	0	27	5	0
WIFI	1	19	13	0
LOCATION	1	30	1	0
Всего	2	76	19	0

Суммирование количества попаданий в лучшие и худшие результаты проводилось по всем таблицам из приложения Г для каждого модуля. В случае таблиц Г.7, Г.8, Г.9, Г.10, Г.11, Г.12 учитывались значения AUC-ROC.

Таблица 4 — Количество итераций кросс-валидации и финального тестирования, на которых алгоритмы показали худший результат

Модуль	CatBoostClassifier	RandomForestClassifier	SVM-SVC	LogisticRegression
BT	2	0	1	29
WIFI	2	0	1	29
LOCATION	0	0	3	29
Всего	4	0	5	87

По таблицам 3 и 4 можно сделать вывод, что лучше всего отработал алгоритм классификации на основе случайного леса (RandomForestClassifier) и SVM. Хуже всего показал себя алгоритм логистической регрессии

(LogisticRegression), что было вполне ожидаемо, так как этот алгоритм является линейным, в то время как остальные алгоритмы являются нелинейными.

На рисунке 15 визуализировано значение EER для нескольких алгоритмов, картина согласуется с выводами о том, что логистическая регрессия отстаёт от остальных тестируемых алгоритмов по качеству предсказаний. Можно заметить, что значение метрики постепенно уменьшается с ростом размера окна в секундах.

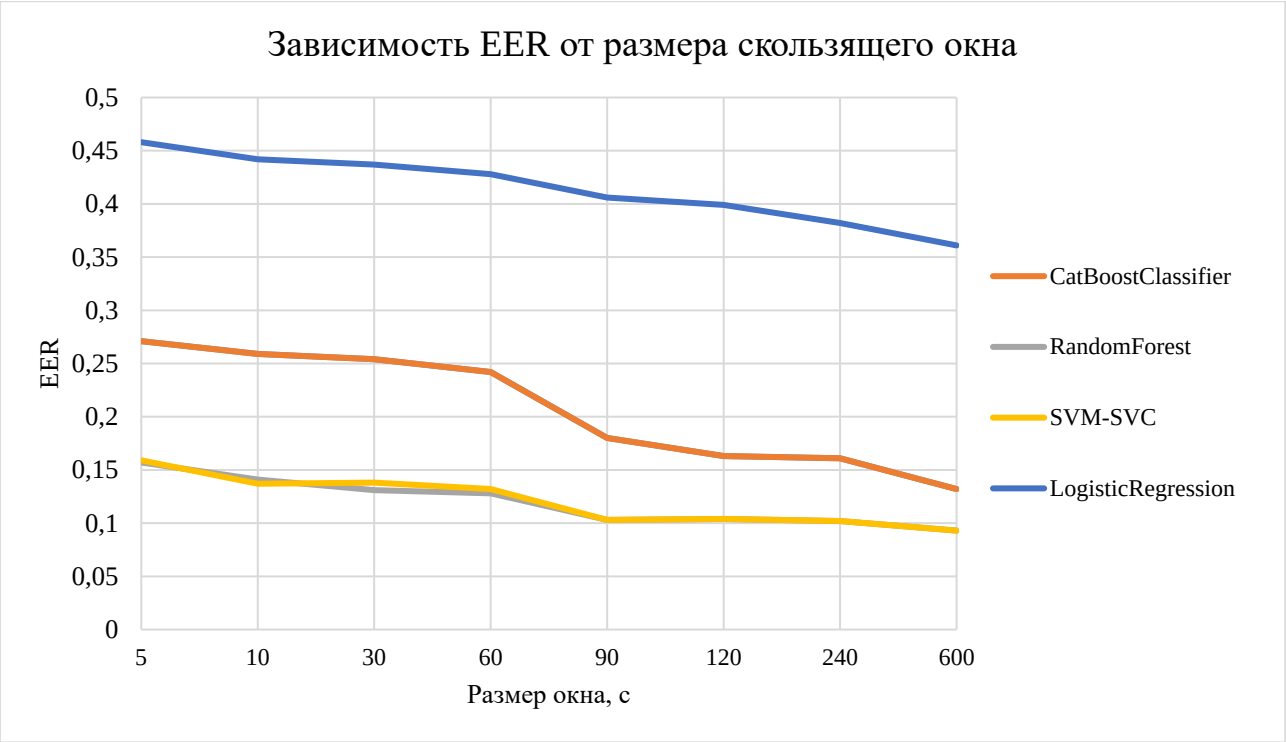

Рисунок 15 — График зависимости значения EER на этапе кросс-валидации для модуля BT (скользящее окно)

Для оценки оптимального размера окна были построены таблицы 5 и 6.

Таблица 5 — Количество лучших результатов алгоритмов для каждого окна

	Размер окна, с							
Модуль	5	10	30	60	90	120	240	600
BT	0	0	0	3	2	3	7	2
WIFI	1	1	5	1	1	3	4	0
LOCATION	1	0	0	0	0	4	7	4
Всего	2	1	5	4	3	10	18	6

В таблице 5 подсчитано количество раз, когда на каждом из окон значение метрики принимало максимальное значение по сравнению со всеми остальными размерами окна, для каждого алгоритма. В таблице 6 же, наоборот, минимальное.

Таблица 6 — Количество худших результатов алгоритмов для каждого окна

	Размер окна, с							
Модуль	5	10	30	60	90	120	240	600
BT	7	9	0	0	0	0	0	0
WIFI	5	6	1	1	0	0	0	4
LOCATION	6	3	3	4	0	1	0	0
Всего	18	18	4	5	0	1	0	4

Видно, что при увеличении размера окна, значения метрик в среднем не убывают. Так, например, для окон размером 120 и 240 секунд было получено больше всего максимальных результатов, а для окон размером 5 и 10 секунд — более всего минимальных.

На рисунке 16 представлена диаграмма, на которой можно заметить, что при росте размера окна значения AUC-ROC в среднем растут.

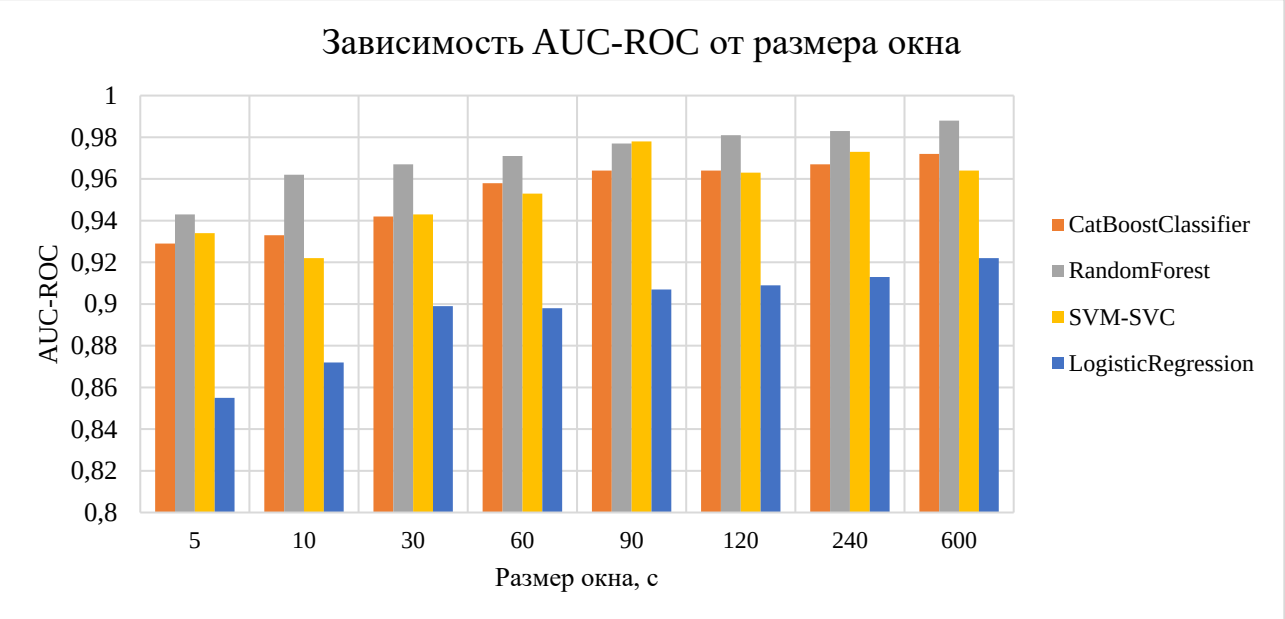

Рисунок 16 — Диаграмма зависимости значения метрики AUC-ROC от размера скользящего окна для модуля BT на этапе финального тестирования

На рисунке 17 представлена диаграмма, визуализирующая различия в средних значениях AUC-ROC для двух различных способов формирования признаков — с помощью стационарного и скользящего окна. Для каждого модуля алгоритмы показывали лучшие значения метрики при скользящем способе формирования окна.

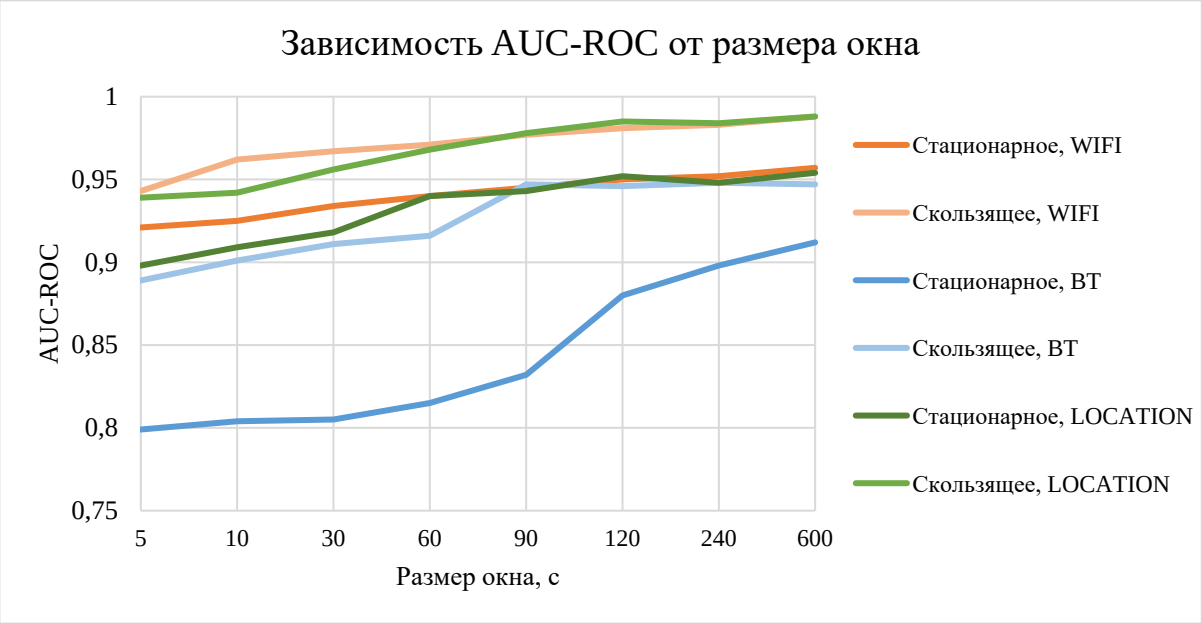

Рисунок 17 — График зависимости значения AUC-ROC от размера окна для модуля BT, WIFI и LOCATION на этапе финального тестирования RandomForestClassifier

На рисунке 18 представлена диаграмма, по которой можно сравнить значения EER для разных модулей. Лучшие показатели принадлежат модулю WIFI, а также модулю LOCATION, BT немного отстаёт от них.

В поставленной задачи классификации пользователей хорошие результаты показали все исследуемые алгоритмы, кроме логистической регрессии. Лучшие значения метрик наблюдались при работе RandomForestClassifier, независимо от модуля. Для него получены значения EER в промежутке между от 0.157 до 0.093 для модуля BT, от 0.102 до 0.029 для модуля WIFI и от 0.109 до 0.037 для модуля LOCATION на этапе финального тестирования для скользящих окон различного размера. На основе полученных результатов для дальнейшего исследования

метода аутентификации решено использовать алгоритм случайного леса (RandomForestClassifier).

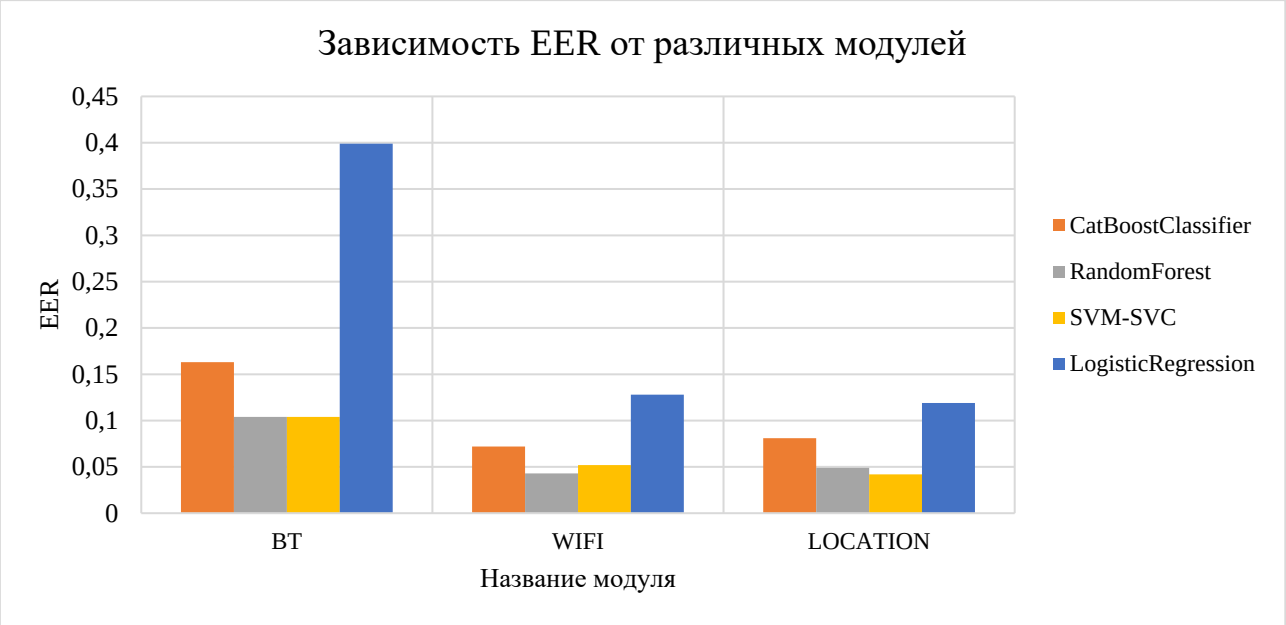

Рисунок 18 — Диаграмма зависимости значения EER от модуля при размере скользящего окна 120 секунд

Также было выяснено, что следует выбирать размер окна более 30-60 секунд, так как лучшие результаты были получены при использовании больших размеров окон, нежели меньших. Для скользящего окна в среднем значения метрик были больше, чем для стационарного, поэтому в дальнейшем исследовании решено использовать только скользящее окно.

Среди всех модулей лучшие значения метрик были получены при тестировании обученных алгоритмов на WIFI и LOCATION. Но отставание модуля BT по уровню EER не так велико и составляет от 0.05 до 0.25.

Следует также обратить внимание, что результаты, полученные на кросс-валидации, отличаются от результатов на финальном тестировании. Это связано с особенностью выбранной схемы кросс-валидации. А именно с тем, что на этапе финального тестирования проверка обученного алгоритма проводилась на выборке, состоящей из данных пользователя, которые отсутствовали в обучающей выборке.

7 Непрерывная аутентификация пользователей мобильных устройств

Непрерывная аутентификация пользователей позволяет путём оценки особенностей поведения пользовать постоянно контролировать доступ к тем или иным ресурсам защищаемой системы. Причём процесс аутентификации может быть неявным, то есть проходить в фоновом режиме, никак не препятствуя основной деятельности пользователя. В данной работе исследованы показатели качества метода непрерывной аутентификации на основе данных с нескольких модулей: WIFI, LOCATION и BT.

7.1 Показатели качества работы методов непрерывной аутентификации

Для оценки качества непрерывных систем аутентификации используются метрики ANGA (среднее число действий подлинного пользователя до момента, когда система выдаст оповещение о нелегальном проникновении) и ANIA (среднее число действий злоумышленника в системе до тех пор, пока не произойдёт блокировка в результате срабатывания системы аутентификации) [2].

Другими словами, ANGA отражает как долго система аутентификации не заблокирует подлинного пользователя, а ANIA — как долго сможет пользоваться системой злоумышленник. Следовательно, при разработке метода непрерывной аутентификации, необходимо стараться обеспечить максимально возможное значение ANGA при минимально возможном ANIA.

7.2 Модуль принятия окончательного решения DECIDER

В данной работе предложен метод непрерывной аутентификации пользователей мобильного устройства, основанный на данных с нескольких модулей.

Предполагается, что на основе поступающих с каждого модуля признаков, соответствующие модулям классификаторы формируют оценки. Финальный же вердикт, основанный на решениях всех классификаторов, предложено получать, комбинируя («смешивая») значения с помощью ещё одного модуля DECIDER,

обеспечивающего принятие финального решения о допуске пользователя к системе. Схема такого метода представлена рисунке 8.

Модуль DECIDER решено реализовать с помощью ещё одного классификатора, независимого от остальных, на вход которому будут поданы значения оценки вероятностей, сформированные ранее классификаторами для модулей WIFI, BT и LOCATION. Таким образом, признаками для нового классификатора являются оценки вероятностей отношения к классу подлинных пользователей. Всего таких признаков будет 3, столько же, сколько и модулей.

Для решения задачи классификации в модуле DECIDER решено использовать логистическую регрессию (LogisticRegression, реализация из библиотеки scikit-learn [67]). Такой выбор сделан на основе предположения о том, что для решаемой задачи подходит простая линейная модель, которая комбинирует решения классификаторов с некоторыми весами, коей и является логистическая регрессия. Fridman и другие применяют подобный способ в своей работе [10]. Однако, в [10] для получения финального решения используется подход на основе минимизации байесовского риска [75]. В логистической модели вероятность отношения объекта выборки к классу строится на основе вычисления линейной комбинации признаков. Вероятность отношения объекта к положительному классу вычисляется по формуле [76] (31):

$$P(C=1|X_1,X_2,\ldots,X_k)=\frac{1}{1+e^{-(\alpha+\sum_{i=1}^{k}\beta_i X_i)}}, \quad (31)$$

где k — число признаков, характеризующих объект выборки, X_i — значение i-го признака на соответствующем объекте выборки, α — свободный коэффициент, β_i — коэффициент при i-ом признаке.

7.3 Формирование тестовых наборов данных

Для проведения тестирования рассматриваемого метода были сформированы специальные выборки данных. Генерирование выборок было проведено следующим образом.

Для каждого пользователя были сформированы наборы элементарных событий продолжительностью в 15 минут в их естественном временном порядке. При этом момент начала каждого такого набора выбирался случайно с помощью генератора псевдослучайных чисел (ГПСЧ), реализованного в стандартной библиотеке языка программирования Python.

Для каждого пользователя поочерёдно были назначены нарушителями другие пользователи, после чего поток событий легального пользователя был конкатенирован с данными злоумышленника. Таким образом, была смоделирована ситуация, при которой в системе сначала 15 минут работает легальный пользователь, а затем с системой на протяжении 15 минут пытается взаимодействовать нарушитель.

Для каждой пары «легальный пользователь — нарушитель» было сформировано по 100 тестовых наборов элементарных событий. В общей сложности было получено 5600 таких наборов.

После чего для каждого отдельного набора были сформированы вектора признаков для подачи на вход классификаторам. При формировании признаков также было посчитано количество событий, входящих в соответствующее временное окно.

Для формирования выборок были разработаны сценарии на языке программирования Python, они представлены в приложении Д.

7.4 Тестирование метода непрерывной аутентификации

Для тестирования выбран алгоритм случайного леса (RandomForestClassifier), так как он продемонстрировал лучшие значения метрик на этапе тестирования моделей машинного обучения.

Также решено воспользоваться моделями, уже обученными ранее на полных выборках, из которых заведомо исключались данные условленного злоумышленника. Для каждой пары «легальный пользователь — нарушитель» выбирался соответствующий обученный алгоритм. Затем алгоритму подавались на вход вектора признаков, сформированные ранее для тестирования. Причём, такая процедура проводилась для каждой пары 100 раз, для каждого сгенерированного набора событий.

На основе полученных результатов будет обучен алгоритм линейной регрессии для смешивания результатов.

Предложенный подход позволяет оценить, как метрики ANGA и ANIA, так и EER, FAR, FRR и AUC-ROC.

ANGA и ANIA будут рассчитываться исходя из количества элементарных событий, участвовавших в формировании соответствующих признаков. Данные метрики необходимы для оценки качества работы метода аутентификации в своей непрерывной реализации.

Метрики AUC-ROC, EER, FAR и FRR будут рассчитываться на основе вердиктов классификатора для всего сгенерированного тестового набора в целом.

ЗАКЛЮЧЕНИЕ

В работе были проанализированы существующие подходы к аутентификации пользователей на основе поведенческой биометрии. Проведено их сравнение. Рассмотрены характеристики различных систем аутентификации на основе поведения пользователя.

На основе рассмотренных работ был предложен метод аутентификации по поведенческому профилю пользователя, основанный на данных, получаемых с помощью датчиков WiFi, Bluetooth и GPS.

Было разработано мобильное приложение под операционную систему Android для сбора данных пользователей от нескольких модулей: WIFI, BT, LOCATION.

В ходе обработки полученных от пользователей данных были сформированы признаки для модулей WIFI, BT и LOCATION, была проведена подготовка выборок для обучения и тестирования алгоритмов.

Для тестирования алгоритмов машинного обучения были разработаны схемы кросс-валидации и финального тестирования. Были протестированы алгоритмы классификации на сформированных выборках. Проведено сравнение результатов по значениям метрик accuracy, AUC-ROC и EER для каждого модуля. Также оценены различные способы формирования признаков на основе событий от каждого из модулей.

Лучшие значения EER были получены при тестировании алгоритма случайного леса. Для него получены значения в промежутке между от 0.157 до 0.093 для модуля BT, от 0.102 до 0.029 для модуля WIFI и от 0.109 до 0.037 для модуля LOCATION на этапе финального тестирования при формировании признаков с помощью скользящих окон различного размера. Результаты по модулям WIFI и LOCATION сопоставимы между собой. Для модуля BT значения EER в среднем оказывались на 0.05-0.25 больше, чем для WIFI и LOCATION. При этом значения метрик при более длительных окнах (120 с, 240 с), использованных при формировании признаков, оказывались в среднем выше,

чем на более коротких. Значения метрик, полученные на выборках, сформированных с помощью скользящего окна, в среднем были лучше, чем для стационарного окна.

На основе полученных результатов было принято решение использовать алгоритм случайного леса в следующих экспериментах. Также решено использовать скользящее окно при формировании признаков.

Для оценки качества работы предложенного метода аутентификации разработана схема тестирования, основанная на генерировании выборок, которые имитируют поток событий при вторжении нарушителя в защищаемую систему.

Для формирования финального вердикта на основе оценок вероятностей, формируемых классификаторами соответствующих модулей, решено было использовать логистическую регрессию в качестве алгоритма классификации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Афанасьев, А. А. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам [Текст] : учебное пособие для вузов / А.А. Афанасьев, Л.Т. Веденьев, А.А. Воронцов и др. — М. : Горячая линия — Телеком, 2012. — 550 с.

2 Mondal, S. Continuous authentication using behavioural biometrics [Текст] / S. Mondal, P. Bours // Collaborative European Research Conference (CERC'13). — 2013. — С. 130-140.

3 National Computer Security Center (US). A Guide to Understanding Identification and Authentication in Trusted Systems [Текст] // National Computer Security Center, 1991. — Т. 17.

4 Ometov, A. Multi-factor authentication: A survey [Текст] / A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, Y. Koucheryavy // Cryptography. — 2018. — Т. 2. — №. 1. — С. 1.

5 Sultana, M. Social behavioral biometrics: An emerging trend [Текст] / M. Sultana, P. P. Paul, M. Gavrilova // International Journal of Pattern Recognition and Artificial Intelligence. — 2015. — Т. 29. — №. 08. — С. 1556013.

6 Alzubaidi, A. Authentication of smartphone users using behavioral biometrics [Текст] / A. Alzubaidi, J. Kalita // IEEE Communications Surveys & Tutorials. — 2016. — Т. 18. — №. 3. — С. 1998-2026.

7 Burgbacher, U. A behavioral biometric challenge and response approach to user authentication on smartphones [Текст] / U. Burgbacher, M. Prätorius, K. Hinrichs // 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC). — IEEE, 2014. — С. 3328-3335.

8 Cao, K. Hacking mobile phones using 2D printed fingerprints [Текст] / K. Cao, A. K. Jain // Department of Computer Science and Engineering, Michigan State University. — 2016.

9 Maro, E. Bypass Biometric Lock Systems With Gelatin Artificial Fingerprint [Текст] / E. Maro, M. Kovalchuk // Proceedings of the 11th International Conference on Security of Information and Networks. — 2018. — С. 1-2.

10 Fridman, L. Active authentication on mobile devices via stylometry, application usage, web browsing, and GPS location [Текст] / L. Fridman, S. Weber, R. Greenstadt, M. Kam // IEEE Systems Journal. — 2016. — Т. 11. — №. 2. — С. 513-521.

11 Meng, W. Surveying the development of biometric user authentication on mobile phones [Текст] / W. Meng, D. S. Wong, S. Furnell, J. Zhou // IEEE Communications Surveys & Tutorials. — 2014. — Т. 17. — №. 3. — С. 1268-1293.

12 Trojahn, M. Toward mobile authentication with keystroke dynamics on mobile phones and tablets [Текст] / M. Trojahn, F. Ortmeier // 2013 27th International Conference on Advanced Information Networking and Applications Workshops. — IEEE, 2013. — С. 697-702.

13 Лебеденко, Ю. И. Биометрические системы безопасности [Текст] : учебное пособие / Ю. И. Лебеденко // М-во образования и науки РФ, Федеральное гос. бюджетное образовательное учреждение высш. проф. образования "Тульский гос. ун-т". — Тула : Изд-во ТулГУ, 2012. — 171 с.

14 Михайлов, А. А. Основные параметры биометрических систем [Текст] / А. А. Михайлов, А. А. Колосков, Ю. И. Дронов // Алгоритм безопасности. — 2015. — №. 5. — С. 58.

15 Kambourakis, G. Introducing touchstroke: keystroke-based authentication system for smartphones [Текст] / G. Kambourakis, D. Damopoulos, D. Papamartzivanos, E. Pavlidakis // Security and Communication Networks. — 2016. — Т. 9. — №. 6. — С. 542-554.

16 Draffin, B. Keysens: Passive user authentication through micro-behavior modeling of soft keyboard interaction [Текст] / B. Draffin, J. Zhu, J. Zhang // International Conference on Mobile Computing, Applications, and Services. — Springer, Cham, 2013. — С. 184-201.

17 Hall, M. The WEKA data mining software: an update [Текст] / M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten // ACM SIGKDD explorations newsletter. — 2009. — Т. 11. — №. 1. — С. 10-18.

18 Quinlan, J. R. Bagging, boosting, and C4.5 [Текст] / J. R. Quinlan // AAAI/IAAI, Vol. 1. — 1996. — С. 725-730.

19 Cleary, J. G. K*: An instance-based learner using an entropic distance measure [Текст] / J. G. Cleary, L. E. Trigg // Machine Learning Proceedings 1995. — Morgan Kaufmann, 1995. — С. 108-114.

20 Pal, S. K. Multilayer perceptron, fuzzy sets, and classification [Текст] / S. K. Pal, S. Mitra // IEEE Transactions on Neural Networks. — 1992. — Т. 3. — №. 5. — С. 683-697.

21 Hwang, Y. S. An efficient method to construct a radial basis function neural network classifier [Текст] / Y. S. Hwang, S. Y. Bang // Neural networks. — 1997. — Т. 10. — №. 8. — С. 1495-1503.

22 Friedman, N. Bayesian network classifiers [Текст] / N. Friedman, D. Geiger, M. Goldszmidt // Machine learning. — 1997. — Т. 29. — №. 2-3. — С. 131-163.

23 Rish, I. An empirical study of the naive Bayes classifier [Текст] / I. Rish // IJCAI 2001 workshop on empirical methods in artificial intelligence. — 2001. — Т. 3. — №. 22. — С. 41-46.

24 Pal, M. Random forest classifier for remote sensing classification [Текст] / M. Pal // International Journal of Remote Sensing. — 2005. — Т. 26. — №. 1. — С. 217-222.

25 Denoeux, T. A k-nearest neighbor classification rule based on Dempster-Shafer theory [Текст] / T. Denoeux // Classic works of the Dempster-Shafer theory of belief functions. — Springer, Berlin, Heidelberg, 2008. — С. 737-760.

26 Bishop, C.M. Neural Networks for Pattern Recognition [Текст] / C.M. Bishop. — New York : Oxford University Press, Inc., 1995. — 482 с.

27 Nickel, C. Classification of acceleration data for biometric gait recognition on mobile devices [Текст] / C. Nickel, H. Brandt, C. Busch // BIOSIG 2011 — Proceedings of the Biometrics Special Interest Group. — 2011.

28 Derawi, M. O. Unobtrusive user-authentication on mobile phones using biometric gait recognition [Текст] / M. O. Derawi, C. Nickel, P. Bours, C. Busch // 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. — IEEE, 2010. — С. 306-311.

29 Официальный сайт компании The MathWorks, Inc. [Электронный ресурс] // The MathWorks, Inc. — Режим доступа: https://www.mathworks.com/help/supportpkg/android/ref/accelerometer.html (Дата обращения: 10.05.2020).

30 Müller, M. Information retrieval for music and motion [Текст] / M. Müller. — Heidelberg : Springer, 2007. — Т. 2. — С. 59.

31 Nickel, C. Authentication of smartphone users based on the way they walk using k-nn algorithm [Текст] / C. Nickel, T. Wirtl, C. Busch // 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. — IEEE, 2012. — С. 16-20.

32 Rabiner, L. R. A tutorial on hidden Markov models and selected applications in speech recognition [Текст] / L. R. Rabiner // Proceedings of the IEEE. — 1989. — Т. 77. — №. 2. — С. 257-286.

33 Nickel, C. Classifying accelerometer data via hidden markov models to authenticate people by the way they walk [Текст] / C. Nickel, C. Busch // IEEE Aerospace and Electronic Systems Magazine. — 2013. — Т. 28. — №. 10. — С. 29-35.

34 Воронцов, К. В. Лекции по логическим алгоритмам классификации [Электронный ресурс] / К. В. Воронцов // Режим доступа: http://www. ccas. ru/voron/teaching. html/LogicAlgs.pdf. — 2007.

35 Patel, V. M. Continuous user authentication on mobile devices: Recent progress and remaining challenges [Текст] / V. M. Patel, R. Chellappa, D. Chandra, B.

Barbello // IEEE Signal Processing Magazine. — 2016. — Т. 33. — №. 4. — С. 49-61.

36 Sim, T. Continuous verification using multimodal biometrics [Текст] / T. Sim, S. Zhang, R. Janakiraman, S. Kumar // IEEE transactions on pattern analysis and machine intelligence. — 2007. — Т. 29. — №. 4. — С. 687-700.

37 Brocardo, M. L. Authorship verification for short messages using stylometry [Текст] / M. L. Brocardo, I. Traore, S. Saad, I. Woungang // 2013 International Conference on Computer, Information and Telecommunication Systems (CITS). — IEEE, 2013. — С. 1-6.

38 Hartley, H. O. Maximum-likelihood estimation for the mixed analysis of variance model [Текст] / H. O. Hartley, J. N. K. Rao // Biometrika. — 1967. — Т. 54. — №. 1-2 — С. 93-108.

39 Abe, S. Support vector machines for pattern classification [Текст] / S. Abe. — London : Springer, 2005. — Т. 2. — С. 44.

40 Tenney, R. R. Detection with distributed sensors [Текст] / R. R. Tenney, N. R. Sandell // IEEE Transactions on Aerospace and Electronic systems. — 1981. — №. 4. — С. 501-510.

41 Saevanee, N. Text-based active authentication for mobile devices [Текст] / H. Saevanee, N. Clarke, S. Furnell, V. Biscione // IFIP International Information Security Conference. — Springer, Berlin, Heidelberg, 2014. — С. 99-112.

42 Li, F. Active authentication for mobile devices utilising behaviour profiling [Текст] / F. Li, N. Clarke, M. Papadaki, P. Dowland // International journal of information security. — 2014. — Т. 13. — №. 3. — С. 229-244.

43 Pentland, A. Inferring social network structure using mobile phone data [Текст] / A. Pentland, N. Eagle, D. Lazer // Proceedings of the National Academy of Sciences (PNAS). — 2009. — Т. 106. — №. 36. — С. 15274-15278.

44 Shetty, J. The Enron email dataset database schema and brief statistical report [Текст] / J. Shetty, J. Adibi // Information sciences institute technical report, University of Southern California. — 2004. — Т. 4. — №. 1. — С. 120-128.

45 Android API reference [Электронный ресурс] // Android Developers. — Режим доступа: https://developer.android.com/reference (Дата обращения: 21.12.2020).

46 GitHub repository [Электронный ресурс] // GitHub, Inc. — Режим доступа: https://github.com/yerseg/behaviour_profiling_app (Дата обращения: 21.12.2020).

47 Pandas documentation [Электронный ресурс] // The pandas development team. — Режим доступа: https://pandas.pydata.org/docs/ (Дата обращения: 21.12.2020).

48 NumPy Documentation [Электронный ресурс] // The SciPy community. — Режим доступа: https://numpy.org/doc/stable/index.html (Дата обращения: 21.12.2020).

49 GeoPy’s documentation [Электронный ресурс] // GeoPy Contributors Revision. — Режим доступа: https://geopy.readthedocs.io/en/stable/ (Дата обращения: 21.12.2020).

50 SciPy Documentation [Электронный ресурс] // SciPy developers. — Режим доступа: https://www.scipy.org/docs.html (Дата обращения: 21.12.2020).

51 Matplotlib overview [Электронный ресурс] // The Matplotlib development team. — Режим доступа: https://matplotlib.org/contents.html (Дата обращения: 21.12.2020).

52 Calhoun, P. Control and provisioning of wireless access points (CAPWAP) protocol binding for IEEE 802.11 [Текст] / P. Calhoun, M. Montemurro, D. Stanley // IETF RFC5416. — 2009.

53 Wang, X. Using Jaccard Distance Measure for Unsupervised Activity Recognition with Smartphone Accelerometers [Текст] / X. Wang, Y. Lu, D. Wang, L. Liu, H. Zhou // Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data. — Springer, Cham, 2017. — С. 74-83.

54 Doane, D. P. Measuring skewness: a forgotten statistic? [Текст] / D. P. Doane, L. E. Seward // Journal of statistics education. — 2011. — Т. 19. — №. 2.

55 Karney, C. F. F. Algorithms for geodesics [Текст] / C. F. F. Karney // Journal of Geodesy. — 2013. — Т. 87. — №. 1. — С. 43-55.

56 Рашка, С. Python и машинное обучение [Текст] : Пер. с англ. / С. Рашка. — М. : ДМК Пресс, 2017. — 418 с.

57 Jain, A. Score normalization in multimodal biometric systems [Текст] / A. Jain, K. Nandakumar, A. Ross // Pattern recognition. — 2005. — Т. 38. — №. 12. — С. 2270-2285.

58 Patro, S. Normalization: A preprocessing stage [Текст] / S. Patro, K. K. Sahu // arXiv preprint arXiv:1503.06462. — 2015.

59 Cousineau, D., Chartier S. Outliers detection and treatment: a review [Текст] / D. Cousineau, S. Chartier // International Journal of Psychological Research. — 2010. — Т. 3. — №. 1. — С. 58-67.

60 Benesty, J. Pearson correlation coefficient [Текст] / J. Benesty, J. Chen, Y. Huang, I. Cohen // Noise reduction in speech processing. — Springer, Berlin, Heidelberg, 2009. — С. 1-4.

61 Clancey, W. J. Classification problem solving [Текст] / W. J. Clancey — Stanford, CA : Stanford University, 1984. — С. 49-55.

62 Hu, J. A k-nearest neighbor approach for user authentication through biometric keystroke dynamics [Текст] / J. Hu, D. Gingrich, A. Sentosa // 2008 IEEE International Conference on Communications. — IEEE, 2008. — С. 1556-1560.

63 Chandola, V. Anomaly detection: A survey [Текст] / V. Chandola, A. Banerjee, V. Kumar // ACM computing surveys (CSUR). — 2009. — Т. 41. — №. 3. — С. 1-58.

64 Natekin, A. Gradient boosting machines, a tutorial [Текст] / A. Natekin, A. Knoll // Frontiers in neurorobotics. — 2013. — Т. 7. — С. 21.

65 Overview of CatBoost [Электронный ресурс] // Yandex, LLC. — Режим доступа: https://catboost.ai/docs/concepts/about.html (Дата обращения: 21.12.2020).

66 Mathur, A. Multiclass and binary SVM classification: Implications for training and classification users [Текст] / A. Mathur, G. M. Foody // IEEE Geoscience and remote sensing letters. — 2008. — Т. 5. — №. 2. — С. 241-245.

67 Scikit-learn User Guide [Электронный ресурс] // Scikit-learn developers. — Режим доступа: https://scikit-learn.org/stable/user_guide.html (Дата обращения: 21.12.2020).

68 Biau, G. A random forest guided tour [Текст] / G. Biau, E. Scornet // Test. — 2016. — Т. 25. — №. 2. — С. 197-227.

69 Buriro, A. Hold and Sign: a novel behavioral biometrics for smartphone user authentication [Текст] / A. Buriro, B. Crispo, F. Delfrari // Security and Privacy Workshops (SPW), 2016 IEEE. — 2016 — С. 276-285.

70 Sebastien, M. Handbook of Biometric Anti-Spoofing: Trusted Biometrics under Spoofing Attacks [Текст] / M. Sebastien, M. S. Nixon, S. Z. Li. — Cham : Springer, 2014. — 281 с.

71 Narkhede, S. Understanding AUC-ROC Curve [Текст] / S. Narkhede // Towards Data Science. — 2018. — Т. 26.

72 Brzezinski, D. Prequential AUC: properties of the area under the ROC curve for data streams with concept drift [Текст] / D. Brzezinski, J. Stefanowski // Knowledge and Information Systems. — 2017. — Т. 52. — №. 2. — С. 531-562.

73 Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection [Текст] / R. Kohavi // 14th International Joint Conference on Artificial Intelligence, Palais de Congres Montreal, Quebec, Canada. — 1995. — С. 1137-1145.

74 Yap, B. W. An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets [Текст] / B. W. Yap, K. A. Rani, H. A. A. Rahman, S. Fong, Z. Khairudin, N. N. Abdullah // Proceedings of the first international conference on advanced data and information engineering (DaEng-2013). — Springer, Singapore, 2014. — С. 13-22.

75 Czado, C. Bayesian risk analysis [Текст] / C. Czado, E. C. Brechmann // Risk-A Multidisciplinary Introduction. — Springer, Cham, 2014. — С. 207-240.

76 Kleinbaum, D. G. Logistic regression [Текст] / D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein — New York : Springer-Verlag, 2002.

Приложение А

Исходный код приложения для сбора данных

В данном приложении приведён исходный код приложения для сбора данных, реализованное на языке программирования Java. Фрагменты исходного кода разбиты на файлы формата JAVA.

MainActivity.java

```
package com.yerseg.profiler;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.NotificationManager;
import android.bluetooth.BluetoothAdapter;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.location.LocationManager;
import android.net.Uri;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.os.Process;
import android.provider.Settings;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Toolbar;

import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;
import androidx.fragment.app.FragmentActivity;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.io.File;
import java.util.LinkedList;
import java.util.List;
import java.util.Locale;

public class MainActivity extends FragmentActivity {

    private final static int PERMISSIONS_REQUEST_ID = 1001;

    public final static String ACTION_LOCATION_SCANNING_SETTINGS =
"android.settings.LOCATION_SCANNING_SETTINGS";

    private final static String LOCATION_SOURCE_SETTINGS_SHOWN =
"com.yerseg.profiler.LOCATION_SOURCE_SETTINGS_SHOWN";
    private final static String APPLICATION_DETAILS_SETTINGS_SHOWN =
"com.yerseg.profiler.APPLICATION_DETAILS_SETTINGS_SHOWN";
    private final static String LOCATION_SCANNING_SETTINGS_SHOWN =
"com.yerseg.profiler.LOCATION_SCANNING_SETTINGS_SHOWN";
```

```
    private final static String REQUEST_IGNORE_BATTERY_OPTIMIZATIONS_SHOWN =
"com.yerseg.profiler.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS_SHOWN";

    Intent mProfilingServiceIntent;
    boolean mIsPermissionsGranted = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setActionBar(toolbar);

        Button startButton = findViewById(R.id.profilingStartButton);
        Button stopButton = findViewById(R.id.profilingStopButton);

        startButton.setOnClickListener(v -> {
            if (!mIsPermissionsGranted) {
                showLongToast("Grant permission please!");
                requestPermissions();
                return;
            }

            // Common location settings
            if (shouldShowSettingsActivity(LOCATION_SOURCE_SETTINGS_SHOWN)) {
                Intent intent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                showLongToast("Turn on all location services please!");
                startActivityForResult(intent, 1);
                markSettingsActivityShown(LOCATION_SOURCE_SETTINGS_SHOWN);
                return;
            }

            if
(shouldShowSettingsActivity(REQUEST_IGNORE_BATTERY_OPTIMIZATIONS_SHOWN)) {
                @SuppressLint("BatteryLife")
                Intent intent = new
Intent(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
                intent.setData(Uri.parse("package:" +
getApplicationContext().getPackageName()));
                showLongToast("Allow the app to ignore battery optimizations
please!");
                startActivityForResult(intent, 1);

markSettingsActivityShown(REQUEST_IGNORE_BATTERY_OPTIMIZATIONS_SHOWN);
                return;
            }

            // App settings
            if (shouldShowSettingsActivity(APPLICATION_DETAILS_SETTINGS_SHOWN))
{
                Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                intent.setData(Uri.parse("package:" +
getApplicationContext().getPackageName()));
                showLongToast("Turn off battery optimizations for app please!");
                startActivityForResult(intent, 1);
                markSettingsActivityShown(APPLICATION_DETAILS_SETTINGS_SHOWN);
                return;
            }
```

```
            // Two switch
            if (shouldShowSettingsActivity(LOCATION_SCANNING_SETTINGS_SHOWN)) {
                Intent intent = new Intent(ACTION_LOCATION_SCANNING_SETTINGS);
                showLongToast("Turn on all switches please!");
                startActivityForResult(intent, 1);
                markSettingsActivityShown(LOCATION_SCANNING_SETTINGS_SHOWN);
                return;
            }

            WifiManager wifiManager = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
            if (!wifiManager.isWifiEnabled()) {
                showLongToast("Turn on WiFi please!");
                return;
            }

            BluetoothAdapter mBluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
            if (mBluetoothAdapter == null || !mBluetoothAdapter.isEnabled()) {
                showLongToast("Turn on Bluetooth please!");
                return;
            }

            LocationManager locationManager = (LocationManager)
getApplicationContext().getSystemService(Context.LOCATION_SERVICE);
            if (!locationManager.isLocationEnabled() ||

!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) ||

!locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {
                showLongToast("Turn on location services please!");
                return;
            }

            startService();
            // Can crash when click on stop button before service completely
start
            startButton.setEnabled(false);
            stopButton.setEnabled(true);
        });

        stopButton.setOnClickListener(v -> {
            stopService();
            // Can crash when click on start button before service completely
stop
            stopButton.setEnabled(false);
            startButton.setEnabled(true);
        });

        TextView textView = findViewById(R.id.textInstruction);
        textView.setVisibility(View.VISIBLE);

        ProgressBar sendZipProgressBar = findViewById(R.id.sendZipProgressBar);

        FloatingActionButton emailSendButton =
findViewById(R.id.SendDataByEmailButton);
        emailSendButton.setOnClickListener(v -> {
            sendZipProgressBar.setVisibility(View.VISIBLE);
            emailSendButton.setEnabled(false);
            new Thread(() -> {
                onSendButtonClick();
                runOnUiThread(() -> {
```

```
                    sendZipProgressBar.setVisibility(View.GONE);
                    emailSendButton.setEnabled(true);
                });
            }).start();
        });
    }

    @Override
    protected void onStart() {
        super.onStart();

        Button startButton = findViewById(R.id.profilingStartButton);
        startButton.setEnabled(!isProfilingServiceRunning());

        Button stopButton = findViewById(R.id.profilingStopButton);
        stopButton.setEnabled(isProfilingServiceRunning());

        requestPermissions();
    }

    private void startService() {
        Intent mProfilingServiceIntent = new Intent(this,
ProfilingService.class).putExtra("inputExtra", "ServiceControl");
        ContextCompat.startForegroundService(this, mProfilingServiceIntent);
    }

    private void stopService() {
        if (mProfilingServiceIntent == null)
            mProfilingServiceIntent = new Intent(this,
ProfilingService.class).putExtra("inputExtra", "ServiceControl");

        stopService(mProfilingServiceIntent);
    }

    private boolean isProfilingServiceRunning() {
        boolean isRunning = false;
        synchronized (this) {
            isRunning = ProfilingService.isRunning;
        }
        return isRunning;
    }

    void requestPermissions() {
        requestPermissions(new String[]{
                Manifest.permission.ACCESS_COARSE_LOCATION,
                Manifest.permission.ACCESS_FINE_LOCATION,
                Manifest.permission.ACCESS_WIFI_STATE,
                Manifest.permission.BIND_DEVICE_ADMIN,
                Manifest.permission.BLUETOOTH,
                Manifest.permission.BLUETOOTH_ADMIN,
                Manifest.permission.CHANGE_WIFI_STATE,
                Manifest.permission.FOREGROUND_SERVICE,
                Manifest.permission.GET_ACCOUNTS,
                Manifest.permission.GET_ACCOUNTS_PRIVILEGED,
                Manifest.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS,
                Manifest.permission.RECEIVE_SMS,
                Manifest.permission.READ_CONTACTS,
                Manifest.permission.UPDATE_DEVICE_STATS
        }, PERMISSIONS_REQUEST_ID);
    }

    @Override
```

```
    public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
        mIsPermissionsGranted = requestCode == PERMISSIONS_REQUEST_ID &&
grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED;
    }

    private boolean shouldShowSettingsActivity(String preferencesKeyName) {
        SharedPreferences prefs =
getSharedPreferences(getApplicationContext().getPackageName(),
Context.MODE_PRIVATE);
        return !prefs.getBoolean(preferencesKeyName, false);
    }

    private void markSettingsActivityShown(String preferencesKeyName) {
        SharedPreferences prefs =
getSharedPreferences(getApplicationContext().getPackageName(),
Context.MODE_PRIVATE);
        prefs.edit().putBoolean(preferencesKeyName, true).apply();
    }

    private void showLongToast(String text) {
        runOnUiThread(() -> Toast.makeText(getApplicationContext(), text,
Toast.LENGTH_LONG).show());
    }

    private void moveDataFilesToTempDirectory(String[] dataFilesNames) {
        for (String fileName : dataFilesNames) {
            try {
                Utils.moveFile(new
File(Utils.getProfilingFilesDir(getApplicationContext()), fileName), new
File(Utils.getTempDataFilesDir(getApplicationContext()), fileName));
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }

    private boolean deleteTempFiles() {
        boolean isAllFilesDeleted = true;
        File tempDir = Utils.getTempDataFilesDir(getApplicationContext());

        if (tempDir.exists()) {
            File[] tempFiles = tempDir.listFiles();
            if (tempFiles != null) {
                for (File tempFile : tempFiles) {
                    if (tempFile.exists()) {
                        if (!Utils.deleteFile(tempFile)) {
                            isAllFilesDeleted = false;
                        }
                    }
                }
            }
        }

        return !isAllFilesDeleted;
    }

    private void onSendButtonClick() {
        Log.d("Profiler [MainActivity]", String.format(Locale.getDefault(),
"\t%d\tonSendButtonClick()", Process.myTid()));

        try {
```

```
            if (deleteTempFiles()) {
                Thread.sleep(300);
                if (deleteTempFiles()) {
                    Log.d("Profiler [MainActivity]", String.format(
                            Locale.getDefault(), "\t%d\tonSendButtonClick(),
Msg: \"%s\"", Process.myTid(), "Temp directory is not clean!"));
                }
            }

            File tempDir = Utils.getTempDataFilesDir(getApplicationContext());

            try {
                moveDataFilesToTempDirectory(ProfilingService.STAT_FILE_NAMES);
            } catch (Exception e) {
                e.printStackTrace();
            }

            List<File> filesList = new LinkedList<>();

            for (String fileName : ProfilingService.STAT_FILE_NAMES) {
                File file = new File(tempDir, fileName);
                if (file.exists())
                    filesList.add(file);
            }

            File zip = Utils.createZip(filesList, tempDir);

            if (zip.exists()) {
                Intent sendStatsIntent = new Intent(Intent.ACTION_SEND);

                String[] to = {"cergei.kazmin@gmail.com"};
                Uri contentUri =
FileProvider.getUriForFile(getApplicationContext(), "com.yerseg.profiler", zip);

                sendStatsIntent.setType("application/zip");
                sendStatsIntent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
                sendStatsIntent.putExtra(Intent.EXTRA_STREAM, contentUri);
                sendStatsIntent.putExtra(Intent.EXTRA_EMAIL, to);
                sendStatsIntent.putExtra(Intent.EXTRA_SUBJECT, "[IMP] Profiling
stats");
                sendStatsIntent.putExtra(Intent.EXTRA_TEXT, "Sending profiling
stats");
                sendStatsIntent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

                Intent chooser = Intent.createChooser(sendStatsIntent, "Send
stats");
                List<ResolveInfo> resolveInfoList =
this.getPackageManager().queryIntentActivities(chooser,
PackageManager.MATCH_DEFAULT_ONLY);
                for (ResolveInfo resolveInfo : resolveInfoList) {
                    String packageName = resolveInfo.activityInfo.packageName;
                    this.grantUriPermission(packageName, contentUri,
Intent.FLAG_GRANT_READ_URI_PERMISSION);
                }

                NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.cancel(ProfilingService.REMINDER_NOTIFICATION_ID);

                startActivity(chooser);
            }
```

```
        } catch (Exception ex) {
            showLongToast("ERROR! Sending failed! Try again!");
            ex.printStackTrace();
        }
    }
}
```

ProfilingService.java

```
package com.yerseg.profiler;

import android.Manifest;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.Service;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanCallback;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.Location;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Handler;
import android.os.HandlerThread;
import android.os.IBinder;
import android.os.Looper;
import android.os.Process;
import android.util.Log;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.app.NotificationCompat;
import androidx.core.content.ContextCompat;
import androidx.work.ExistingPeriodicWorkPolicy;
import androidx.work.PeriodicWorkRequest;
import androidx.work.WorkManager;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;

import java.time.Duration;
import java.util.List;
import java.util.Locale;
import java.util.UUID;

public class ProfilingService extends Service {

    public static final String NOTIFICATION_CHANNEL_ID =
"com.yerseg.profiler.ProfilingService";
```

```
    public static final String PUSH_REMINDER_NOTIFICATION_WORK_TAG =
"com.yerseg.profiler.REMINDER_NOTIFICATION_WORK";

    public static final int SERVICE_NOTIFICATION_ID = 1;
    public static final int REMINDER_NOTIFICATION_ID = 2;

    public static final int WIFI_STATS_UPDATE_FREQ = 5000;
    public static final int BLUETOOTH_STATS_UPDATE_FREQ = 5000;
    public static final int LOCATION_STATS_UPDATE_FREQ = 5000;

    public static final String PROFILING_STATS_DIRECTORY_NAME = "ProfilingData";
    public static final String PROFILING_STATS_TEMP_DIRECTORY_NAME =
"ProfilingDataTemp";

    public static final String BLUETOOTH_STATS_FILE_NAME = "bt.data";
    public static final String LOCATION_STATS_FILE_NAME = "location.data";
    public static final String WIFI_STATS_FILE_NAME = "wifi.data";
    public static final String BROADCASTS_STATS_FILE_NAME = "broadcasts.data";

    public static final String[] STAT_FILE_NAMES = {
            BLUETOOTH_STATS_FILE_NAME,
            LOCATION_STATS_FILE_NAME,
            WIFI_STATS_FILE_NAME,
            BROADCASTS_STATS_FILE_NAME
    };

    public static boolean isRunning = false;
    public static boolean isStopping = true;

    public static boolean isBtLeProfilingStopped = false;

    private HandlerThread mLocationProfilingThread;
    private HandlerThread mWifiProfilingThread;
    private HandlerThread mBluetoothProfilingThread;

    private Handler mLocationProfilingThreadHandler;
    private Handler mWifiProfilingThreadHandler;
    private Handler mBluetoothProfilingThreadHandler;

    private BroadcastReceiver mWifiScanReceiver;
    private BroadcastReceiver mBluetoothBroadcastReceiver;
    private BroadcastReceiver mAnyBroadcastReceiver;

    private LocationCallback mLocationCallback;

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d("Profiler [Service]", String.format(Locale.getDefault(),
"\t%d\tonCreate()", Process.myTid()));

        synchronized (this) {
            isRunning = true;
        }

        createNotificationChannel();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        super.onStartCommand(intent, flags, startId);
```

```
        Log.d("Profiler [Service]", String.format(Locale.getDefault(),
"\t%d\tonStartCommand()", Process.myTid()));

        startLocationTracking();
        startWifiTracking();
        startBluetoothTracking();
        startAnyBroadcastsTracking();

        PeriodicWorkRequest notifyWorkRequest = new
PeriodicWorkRequest.Builder(ReminderNotificationPeriodicWorker.class,
Duration.ofHours(3)).setInitialDelay(Duration.ofMinutes(5)).build();

WorkManager.getInstance(getApplicationContext()).enqueueUniquePeriodicWork(PUSH_
REMINDER_NOTIFICATION_WORK_TAG, ExistingPeriodicWorkPolicy.REPLACE,
notifyWorkRequest);

        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();

        synchronized (this) {
            isStopping = true;
        }

        Log.d("Profiler [Service]", String.format(Locale.getDefault(),
"\t%d\tonDestroy()", Process.myTid()));

        stopLocationTracking();
        stopWifiTracking();
        stopBluetoothTracking();
        stopAnyBroadcastsTracking();


WorkManager.getInstance(getApplicationContext()).cancelUniqueWork(PUSH_REMINDER_
NOTIFICATION_WORK_TAG);

        NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.cancel(SERVICE_NOTIFICATION_ID);
        notificationManager.cancel(REMINDER_NOTIFICATION_ID);

        synchronized (this) {
            isRunning = false;
            isStopping = false;
        }
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        Log.d("Profiler [Service]", String.format(Locale.getDefault(),
"\t%d\tonBind()", Process.myTid()));
        return null;
    }

    private void createNotificationChannel() {
        NotificationChannel notificationChannel = new NotificationChannel(
                NOTIFICATION_CHANNEL_ID,
                "Profiling Service",
```

```
                NotificationManager.IMPORTANCE_HIGH);

        notificationChannel.setLightColor(Color.BLUE);

notificationChannel.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);

        NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.createNotificationChannel(notificationChannel);

        NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(this, NOTIFICATION_CHANNEL_ID);
        Notification notification = notificationBuilder
                .setSmallIcon(R.drawable.ic_launcher_2_foreground)
                .setOngoing(true)
                .setContentTitle("Profiler")
                .setContentText("Profiling service is running and collecting
statistics")
                .setPriority(NotificationManager.IMPORTANCE_MAX)
                .setCategory(Notification.CATEGORY_SERVICE)
                .build();

        startForeground(SERVICE_NOTIFICATION_ID, notification);
    }

    private void startLocationTracking() {
        LocationRequest locationRequest = LocationRequest.create();
        locationRequest.setInterval(LOCATION_STATS_UPDATE_FREQ);
        locationRequest.setFastestInterval(LOCATION_STATS_UPDATE_FREQ / 10);
        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

        FusedLocationProviderClient fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
        int permission = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION);
        if (permission == PackageManager.PERMISSION_GRANTED) {
            mLocationCallback = new LocationCallback() {
                @Override
                public void onLocationResult(@NonNull LocationResult result) {
                    Log.d("Profiler [LocationStat]",
String.format(Locale.getDefault(), "\t%d\tonLocationResult()",
Process.myTid()));

                    try {
                        Location location = result.getLastLocation();
                        String locationStats =
String.format(Locale.getDefault(), "%s;%f;%f;%f;%f\n",
                                Utils.GetTimeStamp(System.currentTimeMillis()),
                                location.getAccuracy(),
                                location.getAltitude(),
                                location.getLatitude(),
                                location.getLongitude()
                        );


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
LOCATION_STATS_FILE_NAME, locationStats);
                    } catch (Exception ex) {
                        ex.printStackTrace();
                    }
                }
            };
```

```
            mLocationProfilingThread = new
HandlerThread("LocationProfilingThread", Process.THREAD_PRIORITY_FOREGROUND);
            mLocationProfilingThread.start();

            Looper looper = mLocationProfilingThread.getLooper();
            mLocationProfilingThreadHandler = new Handler(looper);

            fusedLocationProviderClient.requestLocationUpdates(locationRequest,
mLocationCallback, looper);
        }
    }

    private void startWifiTracking() {
        mWifiScanReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context c, Intent intent) {
                if (intent.getBooleanExtra(WifiManager.EXTRA_RESULTS_UPDATED,
false)) {
                    new Thread(() -> {
                        try {
                            final WifiManager wifiManager = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
                            List<ScanResult> scanResults =
wifiManager.getScanResults();

                            String statResponseId =
UUID.randomUUID().toString();
                            String timestamp =
Utils.GetTimeStamp(System.currentTimeMillis());

                            for (ScanResult result : scanResults) {
                                String wifiStats =
String.format(Locale.getDefault(), "%s;%s;%s;%d;%d;%d\n",
                                        timestamp,
                                        statResponseId,
                                        result.BSSID,
                                        result.channelWidth,
                                        result.frequency,
                                        result.level);


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
WIFI_STATS_FILE_NAME, wifiStats);
                            }
                        } catch (Exception ex) {
                            ex.printStackTrace();
                        }
                    }).start();
                }
            }
        };

        IntentFilter intentFilter = new
IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);
        registerReceiver(mWifiScanReceiver, intentFilter);

        mWifiProfilingThread = new HandlerThread("WifiProfilingThread",
Process.THREAD_PRIORITY_FOREGROUND);
        mWifiProfilingThread.start();

        Looper looper = mWifiProfilingThread.getLooper();
```

```
        mWifiProfilingThreadHandler = new Handler(looper);

        mWifiProfilingThreadHandler.post(new Runnable() {
            @Override
            public void run() {
                try {
                    final WifiManager wifiManager = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
                    //noinspection deprecation
                    wifiManager.startScan();

                    final String timestamp =
Utils.GetTimeStamp(System.currentTimeMillis());
                    final WifiInfo currentInfo =
wifiManager.getConnectionInfo();

                    final String connectionInfo =
String.format(Locale.getDefault(), "%s;CONN;%s;%d;%d;%d;%d;%d;%s\n",
                            timestamp,
                            currentInfo.getBSSID(),
                            currentInfo.getFrequency(),
                            currentInfo.getIpAddress(),
                            currentInfo.getLinkSpeed(),
                            currentInfo.getNetworkId(),
                            currentInfo.getRssi(),
                            currentInfo.getSSID());


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
ProfilingService.WIFI_STATS_FILE_NAME, connectionInfo);
                } catch (Exception ex) {
                    ex.printStackTrace();
                } finally {
                    mWifiProfilingThreadHandler.postDelayed(this,
WIFI_STATS_UPDATE_FREQ);
                }
            }
        });
    }

    private void startBluetoothTracking() {
        mBluetoothBroadcastReceiver = new BroadcastReceiver() {
            public void onReceive(Context context, Intent intent) {
                new Thread(() -> {
                    try {
                        BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

                        if (device != null) {
                            String bluetoothStats =
String.format(Locale.getDefault(), "%s;%s;%s;%d;%d;%d;%d\n",

Utils.GetTimeStamp(System.currentTimeMillis()),
                                    intent.getAction(),
                                    device.getAddress(),

device.getBluetoothClass().getMajorDeviceClass(),
                                    device.getBluetoothClass().getDeviceClass(),
                                    device.getBondState(),
                                    device.getType());
```

```
Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
BLUETOOTH_STATS_FILE_NAME, bluetoothStats);
                        }
                    } catch (Exception ex) {
                        ex.printStackTrace();
                    }
                }).start();
            }
        };

        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);
        intentFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECT_REQUESTED);
        intentFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
        intentFilter.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
        intentFilter.addAction(BluetoothDevice.ACTION_CLASS_CHANGED);
        intentFilter.addAction(BluetoothDevice.ACTION_FOUND);
        intentFilter.addAction(BluetoothDevice.ACTION_NAME_CHANGED);
        intentFilter.addAction(BluetoothDevice.ACTION_PAIRING_REQUEST);
        intentFilter.addAction(BluetoothDevice.ACTION_UUID);

        registerReceiver(mBluetoothBroadcastReceiver, intentFilter);

        synchronized (this) {
            isBtLeProfilingStopped = false;
        }

        mBluetoothProfilingThread = new
HandlerThread("BluetoothProfilingThread", Process.THREAD_PRIORITY_FOREGROUND);
        mBluetoothProfilingThread.start();

        Looper looper = mBluetoothProfilingThread.getLooper();
        mBluetoothProfilingThreadHandler = new Handler(looper);

        mBluetoothProfilingThreadHandler.post(new Runnable() {
            @Override
            public void run() {
                try {
                    final BluetoothAdapter bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();

                    if (bluetoothAdapter != null) {
                        if (!bluetoothAdapter.isDiscovering())
                            bluetoothAdapter.startDiscovery();

                        final BluetoothLeScanner btScanner =
bluetoothAdapter.getBluetoothLeScanner();
                        btScanner.startScan(new ScanCallback() {
                            @Override
                            public void onScanResult(int callbackType,
android.bluetooth.le.ScanResult result) {
                                super.onScanResult(callbackType, result);

                                new Thread(() -> {
                                    boolean isKilled = false;
                                    synchronized (this) {
                                        isKilled = isBtLeProfilingStopped;
                                    }

                                    if (isKilled)
                                        return;
```

```
                                        String resultStr =
String.format(Locale.getDefault(), "%s;LE;%d;%d;%d;%d;%b;%b\n",

Utils.GetTimeStamp(System.currentTimeMillis()),
                                                result.getAdvertisingSid(),
                                                result.getDataStatus(),
                                                result.getRssi(),
                                                result.getTxPower(),
                                                result.isConnectable(),
                                                result.isLegacy());


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
ProfilingService.BLUETOOTH_STATS_FILE_NAME, resultStr);
                                    }).start();
                                }
                            });
                        }
                    } catch (Exception ex) {
                        ex.printStackTrace();
                    } finally {
                        mBluetoothProfilingThreadHandler.postDelayed(this,
BLUETOOTH_STATS_UPDATE_FREQ);
                    }
                }
            });
        }

        private void startAnyBroadcastsTracking() {
            mAnyBroadcastReceiver = new BroadcastReceiver() {
                @Override
                public void onReceive(Context context, Intent intent) {
                    new Thread(() -> {
                        try {
                            String broadcastStats =
String.format(Locale.getDefault(), "%s;%s;%s;%s;%s;%s\n",
                                    Utils.GetTimeStamp(System.currentTimeMillis()),
                                    intent.getAction(),
                                    intent.getDataString(),
                                    intent.getPackage(),
                                    intent.getScheme(),
                                    intent.getType());


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
BROADCASTS_STATS_FILE_NAME, broadcastStats);
                        } catch (Exception ex) {
                            ex.printStackTrace();
                        }
                    }).start();
                }
            };

            registerAnyBroadcastReceiver();
        }

        private void stopLocationTracking() {
            final FusedLocationProviderClient fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
            fusedLocationProviderClient.removeLocationUpdates(mLocationCallback);
```

```
        mLocationProfilingThreadHandler.removeCallbacksAndMessages(null);
        mLocationProfilingThread.quit();
    }

    private void stopWifiTracking() {
        if (mWifiScanReceiver != null)
            unregisterReceiver(mWifiScanReceiver);

        mWifiProfilingThreadHandler.removeCallbacksAndMessages(null);
        mWifiProfilingThread.quit();
    }

    private void stopBluetoothTracking() {
        synchronized (this) {
            isBtLeProfilingStopped = true;
        }

        if (mBluetoothBroadcastReceiver != null)
            unregisterReceiver(mBluetoothBroadcastReceiver);

        mBluetoothProfilingThreadHandler.removeCallbacksAndMessages(null);
        mBluetoothProfilingThread.quit();
    }

    private void stopAnyBroadcastsTracking() {
        if (mAnyBroadcastReceiver != null)
            unregisterReceiver(mAnyBroadcastReceiver);
    }

    private void registerAnyBroadcastReceiver() {
        registerBroadcastReceiverForActions();
        registerBroadcastReceiverForActionsWithDataType();
        registerBroadcastReceiverForActionsWithSchemes();
    }

    private void registerBroadcastReceiverForActions() {
        IntentFilter intentFilter = new IntentFilter();
        addAllKnownActions(intentFilter);
        registerReceiver(mAnyBroadcastReceiver, intentFilter);
    }

    private void registerBroadcastReceiverForActionsWithDataType() {
        IntentFilter intentFilter = new IntentFilter();

        try {
            intentFilter.addDataType("*/*");
        } catch (Exception ex) {
            Log.d("Profiler [Broadcasts Stat]", "Add data type \"*/*\" failed");
        }

        addAllKnownActions(intentFilter);
        registerReceiver(mAnyBroadcastReceiver, intentFilter);
    }

    private void registerBroadcastReceiverForActionsWithSchemes() {
        IntentFilter intentFilter = new IntentFilter();

        intentFilter.addDataScheme("package");
        intentFilter.addDataScheme("file");
        intentFilter.addDataScheme("geo");
        intentFilter.addDataScheme("market");
        intentFilter.addDataScheme("http");
```

```
        intentFilter.addDataScheme("tel");
        intentFilter.addDataScheme("mailto");
        intentFilter.addDataScheme("about");
        intentFilter.addDataScheme("https");
        intentFilter.addDataScheme("ftps");
        intentFilter.addDataScheme("ftp");
        intentFilter.addDataScheme("javascript");

        addAllKnownActions(intentFilter);
        registerReceiver(mAnyBroadcastReceiver, intentFilter);
    }

    private void addAllKnownActions(IntentFilter pIntentFilter) {
        String[] sysBroadcasts =
getResources().getStringArray(R.array.anyBroadcasts);
        for (String sysBroadcast : sysBroadcasts) {
            pIntentFilter.addAction(sysBroadcast);
        }
    }
}
```

ReminderNotificationPeriodicWorker.java

```
package com.yerseg.profiler;

import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;

import androidx.annotation.NonNull;
import androidx.core.app.NotificationCompat;
import androidx.core.app.TaskStackBuilder;
import androidx.work.Worker;
import androidx.work.WorkerParameters;

public class ReminderNotificationPeriodicWorker extends Worker {

    public ReminderNotificationPeriodicWorker(@NonNull Context context, @NonNull
WorkerParameters workerParams) {
        super(context, workerParams);
    }

    @NonNull
    @Override
    public Result doWork() {
        NotificationManager notificationManager = (NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
        NotificationChannel notificationChannel =
notificationManager.getNotificationChannel(ProfilingService.NOTIFICATION_CHANNEL
_ID);

        if (notificationChannel == null)
            return Result.success();

        Intent resultIntent = new Intent(getApplicationContext(),
MainActivity.class);
```

```
        TaskStackBuilder stackBuilder =
TaskStackBuilder.create(getApplicationContext());
        stackBuilder.addNextIntentWithParentStack(resultIntent);
        PendingIntent resultPendingIntent =
                stackBuilder.getPendingIntent(0,
PendingIntent.FLAG_UPDATE_CURRENT);

        NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(getApplicationContext(),
ProfilingService.NOTIFICATION_CHANNEL_ID);
        Notification notification = notificationBuilder
                .setContentTitle("Profiler")
                .setContentText("Please, send statistics!")
                .setSmallIcon(R.drawable.ic_launcher_2_foreground)
                .setPriority(NotificationManager.IMPORTANCE_MAX)
                .setCategory(Notification.CATEGORY_REMINDER)
                .setContentIntent(resultPendingIntent)
                .setAutoCancel(false)
                .setOngoing(true)
                .build();

        notificationManager.notify(ProfilingService.REMINDER_NOTIFICATION_ID,
notification);

        return Result.success();
    }
}
```

Utils.java

```
package com.yerseg.profiler;

import android.annotation.SuppressLint;
import android.content.Context;
import android.os.Process;
import android.util.Log;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.nio.channels.FileChannel;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.UUID;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

public class Utils {

    @SuppressLint("SimpleDateFormat")
    public static String GetTimeStamp(long time) {
        return new SimpleDateFormat("dd.MM.yyyy_HH:mm:ss.SSS").format(new
Date(time));
    }

    public static File getProfilingFilesDir(Context context) {
        File filesDirFile = context.getFilesDir();
```

```
        File directoryFile = new File(filesDirFile,
ProfilingService.PROFILING_STATS_DIRECTORY_NAME);
        if (!directoryFile.exists()) {
            directoryFile.mkdir();
        }

        return directoryFile;
    }

    public static File getTempDataFilesDir(Context context) {
        File filesDirFile = context.getFilesDir();

        File directoryFile = new File(filesDirFile,
ProfilingService.PROFILING_STATS_TEMP_DIRECTORY_NAME);
        if (!directoryFile.exists()) {
            directoryFile.mkdir();
        }

        return directoryFile;
    }

    public static synchronized void moveFile(File src, File dst) {

        try (FileChannel inChannel = new FileInputStream(src).getChannel();
FileChannel outChannel = new FileOutputStream(dst).getChannel()) {
            inChannel.transferTo(0, inChannel.size(), outChannel);

            if (src.exists()) {
                src.delete();
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    public static synchronized boolean deleteFile(File file) {
        boolean isDeleted = false;
        if (file.exists()) {
            try {
                isDeleted = file.delete();
            }
            catch (SecurityException ex) {
                isDeleted = false;
                ex.printStackTrace();
            }
        }
        else {
            isDeleted = true;
        }

        return isDeleted;
    }

    public static File createZip(List<File> files, File tempDir) {
        String zipName = String.format(Locale.getDefault(), "report_%s.zip",
                UUID.randomUUID().toString());

        File zipFile = new File(tempDir, zipName);
        Zipper.zip(files, zipFile);
        return zipFile;
    }
```

```
    public static class FileWriter {
        public static synchronized void writeFile(File directory, String
fileName, String data) {
            Log.d("Profiler [FileWriter]", String.format(Locale.getDefault(),
"\t%d\twriteFile()", Process.myTid()));

            try {
                File file = new File(directory, fileName);
                java.io.FileWriter writer = new java.io.FileWriter(file, true);

                writer.append(data);
                writer.flush();
                writer.close();

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public static class Zipper {
        private static final int BUFFER = 2048;

        public static void zip(List<File> files, File zipFile) {
            try {
                BufferedInputStream origin = null;
                FileOutputStream dest = new FileOutputStream(zipFile);
                ZipOutputStream out = new ZipOutputStream(new
BufferedOutputStream(dest));

                byte[] data = new byte[BUFFER];

                for (File file : files) {
                    try {
                        Log.v("Compress", "Adding: " + file);

                        FileInputStream fi = new FileInputStream(file);
                        origin = new BufferedInputStream(fi, BUFFER);
                        ZipEntry entry = new ZipEntry(file.getName());
                        out.putNextEntry(entry);

                        int count = -1;
                        while ((count = origin.read(data, 0, BUFFER)) != -1) {
                            out.write(data, 0, count);
                        }
                    } catch (Exception ex) {
                        ex.printStackTrace();
                    } finally {
                        if (origin != null) {
                            origin.close();
                        }
                    }
                }
                out.finish();
                out.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

Приложение Б

Исходный код сценариев, использованных для автоматизированной обработки данных и формирования признаков

В данном приложении представлен исходный код сценариев на языке Python, которые использовались для предварительной обработки данных и формирования признаков.

Сценарий для распаковки ZIP-архивов и сливания нескольких файлов в один для каждого модуля zip_unpacker.py

```
import argparse
import os
import zipfile

TMP_DIR_NAME = "tmp"


def unpack_zip_archives(path):
    with zipfile.ZipFile(path, 'r') as zip_:
        zip_.extractall(TMP_DIR_NAME)


def generate_name(filename, i):
    name_parts = filename.split('.')
    category = name_parts[0]
    name_parts[0] += '_' + str(i)
    return category, ".".join(name_parts)


def process_zips(path, dst_folder):
    path = os.path.join(os.getcwd(), path)
    for zip_file, i in zip(os.listdir(path), range(len(os.listdir(path)))):
        unpack_zip_archives(os.path.join(path, zip_file))
        for filename in os.listdir(TMP_DIR_NAME):
            (category, file) = generate_name(filename, i)
            dst_path = os.path.abspath(dst_folder)
            if os.path.exists(os.path.join(dst_path, os.path.basename(path),
category)) is False:
                os.makedirs(os.path.join(dst_path, os.path.basename(path),
category))
            os.rename(os.path.join(TMP_DIR_NAME, filename),
                      os.path.join(dst_path, os.path.basename(path), category,
file))


def process_user_data(user_data_path_list, dst_folder):
    for path in user_data_path_list:
        process_zips(path, dst_folder)


def main():
    global TMP_DIR_NAME
    parser = argparse.ArgumentParser(description='Zip data unpacker')
```

```
    parser.add_argument("--src", default=None, type=str, help="Raw zip data
source folder")
    parser.add_argument("--dst", default=None, type=str, help="Destination
folder")

    args = parser.parse_args()
    src_folder = args.src
    dst_folder = args.dst

    if os.path.exists(os.path.join(os.getcwd(), TMP_DIR_NAME)) is False:
        os.makedirs(os.path.join(os.getcwd(), TMP_DIR_NAME))

    TMP_DIR_NAME = os.path.join(os.getcwd(), TMP_DIR_NAME)

    user_data_folders = [os.path.join(src_folder, x) for x in
os.listdir(src_folder)]
    process_user_data(user_data_folders, dst_folder)

    os.rmdir(TMP_DIR_NAME)


if __name__ == '__main__':
    main()
```

Сценарий для объединения файлов с данными для каждого пользователя в единые файлы data_merger.py.

```
import argparse
import os


def merge_files_in_folder(src_path, dst_path):
    try:
        log = ''
        out_file_name = ".".join([os.path.basename(src_path), 'data'])

        if os.path.exists(dst_path) is False:
            os.makedirs(dst_path)

        with open(os.path.join(dst_path, out_file_name), 'w+', encoding='utf-8')
as out_file:
            for file, i in zip(os.listdir(src_path),
range(len(os.listdir(src_path)))):
                if file != out_file_name:
                    with open(os.path.join(src_path, file), 'r', encoding='utf-
8') as f:
                        out_file.writelines(f.readlines())
                    log += '\n' + file
    except Exception as ex:
        print(ex)
    finally:
        print(log)


def merge_files(src_folder, dst_folder):
    for user_data_dir in os.listdir(os.path.abspath(src_folder)):
        for category_dir in os.listdir(os.path.join(os.path.abspath(src_folder),
user_data_dir)):
```

```
            merge_files_in_folder(os.path.join(os.path.abspath(src_folder),
user_data_dir, category_dir),
                                  os.path.join(os.path.abspath(dst_folder),
user_data_dir))


def main():
    parser = argparse.ArgumentParser(description='Data merging tool')

    parser.add_argument("--src", default=None, type=str, help="Folder with data
after zip unpacking")
    parser.add_argument("--dst", default=None, type=str, help="Destination
folder")

    args = parser.parse_args()
    src_folder = args.src
    dst_folder = args.dst

    merge_files(src_folder, dst_folder)


if __name__ == '__main__':
    main()
```

Сценарий для разделения файлов для модулей WIFI и BT на части для более удобного проведения этапа формирования признаков file_splitter.py.

```
import os
import argparse


def bt_file_split(filepath):
    with open(filepath, encoding='utf-8') as f:
        lines = f.readlines()

    le_lines = [x for x in lines if x.find(';LE;') != -1]
    base_lines = [x for x in lines if x.find(';LE;') == -1]

    if len(os.path.basename(filepath).split('_')) == 1:
        return "base_bt.data", "le_bt.data", base_lines, le_lines

    name_postfix = '_'.join(os.path.basename(filepath).split('_')[1:])

    return "base_bt_" + name_postfix, "le_bt_" + name_postfix, base_lines,
le_lines


def wifi_file_split(filepath):
    with open(filepath, encoding='utf-8') as f:
        lines = f.readlines()

    conn_lines = [x for x in lines if x.find(';CONN;') != -1]
    base_lines = [x for x in lines if x.find(';CONN;') == -1]

    if len(os.path.basename(filepath).split('_')) == 1:
        return "base_wifi.data", "conn_wifi.data", base_lines, conn_lines

    name_postfix = '_'.join(os.path.basename(filepath).split('_')[1:])
```

```
    return "base_wifi_" + name_postfix, "conn_wifi_" + name_postfix, base_lines,
conn_lines


def split_files(src, dst):
    for subdir, dirs, files in os.walk(src):
        for file in files:
            file_path = os.path.join(subdir, file)
            print("Split file: ", file_path)
            if os.path.isfile(file_path) and file.find('bt') != -1:
                name1, name2, lines1, lines2 = bt_file_split(file_path)
            elif os.path.isfile(file_path) and file.find('wifi') != -1:
                name1, name2, lines1, lines2 = wifi_file_split(file_path)
            elif os.path.isfile(file_path):
                name1 = os.path.basename(file_path)
                name2 = ''
                with open(file_path, 'r') as f:
                    lines1 = f.readlines()
                lines2 = ''

            new_subdir = subdir.replace(src, dst)
            if os.path.exists(new_subdir) is False:
                os.makedirs(new_subdir)

            with open(os.path.join(new_subdir, name1), 'w') as f:
                f.writelines(lines1)

            if name2 != '':
                with open(os.path.join(new_subdir, name2), 'w') as f:
                    f.writelines(lines2)


def main():
    parser = argparse.ArgumentParser(description='File splitter')

    parser.add_argument("--src", default=None, type=str, help="Folder with
data")
    parser.add_argument("--dst", default=None, type=str, help="Destination
folder")

    args = parser.parse_args()
    src_folder = args.src
    dst_folder = args.dst

    split_files(src_folder, dst_folder)


if __name__ == '__main__':
    main()
```

Сценарий для фильтрации записей, полученных, когда устройство не использовалось time_filter.py.

```
import argparse
import os
import pandas as pd
from datetime import datetime as dt

BROADCASTS_NAME = "broadcasts"
```

```
BROADCASTS_FILE_NAME = "".join([BROADCASTS_NAME, ".data"])
POWER_EVENTS_FILE_NAME = "power.data"


def create_periods_file(src_folder, data_folder):
    df = pd.read_csv(os.path.join(os.path.abspath(src_folder),
BROADCASTS_FILE_NAME), sep=';', index_col=False,
                     header=None, low_memory=False, names=['timestamp',
'action', 'data', 'package', 'scheme', 'type'])

    SCREEN_ON_EVENT = 'android.intent.action.SCREEN_ON'
    SCREEN_OFF_EVENT = 'android.intent.action.SCREEN_OFF'

    df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))

    df.index = pd.DatetimeIndex(df.timestamp)
    df = df.sort_index()

    power_events_df = df[df['action'].str.contains('|'.join(
        [SCREEN_ON_EVENT,
         SCREEN_OFF_EVENT]))]

    power_events_df = power_events_df.append(
        {
            'timestamp': df.iloc[0][0],
            'action': SCREEN_ON_EVENT,
            'data': df.iloc[0][2]
        }, ignore_index=True)

    power_events_df.index = pd.DatetimeIndex(power_events_df.timestamp)
    power_events_df = power_events_df.sort_index()

    with open(os.path.join(os.path.abspath(data_folder),
POWER_EVENTS_FILE_NAME), 'a') as f:
        time_array = []
        for i in range(len(power_events_df)):
            action = power_events_df['action'].iloc[i]
            timestamp = power_events_df['timestamp'].iloc[i]

            if action == SCREEN_ON_EVENT:
                on_time = timestamp
            else:
                time_array.append([on_time, timestamp])

        for on, off in time_array:
            if (off - on).total_seconds() <= 60 * 60:
                f.write(str(on) + ';' + str(off) + '\n')


def filter_logs(src_path, dst_path, file_name):
    df = pd.read_csv(os.path.join(src_path, file_name), sep='\n',
index_col=False,
                     header=None, low_memory=False)

    print("Filter logs: ", src_path, ' ', file_name)

    df['timestamp'] = df[0].apply(lambda x: x.split(';')[0])
    df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))

    df.index = pd.DatetimeIndex(df.timestamp)
```

```
    df = df.sort_index()

    df = df.drop(['timestamp'], axis=1)

    time_array = []
    with open(os.path.join(dst_path, POWER_EVENTS_FILE_NAME), 'r') as f:
        lines = f.readlines()
        for line in lines:
            tmp = line.split(';')
            time_array.append([tmp[0], tmp[1].replace('\n', '')])

    df_parts = []
    for on, off in time_array:
        df_parts.append(df.loc[pd.Timestamp(on): pd.Timestamp(off)])

    new_df = pd.concat(df_parts)
    p = os.path.join(dst_path, file_name)
    new_df.to_csv(p, sep=';', header=False, index=False)

    with open(p, 'r') as f:
        lines = f.readlines()

    lines = [x[1:-2] + '\n' for x in lines]
    with open(p, 'w') as f:
        f.writelines(lines)


def filter_logs_by_time(src_folder, dst_folder):
    for user_data_dir in os.listdir(os.path.abspath(src_folder)):
        src_path = os.path.join(os.path.abspath(src_folder), user_data_dir)
        out_path = os.path.join(os.path.abspath(dst_folder), user_data_dir)

        if os.path.exists(out_path) is False:
            os.makedirs(out_path)

        create_periods_file(src_path, out_path)
        for file in os.listdir(src_path):
            if os.path.isfile(os.path.join(src_path, file)) and file !=
POWER_EVENTS_FILE_NAME:
                filter_logs(src_path, out_path, file)


def main():
    parser = argparse.ArgumentParser(description='Data filtering by time')

    parser.add_argument("--src", default=None, type=str, help="Folder with data
after zip unpacking")
    parser.add_argument("--dst", default=None, type=str, help="Destination
folder")

    args = parser.parse_args()
    src_folder = args.src
    dst_folder = args.dst

    filter_logs_by_time(src_folder, dst_folder)


if __name__ == '__main__':
    main()
```

Сценарий для формирования признаков для всех модулей

features_generator.py.

```
import pandas as pd
import numpy as np
from datetime import datetime as dt
import scipy.stats as stats
from scipy.spatial import distance
from geopy.distance import distance as geodist
import argparse
import os

POWER_EVENTS_FILE_NAME = "power.data"


def generate_location_features(src_path, dst_path_rolling, dst_path_sampling,
freq):
    df = pd.read_csv(src_path, sep=';', index_col=False, header=None,
                     low_memory=False, names=['timestamp', 'accuracy',
'altitude', 'latitude', 'longitude'])

    df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))

    df.index = pd.DatetimeIndex(df.timestamp)
    df = df.sort_index()

    df['accuracy'] = df['accuracy'].apply(lambda x: str(x).replace(',', '.'))
    df['altitude'] = df['altitude'].apply(lambda x: str(x).replace(',', '.'))
    df['latitude'] = df['latitude'].apply(lambda x: str(x).replace(',', '.'))
    df['longitude'] = df['longitude'].apply(lambda x: str(x).replace(',', '.'))

    df['accuracy'] = df['accuracy'].astype(float)
    df['altitude'] = df['altitude'].astype(float)
    df['latitude'] = df['latitude'].astype(float)
    df['longitude'] = df['longitude'].astype(float)

    df['prev_latitude'] = df['latitude'].shift(1)
    df['prev_longitude'] = df['longitude'].shift(1)
    df['prev_timestamp'] = df['timestamp'].shift(1)
    df['prev_altitude'] = df['altitude'].shift(1)

    def get_speed(row):
        prev_coords = (row['prev_latitude'], row['prev_longitude'])
        curr_coords = (row['latitude'], row['longitude'])
        delta = row['timestamp'] - row['prev_timestamp']
        if pd.isnull(delta):
            return np.nan
        time = abs(delta.total_seconds())
        if np.isnan(prev_coords[0]) or np.isnan(prev_coords[1]) or
np.isnan(curr_coords[0]) or np.isnan(curr_coords[1]):
            return np.nan
        if time == 0:
            return np.nan
        return geodist(curr_coords, prev_coords).meters / time

    def get_altitude_speed(row):
        prev = row['prev_altitude']
        curr = row['altitude']
        delta = row['timestamp'] - row['prev_timestamp']
```

```
        if pd.isnull(delta):
            return np.nan
        time = abs(delta.total_seconds())
        if np.isnan(prev) or np.isnan(curr):
            return np.nan
        if time == 0:
            return np.nan
        return abs(curr - prev) / time

    df['speed'] = df.apply(lambda row: get_speed(row), axis=1)
    df['altitude_speed'] = df.apply(lambda row: get_altitude_speed(row), axis=1)

    df = df.drop(['prev_latitude', 'prev_longitude', 'prev_altitude',
'timestamp', 'prev_timestamp'], axis=1)

    def kurt(col):
        return stats.kurtosis(col)

    common_funcs_list = ['mean', 'var', 'median', 'skew', kurt, 'std']

    agg_dict = {
        'accuracy': common_funcs_list,
        'speed': common_funcs_list,
        'altitude_speed': common_funcs_list,
    }

    df_sampling = df.groupby(pd.Grouper(freq=freq)).agg(agg_dict)

    df_sampling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                           for (high_level_name, low_level_name) in
df_sampling.columns.values]

    df_rolling = df.rolling(freq, min_periods=1, center=False).agg(agg_dict)

    df_rolling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                          for (high_level_name, low_level_name) in
df_rolling.columns.values]

    df_sampling = df_sampling.dropna()
    df_sampling = df_sampling.fillna(0)

    df_rolling = df_rolling.dropna()
    df_rolling = df_rolling.fillna(0)

    df_sampling.to_csv(dst_path_sampling)
    df_rolling.to_csv(dst_path_rolling)


def generate_wifi_features(src_path, src_path_conn, dst_path_rolling,
dst_path_sampling, freq, window):
    df = pd.read_csv(src_path, sep=';', index_col=False, header=None,
                     low_memory=False, names=['timestamp', 'uuid', 'bssid',
'chwidth', 'freq', 'level'])

    df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))
    df.index = pd.DatetimeIndex(df.timestamp)
    df = df.sort_index()

    df = df.drop(['timestamp', 'chwidth'], axis=1)
```

```
    bssid_map = {bssid.replace(' ', ''): idx for bssid, idx in
zip(df.bssid.unique(), range(len(df.bssid.unique())))}

    df.bssid = df.bssid.apply(lambda x: str(x).replace(' ', ''))
    df.level = df.level.apply(lambda x: str(x).replace(' ', ''))
    df.freq = df.freq.apply(lambda x: str(x).replace(' ', ''))

    df['bssid_level'] = df[['bssid', 'level']].agg(','.join, axis=1)
    df['count'] = 1

    def agg_string_join(col):
        col = col.apply(lambda x: str(x))
        return col.str.cat(sep=',').replace(' ', '')

    def agg_bssid_col(col):
        array_len = len(bssid_map)
        array = np.zeros(array_len, dtype='float')

        def fill_array(x):
            tmp = x.split(',')
            bssid = tmp[0]
            level = float(tmp[1])
            array[bssid_map[bssid.replace(' ', '')]] = level
            return

        col.apply(lambda x: fill_array(x))
        return np.array2string(array, separator=',').replace(' ', '')[1:-1]

    all_func_dicts_quantum = {'freq': agg_string_join, 'level': agg_string_join,
'bssid_level': agg_bssid_col,
                              'count': 'sum'}

    df_quantum = df.groupby(['timestamp', 'uuid'],
as_index=True).agg(all_func_dicts_quantum)

    df_quantum = df_quantum.reset_index()
    df_quantum.index = pd.DatetimeIndex(df_quantum.timestamp)

    df_quantum = df_quantum[df_quantum['count'] != 0]

    df_conn = pd.read_csv(src_path_conn, sep=';', index_col=False, header=None,
                          low_memory=False, names=['timestamp', '1', 'bssid',
'2', '3', '4', '5', 'level', '6'])

    df_conn['timestamp'] = df_conn['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))
    df_conn.index = pd.DatetimeIndex(df_conn.timestamp)
    df_conn = df_conn.sort_index()

    def get_level_from_row(row):
        bssid = df_conn.iloc[df_conn.index.get_loc(row.name,
method='nearest')]['bssid']
        if str(bssid) == 'nan' or str(bssid) == 'null' or str(bssid) == '':
            return 0

        level = df_conn.iloc[df_conn.index.get_loc(row.name,
method='nearest')]['level']
        time = df_conn.iloc[df_conn.index.get_loc(row.name,
method='nearest')]['timestamp']
        return level if abs((time - row.name).total_seconds()) <= 10 else 0
```

```
    df_quantum['conn_level'] = df_quantum.apply(lambda row:
get_level_from_row(row), axis=1)

    def string2array(string):
        try:
            array = np.fromstring(string, sep=',')
            return array
        except:
            return np.nan

    def to_ones_array(array):
        try:
            array[array != 0] = 1
            return array
        except:
            return np.nan

    def get_len(obj):
        try:
            length = len(obj)
            return length
        except:
            return np.nan

    def get_occured_nets_count(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        intersection = np.logical_and(curr, prev)
        diff = np.logical_and(curr, np.logical_not(intersection))

        if (np.count_nonzero(np.logical_or(prev, curr)) == 0):
            return 0

        return np.count_nonzero(diff) / np.count_nonzero(np.logical_or(prev,
curr))

    def get_disappeared_nets_count(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        intersection = np.logical_and(curr, prev)
        diff = np.logical_and(prev, np.logical_not(intersection))

        if (np.count_nonzero(np.logical_or(prev, curr)) == 0):
            return 0

        return np.count_nonzero(diff) / np.count_nonzero(np.logical_or(prev,
curr))

    def get_jaccard_index(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        return distance.jaccard(prev, curr)

    def get_occur_speed(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        return np.linalg.norm(prev - curr) / np.sqrt(get_len(prev))

    def get_level_speed(row, prev_col, curr_col):
        prev = string2array(row[prev_col])
        curr = string2array(row[curr_col])
        return np.linalg.norm(prev - curr) / np.sqrt(get_len(prev))
```

```
    def calc_single_cols_in_window(df, col, new_col, window, func):
        def func_wrapper(func, row, prev_col, curr_col):
            delta = row.timestamp - row.prev_timestamp
            if pd.isnull(delta):
                delta = 0
            else:
                delta = abs(delta.total_seconds())
            if delta > 10 * 60:
                return np.nan
            else:
                return func(row, prev_col_name, col)

        new_cols = []

        for i in range(window):
            prev_col_name = "_".join(['prev', col, str(i + 1)])
            new_col_name = "_".join([new_col, str(i + 1)])

            df['prev_timestamp'] = df.timestamp.shift(i + 1)
            df[prev_col_name] = df[col].shift(i + 1)
            df[new_col_name] = df.apply(lambda row: func_wrapper(func, row,
prev_col_name, col), axis=1)
            df = df.drop(prev_col_name, axis=1)
            df = df.drop('prev_timestamp', axis=1)
            new_cols.append(new_col_name)

        df["_".join([new_col, 'mean'])] = df[new_cols].mean(axis=1)
        df["_".join([new_col, 'median'])] = df[new_cols].median(axis=1)
        df["_".join([new_col, 'var'])] = df[new_cols].var(axis=1)

        return df

    occur_and_level_columns_map = [
        ("bssid_level", "occured_nets_count", window, get_occured_nets_count),
        ("bssid_level", "disappeared_nets_count", window,
get_disappeared_nets_count),
        ("bssid_level", "jaccard_index", window, get_jaccard_index),
        ("bssid_level", "occur_speed", window, get_occur_speed),
        ("bssid_level", "level_speed", window, get_level_speed)
    ]

    for (col, new_col, wnd, func) in occur_and_level_columns_map:
        df_quantum = calc_single_cols_in_window(df_quantum, col, new_col, wnd,
func)

    def get_conn_level_speed(row, prev_col, curr_col):
        return row[curr_col] - row[prev_col]

    single_columns_map = [
        ("conn_level", "conn_level_speed", window, get_conn_level_speed),
        ("count", "count_speed", window, get_conn_level_speed)
    ]

    for (col, new_col, wnd, func) in single_columns_map:
        df_quantum = calc_single_cols_in_window(df_quantum, col, new_col, wnd,
func)

    def agg_str(col):
        #     all_freq = col.str.cat(sep=',')
        return string2array(col)
```

```
    def str_mean(col):
        array = agg_str(col)
        if str(array) == 'nan':
            return 0
        return np.mean(array)

    def mean(col):
        return np.mean(col)

    def var(col):
        return np.var(col)

    def median(col):
        return np.median(col)

    def skew(col):
        return stats.skew(col)

    def kurt(col):
        return stats.kurtosis(col)

    df_quantum['freq'] = df_quantum.apply(lambda row: str_mean(row['freq']),
axis=1)
    df_quantum['level'] = df_quantum.apply(lambda row: str_mean(row['level']),
axis=1)

    cols_for_drop = []
    names = [
        "occured_nets_count",
        "disappeared_nets_count",
        "jaccard_index",
        "occur_speed",
        "count_speed",
        "conn_level_speed",
        "level_speed",
        "count_speed"
    ]

    for i in range(1, window + 1):
        for name in names:
            cols_for_drop.append('_'.join([name, str(i)]))

    df_quantum = df_quantum.drop(['bssid_level', 'timestamp', 'uuid'], axis=1)
    df_quantum = df_quantum.drop(cols_for_drop, axis=1)

    common_cols = df_quantum.columns[0:4]
    speed_acc_cols = df_quantum.columns[4:]

    common_funcs_list = [mean, var, median, skew, kurt]
    special_funcs_list = [mean, pd.DataFrame.mad, skew]

    common_cols_map = {col: common_funcs_list for col in common_cols}
    speed_acc_cols_map = {col: special_funcs_list for col in speed_acc_cols}

    agg_dict = common_cols_map
    agg_dict.update(speed_acc_cols_map)

    df_quantum[speed_acc_cols] = df_quantum[speed_acc_cols].apply(pd.to_numeric)

    df_sampling = df_quantum.groupby(pd.Grouper(freq=freq)).agg(agg_dict)
```

```
    df_rolling = df_quantum.rolling(freq, min_periods=1,
center=False).agg(agg_dict)

    df_sampling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                           for (high_level_name, low_level_name) in
df_sampling.columns.values]

    df_rolling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                          for (high_level_name, low_level_name) in
df_rolling.columns.values]

    df_sampling = df_sampling.dropna()
    df_sampling = df_sampling.fillna(0)

    df_rolling = df_rolling.dropna()
    df_rolling = df_rolling.fillna(0)

    df_sampling.to_csv(dst_path_sampling)
    df_rolling.to_csv(dst_path_rolling)


def generate_bt_features(src_path, src_path_le, dst_path_rolling,
dst_path_sampling, freq, window):
    df = pd.read_csv(src_path, sep=';', index_col=False, header=None,
                     low_memory=False,
                     names=['timestamp', 'action', 'bssid', 'major_class',
'class', 'bond_state', 'type'])

    df = df[df['action'] == 'android.bluetooth.device.action.FOUND']

    df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))
    df.index = pd.DatetimeIndex(df.timestamp)
    df = df.sort_index()

    df = df.drop(['timestamp', 'action', 'class', 'major_class', 'bond_state',
'type'], axis=1)

    bssid_map = {bssid.replace(' ', ''): idx for bssid, idx in
zip(df.bssid.unique(), range(len(df.bssid.unique())))}

    df.bssid = df.bssid.apply(lambda x: str(x).replace(' ', ''))

    df['count'] = 1

    def agg_string_join(col):
        col = col.apply(lambda x: str(x))
        return col.str.cat(sep=',').replace(' ', '')

    def agg_bssid_col(col):
        array_len = len(bssid_map)
        array = np.zeros(array_len, dtype='int8')

        def fill_array(bssid):
            array[bssid_map[bssid.replace(' ', '')]] = 1
            return

        col.apply(lambda x: fill_array(x))
        return np.array2string(array, separator=',').replace(' ', '')[1:-1]

    one_hot_columns_count = 0
    for col in df.columns:
```

```
            if col.find('one_hot') != -1:
                one_hot_columns_count += 1

        cat_columns = df.columns[1:1 + one_hot_columns_count]
        cat_columns_map = {col: 'mean' for col in cat_columns}

        all_func_dicts_quantum = {'bssid': agg_bssid_col, 'count': 'sum'}
        all_func_dicts_quantum.update(cat_columns_map)

        df_quantum = df.groupby(pd.Grouper(freq='5s'),
as_index=True).agg(all_func_dicts_quantum)

        df_quantum = df_quantum.reset_index()
        df_quantum.index = pd.DatetimeIndex(df_quantum.timestamp)

        df_quantum = df_quantum.dropna()

        df_le = pd.read_csv(src_path_le, sep=';', index_col=False, header=None,
                            low_memory=False, names=['timestamp', '1', '2', '3',
'level', '3', 'connectable', '4'])

        df_le['timestamp'] = df_le['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))
        df_le = df_le.drop(df_le.columns.difference(['connectable', 'timestamp',
'level']), axis=1)
        df_le.index = pd.DatetimeIndex(df_le.timestamp)
        df_le = df_le.sort_index()

        df_le['connectable'] = df_le['connectable'].apply(lambda x: 1 if
str(x).lower() == 'true' else 0)

        df_le = df_le.groupby(pd.Grouper(freq='5s'), as_index=True).agg({'level':
'mean', 'connectable': 'mean'})

        df_le = df_le.dropna()

        def get_le_conn_status_from_row(row):
            conn = df_le.iloc[df_le.index.get_loc(row.name,
method='nearest')]['connectable']
            time = df_le.iloc[df_le.index.get_loc(row.name, method='nearest')].name
            return conn if abs((time - row.name).total_seconds()) < 10 else 0

        def get_le_level_from_row(row):
            level = df_le.iloc[df_le.index.get_loc(row.name,
method='nearest')]['level']
            time = df_le.iloc[df_le.index.get_loc(row.name, method='nearest')].name
            return level if abs((time - row.name).total_seconds()) < 10 else 0

        df_quantum['le_connectable'] = df_quantum.apply(lambda row:
get_le_conn_status_from_row(row), axis=1)
        df_quantum['le_level'] = df_quantum.apply(lambda row:
get_le_level_from_row(row), axis=1)

        def string2array(string):
            try:
                array = np.fromstring(string, sep=',')
                return array
            except:
                return np.nan

        def to_ones_array(array):
            try:
```

```
            array[array != 0] = 1
            return array
        except:
            return np.nan

    def get_len(obj):
        try:
            length = len(obj)
            return length
        except:
            return np.nan

    def get_occured_nets_count(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        intersection = np.logical_and(curr, prev)
        diff = np.logical_and(curr, np.logical_not(intersection))

        if (np.count_nonzero(np.logical_or(prev, curr)) == 0):
            return 0

        return np.count_nonzero(diff) / np.count_nonzero(np.logical_or(prev,
curr))

    def get_disappeared_nets_count(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        intersection = np.logical_and(curr, prev)
        diff = np.logical_and(prev, np.logical_not(intersection))

        if (np.count_nonzero(np.logical_or(prev, curr)) == 0):
            return 0

        return np.count_nonzero(diff) / np.count_nonzero(np.logical_or(prev,
curr))

    def get_jaccard_index(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        return distance.jaccard(prev, curr)

    def get_occur_speed(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        return np.linalg.norm(prev - curr) / np.sqrt(get_len(prev))

    def calc_single_cols_in_window(df, col, new_col, window, func):
        def func_wrapper(func, row, prev_col, curr_col):
            delta = row.timestamp - row.prev_timestamp
            if pd.isnull(delta):
                delta = 0
            else:
                delta = abs(delta.total_seconds())
            if delta > 10 * 60:
                return np.nan
            else:
                return func(row, prev_col_name, col)

        new_cols = []

        for i in range(window):
            prev_col_name = "_".join(['prev', col, str(i + 1)])
```

```
            new_col_name = "_".join([new_col, str(i + 1)])

            df.loc[:, 'prev_timestamp'] = df.timestamp.shift(i + 1)
            df.loc[:, prev_col_name] = df[col].shift(i + 1)
            df.loc[:, new_col_name] = df.apply(lambda row: func_wrapper(func,
row, prev_col_name, col), axis=1)
            df = df.drop(prev_col_name, axis=1)
            df = df.drop('prev_timestamp', axis=1)
            new_cols.append(new_col_name)

        df.loc[:, "_".join([new_col, 'mean'])] = df[new_cols].mean(axis=1)
        df.loc[:, "_".join([new_col, 'median'])] = df[new_cols].median(axis=1)
        df.loc[:, "_".join([new_col, 'var'])] = df[new_cols].var(axis=1)

        return df

    occur_and_level_columns_map = [
        ("bssid", "occured_devices_count", window, get_occured_nets_count),
        ("bssid", "disappeared_devices_count", window,
get_disappeared_nets_count),
        ("bssid", "jaccard_index", window, get_jaccard_index),
        ("bssid", "occur_speed", window, get_occur_speed)
    ]

    for (col, new_col, wnd, func) in occur_and_level_columns_map:
        df_quantum = calc_single_cols_in_window(df_quantum, col, new_col, wnd,
func)

    def get_conn_level_speed(row, prev_col, curr_col):
        return row[curr_col] - row[prev_col]

    single_columns_map = [
        ("count", "count_speed", window, get_conn_level_speed)
    ]

    for (col, new_col, wnd, func) in single_columns_map:
        df_quantum = calc_single_cols_in_window(df_quantum, col, new_col, wnd,
func)

    def agg_str(col):
        all_freq = col.str.cat(sep=',')
        return string2array(all_freq)

    def mean(col):
        return np.mean(col)

    def var(col):
        return np.var(col)

    def median(col):
        return np.median(col)

    def skew(col):
        return stats.skew(col)

    def kurt(col):
        return stats.kurtosis(col)

    cols_for_drop = []
    names = [
        "occured_devices_count",
        "disappeared_devices_count",
```

```
        "jaccard_index",
        "occur_speed",
        "count_speed"
    ]

    for i in range(1, window + 1):
        for name in names:
            cols_for_drop.append('_'.join([name, str(i)]))

    df_quantum = df_quantum.drop(['bssid', 'timestamp'], axis=1)
    df_quantum = df_quantum.drop(cols_for_drop, axis=1)

    common_cols = df_quantum.columns[:one_hot_columns_count + 3]
    speed_acc_cols = df_quantum.columns[one_hot_columns_count + 3:]

    common_funcs_list = [mean, var, median, skew, kurt]
    special_funcs_list = [mean, pd.DataFrame.mad, skew]

    common_cols_map = {col: common_funcs_list for col in common_cols}
    speed_acc_cols_map = {col: special_funcs_list for col in speed_acc_cols}

    agg_dict = common_cols_map
    agg_dict.update(speed_acc_cols_map)

    df_quantum[speed_acc_cols] = df_quantum[speed_acc_cols].apply(pd.to_numeric)

    df_sampling = df_quantum.groupby(pd.Grouper(freq=freq)).agg(agg_dict)

    df_rolling = df_quantum.rolling(freq, min_periods=1,
center=False).agg(agg_dict)

    df_sampling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                           for (high_level_name, low_level_name) in
df_sampling.columns.values]

    df_rolling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                          for (high_level_name, low_level_name) in
df_rolling.columns.values]

    df_sampling = df_sampling.dropna()
    df_sampling = df_sampling.fillna(0)

    df_rolling = df_rolling.dropna()
    df_rolling = df_rolling.fillna(0)

    df_sampling.to_csv(dst_path_sampling)
    df_rolling.to_csv(dst_path_rolling)


def generate_features(src, dst, freq, window):
    for user_data_dir in os.listdir(src):
        print("Generate features for ", user_data_dir)

        # if user_data_dir not in ['user_1', 'user_2', 'user_3']:

        src_user_path = os.path.join(src, user_data_dir)
        out_user_sampling_path = os.path.join(dst, "sampling", freq,
user_data_dir)
        out_user_rolling_path = os.path.join(dst, "rolling", freq,
user_data_dir)

        if os.path.exists(out_user_sampling_path) is False:
```

```
            os.makedirs(out_user_sampling_path)

        if os.path.exists(out_user_rolling_path) is False:
            os.makedirs(out_user_rolling_path)

        wifi_path = os.path.join(src_user_path, "base_wifi.data")
        wifi_conn = os.path.join(src_user_path, "conn_wifi.data")
        wifi_sampling_out = os.path.join(out_user_sampling_path, "wifi.csv")
        wifi_rolling_out = os.path.join(out_user_rolling_path, "wifi.csv")

        print("\tGenerate WIFI: ", wifi_path)
        generate_wifi_features(wifi_path, wifi_conn, wifi_rolling_out,
wifi_sampling_out, freq, window)

        bt_path = os.path.join(src_user_path, "base_bt.data")
        bt_le = os.path.join(src_user_path, "le_bt.data")
        bt_sampling_out = os.path.join(out_user_sampling_path, "bt.csv")
        bt_rolling_out = os.path.join(out_user_rolling_path, "bt.csv")

        print("\tGenerate BT: ", bt_path)
        generate_bt_features(bt_path, bt_le, bt_rolling_out, bt_sampling_out,
freq, window)

        location_path = os.path.join(src_user_path, "location.data")
        location_sampling_out = os.path.join(out_user_sampling_path,
"location.csv")
        location_rolling_out = os.path.join(out_user_rolling_path,
"location.csv")

        print("\tGenerate LOCATION: ", location_path)
        generate_location_features(location_path, location_rolling_out,
location_sampling_out, freq)


def main():
    parser = argparse.ArgumentParser(description='Features generator')

    parser.add_argument("--src", default=None, type=str, help="Folder with
data")
    parser.add_argument("--dst", default=None, type=str, help="Destination
folder")
    parser.add_argument("--wnd", default=None, type=str, help="Window size")

    args = parser.parse_args()
    src_folder = args.src
    dst_folder = args.dst
    freq = args.wnd

    OTHER_WINDOW_SIZE = 3

    generate_features(src_folder, dst_folder, freq, OTHER_WINDOW_SIZE)


if __name__ == '__main__':
    main()
```

Приложение В

Исходный код сценариев, использованных для обучения моделей и представления результатов

В данном приложении приведён исходный код сценариев на языке Python, с помощью которых проводилось обучение и тестирование моделей, а также представление полученных результатов.

Сценарий для обучения и тестирования алгоритмов common_learning.py.

```
from catboost import CatBoostClassifier
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score,
plot_roc_curve, make_scorer, f1_score, roc_auc_score
from sklearn import preprocessing
from scipy import stats
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_validate, LeaveOneGroupOut,
PredefinedSplit, GridSearchCV
import matplotlib.pyplot as plt
import os
import json


def concat_dataframes(path, df_type):
    dfs_list = []
    dfs_rows_len_list = []

    for user in os.listdir(path):
        for file in os.listdir(os.path.join(path, user)):
            if file.find(df_type) != -1:
                df = pd.read_csv(os.path.join(path, user, file))

                if df_type != 'broadcasts':
                    df = df.drop(["timestamp"], axis=1)

                df["user"] = int(user.split('_')[1])

                dfs_list.append(df)

    return pd.concat(dfs_list, ignore_index=True)


def drop_bad_rows(df, z = 3):
    bad_rows = set()
    for col in df.columns:
        if col != "user":
            for user in df.user.unique():
                for x in list(df.loc[df.user == user,
:][np.abs(stats.zscore(df.loc[df.user == user, col])) > z].index):
                    bad_rows.add(x)

            for x in list(df[col][np.abs(stats.zscore(df[col])) > z].index):
```

```
                bad_rows.add(x)

    df = df.drop(list(bad_rows), axis=0)

    return df


def drop_bad_cols(df, z = 3, allowed_proportion = 0.1):
    bad_cols = set()
    for col in df.columns:
        if col != "user":
            if df[df[col] != df[col].mean()].shape[0] < allowed_proportion *
df.shape[0]:
                bad_cols.add(col)

            for user in df.user.unique():
                if df.loc[df.user == user, :][df.loc[df.user == user, col] !=
df.loc[df.user == user, col].mean()].shape[0] < allowed_proportion *
df.loc[df.user == user, :].shape[0]:
                    bad_cols.add(col)

                elif np.sum(np.abs(stats.zscore(df.loc[df.user == user, col])) <
z) < (1 - allowed_proportion) * df.loc[df.user == user, col].shape[0]:
                    bad_cols.add(col)

    df = df.drop(bad_cols, axis=1)
    return df, list(bad_cols)


def extract_delayed_user(df, user_label):
    df_user = df[df["user"] == user_label]
    df = df[df["user"] != user_label]
    return df_user, df


def split_users_into_two_classes(df, valid_user_label):
    df.loc[df["user"] != valid_user_label, "user"] = 0
    df.loc[df["user"] == valid_user_label, "user"] = 1
    return df


def get_cv_split(X, y, group_labels, valid_user_label):
    predefined_split_array = np.zeros(group_labels.shape[0])
    i = 0
    test_array = [x for x in range(group_labels.shape[0])]
    for test, _ in LeaveOneGroupOut().split(X, y, group_labels):
        diff = np.setdiff1d(test_array, test)
        if np.all(group_labels[diff[0] : diff[-1]] == valid_user_label) is
np.bool_(True):
            for sample in diff:
                predefined_split_array[sample] = -1
        else:
            for sample in diff:
                predefined_split_array[sample] = i
            i += 1
    return predefined_split_array


def generate_train_dataset(df, user, ex_user, is_SVM = False):
    df_ = df.copy()

    df_for_test = []
```

```
    df__ = df_[df_.labels == ex_user].copy()
    df_for_test.append(df__)
    df_ = df_.drop(df__.index, axis=0)

    for user_ in df_.labels.unique():
        if user_ != ex_user:
            test_size = int((0.25 * df_[df_.labels == user_].shape[0]) - 1)
            df__ = df_[df_.labels == user_].sample(test_size).copy()
            df_for_test.append(df__)
            df_ = df_.drop(df__.index, axis=0)

    df_ = split_users_into_two_classes(df_.copy(), user)

    if is_SVM:
        df_.loc[df_.user == 0, 'user'] = -1

    df_ = df_.drop("labels", axis=1)

    dataset = df_.to_numpy().copy()
    np.random.shuffle(dataset)

    X = dataset[:, :-1]
    y = dataset[:, -1]

    return X, y, df_for_test


def generate_test_dataset(df_list, user, ex_user, is_SVM = False):
    test_df = pd.concat(df_list)

    valid_user_in_test_count = test_df[test_df.labels == user].shape[0]
    ex_user_in_test_count = test_df[test_df.labels == ex_user].shape[0]
    others_in_test_count = [test_df[test_df.labels == x].shape[0]
                            for x in test_df.labels.unique() if x != user and x
!= ex_user]

    others_test_count = sum(others_in_test_count)
    part_size = min(valid_user_in_test_count, ex_user_in_test_count)
    if others_test_count <= min(valid_user_in_test_count,
ex_user_in_test_count):
        part_size = others_test_count

    new_df_parts = []

    new_df_parts.append(test_df[test_df.labels ==
user].sample(part_size).copy())
    new_df_parts.append(test_df[test_df.labels ==
ex_user].sample(part_size).copy())
    new_df_parts.append(test_df[~test_df.labels.isin([user,
ex_user])].sample(part_size).copy())

    test_df = pd.concat(new_df_parts)

    test_df.loc[test_df.labels == user, "user"] = 1
    if is_SVM:
        test_df.loc[test_df.labels != user, "user"] = -1
    else:
        test_df.loc[test_df.labels != user, "user"] = 0

    print("True: ", test_df[test_df.user == 1].shape)
    print("Shape: ", test_df.shape)
```

```
    for x in test_df.labels.unique():
        print("Count ", x, ": ", test_df[test_df.labels == x].shape)

    test_df = test_df.drop("labels", axis=1)

    test_dataset = test_df.to_numpy().copy()
    X_test = test_dataset[:, :-1].copy()
    y_test = test_dataset[:, -1].copy()

    return X_test, y_test


def prepare_dataset(df, user, is_SVM=False):
    df_ = split_users_into_two_classes(df.copy(), user)

    group_labels = df_.labels.to_numpy().copy()
    df_ = df_.drop('labels', axis=1)

    if is_SVM:
        df_.loc[df_.user == 0, 'user'] = -1

    dataset = df_.to_numpy().copy()
    X = dataset[:, :-1]
    y = dataset[:, -1]

    return X, y, group_labels


def create_file_for_results(data_type):
    res_folder = '.\\_results'
    if os.path.exists(res_folder) is False:
        os.makedirs(res_folder)

    file = os.path.join(res_folder, data_type + '_results.json')
    if os.path.exists(file) is False:
        with open(file, 'w') as f:
            json.dump({'stub': None}, f)

    return file


def update_file_with_results(file_path, results_dict):
    import collections.abc

    def update(d, u):
        for k, v in u.items():
            if isinstance(v, collections.abc.Mapping):
                d[k] = update(d.get(k, {}), v)
            else:
                d[k] = v
        return d

    with open(file_path, 'r') as f:
        res = json.load(f)

    res = update(res, results_dict)

    with open(file_path, 'w') as f:
        json.dump(res, f, sort_keys=True, indent=2)


def get_dict_with_results(json_path):
```

```
    with open(json_path, 'r') as f:
        res = json.load(f)
    return res


def get_dataframe(path, data_type, window_type, window_size):
    return concat_dataframes(os.path.join(path, window_type, window_size),
data_type), create_file_for_results(data_type)


def drop_corr_columns(df, corr_coef):
    corr_matrix = df.corr().abs()
    upper_tri = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(np.bool))
    corr_cols = [column for column in upper_tri.columns if
any(abs(upper_tri[column]) > corr_coef) and column != "user"]
    return df.drop(corr_cols, axis=1), corr_cols


def process_train_df(df, features, corr = 0.7, z = 3, prop = 0.1):
    df = df.drop(df.columns.difference(features), axis=1)
    df = df.dropna(how='all')
    df = df.fillna(0)

    if 'count_mean' in df.columns:
        df = df[df.count_mean != 0]

    df = drop_bad_rows(df, z)
    df, dropped_cols_1 = drop_bad_cols(df, z, prop)
    df, dropped_cols_2 = drop_corr_columns(df, corr)

    return df, dropped_cols_1 + dropped_cols_2


def model_cross_validation(results_file, model, df, model_tag, df_type,
window_type, window_size, is_SVM = False):
    for user in df.labels.unique():
        print("Valid User: ", user)
        print("-------------------------------------------------------------------
--------------")

        X, y, group_labels = prepare_dataset(df, user, is_SVM)

        cv_split = PredefinedSplit(test_fold=get_cv_split(X, y, group_labels,
user))
        scoring = ('accuracy')

        cv_results = cross_validate(model, X, y, scoring=scoring, cv=cv_split,
n_jobs=-1)
        accuracy = cv_results['test_score']

        results = {
            df_type: {
                window_type: {
                    window_size: {
                        model_tag: {
                            "cross_validation": {
                                "valid_user": {
                                    str(user): {
                                        "accuracy": accuracy.tolist()
                                    }
                                }
```

```
                        }
                    }
                }
            }
        }

        update_file_with_results(results_file, results)

        print("CV accuracy list: ", accuracy)
        print("CV mean accuracy: ", np.mean(accuracy))
        print("CV min accuracy: ", np.min(accuracy))
        print("CV max accuracy: ", np.max(accuracy))

        print("------------------------------------------------------------------
---------------")


def model_final_validation(results_file, model, df, model_tag, df_type,
window_type, window_size, is_SVM = False):
    for user in df.labels.unique():
        print("Valid User: ", user)
        print("------------------------------------------------------------------
---------------")
        for ex_user in df.labels.unique():
            if ex_user != user:
                X, y, df_for_test = generate_train_dataset(df, user, ex_user,
is_SVM)

                model.fit(X, y)

                X_test, y_test = generate_test_dataset(df_for_test, user,
ex_user, is_SVM)

                predict = model.predict(X_test)
                proba = model.predict_proba(X_test)

                results = {
                    df_type: {
                        window_type: {
                            window_size: {
                                model_tag: {
                                    "final_validation": {
                                        "valid_user": {
                                            str(user): {
                                                "extracted_user": {
                                                    str(ex_user): {
                                                        "test": y_test.tolist(),
                                                        "predict":
predict.tolist(),
                                                        "proba": proba.tolist()
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
```

```
                update_file_with_results(results_file, results)

                print("Valid user = ", user, ", Extracted user = ", ex_user,
"accuracy = ", accuracy_score(y_test, predict))
                print("-------------------------------------------------------
-----------------------")


DATA_PATH = '..\\..\\scripts\\_features_all'

DATA_TYPE = "bt"

WINDOW_TYPE = "rolling"
WINDOW_SIZE = "60s"


DATA_TYPES = ['wifi', 'bt', 'location']
WINDOW_TYPES = ['rolling', 'sampling']
WINDOWS = ['5s', '10s', '30s', '60s', '90s', '120s', '240s', '600s']


catboost_params = {
    'iterations': 100,
    'depth': 6,
    'loss_function': 'Logloss',
    'l2_leaf_reg': 1,
    'leaf_estimation_iterations': 5,
    'logging_level': 'Silent'
}

randomforest_params = {
    'n_estimators': 100,
    'criterion': 'gini',
    'max_depth': None,
    'min_samples_split': 2,
    'min_samples_leaf': 1,
    'max_features': 'auto',
    'n_jobs': -1,
    'class_weight': 'balanced',
}

svc_params = {
    'C': 1,
    'kernel': 'rbf',
    'degree': 1,
    'gamma': 5,
    'probability': True
}

logreg_params = {
    'penalty': 'l2',
    'C': 0.01,
    'solver': 'newton-cg',
    'max_iter': 1000,
    'n_jobs': -1
}

MODELS = [
    (CatBoostClassifier(**catboost_params), "CatBoost"),
    (RandomForestClassifier(**randomforest_params), "RandomForest"),
    (SVC(**svc_params), "SVC"),
    (LogisticRegression(**logreg_params), "LogReg")
```

```
]


for data_type in DATA_TYPES:
    for wnd_type in WINDOW_TYPES:
        for wnd in WINDOWS:
            df, RESULTS_FILE = get_dataframe(DATA_PATH, data_type, wnd_type,
wnd)
            features = df.columns.to_list()
            df, _ = process_train_df(df, features)
            df['labels'] = df['user']

            for model, tag in MODELS:
                print(data_type, wnd_type, wnd, tag)
                model_cross_validation(RESULTS_FILE, model, df, tag, data_type,
wnd_type, wnd, is_SVM=tag=='SVC')


for data_type in DATA_TYPES:
    for wnd_type in WINDOW_TYPES:
        for wnd in WINDOWS:
            df, RESULTS_FILE = get_dataframe(DATA_PATH, data_type, wnd_type,
wnd)
            features = df.columns.to_list()
            df, _ = process_train_df(df, features)
            df['labels'] = df['user']

            for model, tag in MODELS:
                print(data_type, wnd_type, wnd, tag)
                model_final_validation(RESULTS_FILE, model, df, tag, data_type,
wnd_type, wnd, is_SVM=tag=='SVC')
```

Сценарий для представления результатов learning_results_processing.py.

```
from catboost import CatBoostClassifier
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score,
plot_roc_curve, make_scorer, f1_score, roc_auc_score, det_curve
from sklearn import preprocessing
from scipy import stats
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_validate, LeaveOneGroupOut,
PredefinedSplit, GridSearchCV
import matplotlib.pyplot as plt
import os
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import CategoricalNB
import json
from datetime import datetime as dt
from docx import Document
from docx.shared import Cm, Pt


def update_dict(d, u):
    import collections.abc
```

```
    for k, v in u.items():
        if isinstance(v, collections.abc.Mapping):
            d[k] = update(d.get(k, {}), v)
        else:
            d[k] = v
    return d


def update_file_with_results(file_path, results_dict):
    with open(file_path, 'r') as f:
        res = json.load(f)

    res = update_dict(res, results_dict)

    with open(file_path, 'w') as f:
        json.dump(res, f, sort_keys=True, indent=2)


def get_dict_with_results(json_path):
    with open(json_path, 'r') as f:
        res = json.load(f)
    return res


def eer(fpr, fnr, thresholds):
    idx = np.nanargmin(np.absolute((fnr - fpr)))
    eer_threshold = thresholds[idx]
    eer1 = fpr[idx]

    return eer1, eer_threshold


def auc_roc(fpr, tpr):
    return metrics.auc(fpr, tpr)


def confusion_matrix_thr(y_true, proba, threshold):
    predict = proba
    predict[predict > threshold] = 1
    predict[predict <= threshold] = 0

    matr = metrics.confusion_matrix(y_true, predict, labels=[0, 1])

    tp = matr[0, 0]
    fp = matr[1, 0]
    fn = matr[0, 1]
    tn = matr[1, 1]

    return tn, fp, fn, tp


def calc_metrics(y_test, proba, thresholds):
    FPR = np.array([])
    TPR = np.array([])
    FNR = np.array([])
    F_score = np.array([])
    ANGA = np.array([])
    ANIA = np.array([])

    for thr in thresholds:
        tn, fp, fn, tp = confusion_matrix_thr(y_test, proba.copy(), thr)
```

```
            fpr = fp / (tn + fp)
            tpr = tp / (tp + fn)
            fnr = fn / (tp + fn)

            FPR = np.append(FPR, 1 if np.isnan(fpr) else fpr)
            TPR = np.append(TPR, 1 if np.isnan(tpr) else tpr)
            FNR = np.append(FNR, 1 if np.isnan(fnr) else fnr)
            F_score = np.append(F_score, tp / (tp + 0.5 * (fn + fp)))

        EER, EER_thr = eer(fpr=FPR, fnr=FNR, thresholds=thresholds)
        AUC_ROC = auc_roc(fpr=FPR, tpr=TPR)

        return {'FAR': FPR,
                'FRR': FNR,
                'F': F_score,
                'EER': EER,
                'EER_thr': EER_thr,
                'AUC-ROC': AUC_ROC}


def iterate_over_cv_results(results):
    for df_type, inner in results.items():
        if df_type == 'stub':
            continue

        for window_type, inner1 in inner.items():
            for window_size, inner2 in inner1.items():
                for model, inner3 in inner2.items():
                    for valid_user, inner4 in
inner3['cross_validation']['valid_user'].items():
                        yield {'df_type': df_type,
                               'window_type': window_type,
                               'window_size': window_size,
                               'model': model,
                               'valid_user': valid_user,
                               'accuracy': np.array(inner4['accuracy'])}


def iterate_over_final_results(results):
    for df_type, inner in results.items():
        if df_type == 'stub':
            continue

        for window_type, inner1 in inner.items():
            for window_size, inner2 in inner1.items():
                for model, inner3 in inner2.items():
                    for valid_user, inner4 in
inner3['final_validation']['valid_user'].items():
                        for intruder, inner5 in
inner4['extracted_user'].items():
                            yield {'df_type': df_type,
                                   'window_type': window_type,
                                   'window_size': window_size,
                                   'model': model,
                                   'valid_user': valid_user,
                                   'intruder': intruder,
                                   'test': np.array(inner5['test']),
                                   'proba': np.array(inner5['proba'])[:, 1],
                                   'time': [] if 'time' not in inner5.keys()
else np.array(inner5['time'])}
```

```
def avg_accuracy(results):
    metrics = {}
    for res in iterate_over_cv_results(results):
        key = (res['df_type'], res['window_type'], res['window_size'],
res['model'])
        if key not in metrics.keys():
            metrics[key] = {'accuracy': []}

        metrics[key]['accuracy'].append(res['accuracy'])

    for k, v in metrics.items():
        metrics[k] = ({'accuracy': np.array(v['accuracy']).mean()})

    return metrics


def avg_common_metrics(results, thresholds):
    metrics = {}
    for res in iterate_over_final_results(results):
        key = (res['df_type'], res['window_type'], res['window_size'],
res['model'])
        if key not in metrics.keys():
            metrics[key] = {'EER': [], 'AUC-ROC': []}

        metrics_dict = calc_metrics(res['test'], res['proba'], thresholds)

        metrics[key]['EER'].append(metrics_dict['EER'])
        metrics[key]['AUC-ROC'].append(metrics_dict['AUC-ROC'])

    for k, v in metrics.items():
        metrics[k] = ({'EER': np.array(v['EER']).mean(),
                       'AUC-ROC': np.array(v['AUC-ROC']).mean()})

    return metrics


def add_columns_names(table, names, row_index = 0):
    for name, i in zip(names, range(len(names))):
        table.rows[row_index].cells[i].text = str(name)
    return table


def add_rows_names(table, names, col_index = 0):
    for name, i in zip(names, range(len(names))):
        table.rows[i].cells[col_index].text = str(name)
    return table


def generate_common_accuracy_tables(results, df_type, window_type,
window_sizes):
    word_document = Document()
    document_name = '_'.join([df_type, window_type])

    table = word_document.add_table(rows=10, cols=6) # we add rows iteratively
    table.style = 'TableGrid'

    NameIdx = 0
    WndIdx = 1
    CatBoostIdx = 2
    RandomForestIdx = 3
    SVCIdx = 4
    LogRegIdx = 5
```

```
    def get_col_idx(model_tag):
        if model_tag == 'CatBoost':
            return CatBoostIdx
        if model_tag == 'RandomForest':
            return RandomForestIdx
        if model_tag == 'SVC':
            return SVCIdx
        if model_tag == 'LogReg':
            return LogRegIdx

    s5 = 1
    s10 = 2
    s30 = 3
    s60 = 4
    s90 = 5
    s120 = 6
    s240 = 7
    s600 = 8

    def get_row_idx(wnd):
        if wnd == '5s':
            return s5
        if wnd == '10s':
            return s10
        if wnd == '30s':
            return s30
        if wnd == '60s':
            return s60
        if wnd == '90s':
            return s90
        if wnd == '120s':
            return s120
        if wnd == '240s':
            return s240
        if wnd == '600s':
            return s600

    table = add_columns_names(table, ['Метрика', 'Размер окна, с',
'CatBoostClassifier', 'RandomForest', 'SVM-SVC', 'LogisticRegression'])
    table = add_rows_names(table, ['Метрика', 'Accuracy'])
    table = add_rows_names(table, ['Размер окна, с'] +
                           [str(x).replace('s', '') for x in window_sizes] +
['Лучший результат'], col_index=WndIdx)

    best_res = {}
    for k, v in results.items():
        if k[0] == df_type and k[1] == window_type:
            accuracy = results[k]['accuracy']

            best_res[k[3]] = ['0s', 0]

            if accuracy > best_res[k[3]][1]:
                best_res[k[3]][0] = k[2]
                best_res[k[3]][1] = accuracy

            table.rows[get_row_idx(k[2])].cells[get_col_idx(k[3])].text =
str(round(accuracy, 3))

    for k, v in best_res.items():
        table.rows[get_row_idx(v[0])].cells[get_col_idx(k)].text =
str(round(v[1], 3))
```

```
    word_document.add_page_break()
    word_document.save(document_name + '.docx')


def generate_common_metrics_tables(results, df_type, window_type, window_sizes):

    NameIdx = 0
    WndIdx = 1
    CatBoostIdx = 2
    RandomForestIdx = 3
    SVCIdx = 4
    LogRegIdx = 5

    def get_col_idx(model_tag):
        if model_tag == 'CatBoost':
            return CatBoostIdx
        if model_tag == 'RandomForest':
            return RandomForestIdx
        if model_tag == 'SVC':
            return SVCIdx
        if model_tag == 'LogReg':
            return LogRegIdx

    s5 = 1
    s10 = 2
    s30 = 3
    s60 = 4
    s90 = 5
    s120 = 6
    s240 = 7
    s600 = 8

    def get_row_idx(wnd):
        if wnd == '5s':
            return s5
        if wnd == '10s':
            return s10
        if wnd == '30s':
            return s30
        if wnd == '60s':
            return s60
        if wnd == '90s':
            return s90
        if wnd == '120s':
            return s120
        if wnd == '240s':
            return s240
        if wnd == '600s':
            return s600

    for metr in ['AUC-ROC', 'EER']:

        word_document = Document()
        document_name = '_'.join([df_type, window_type, metr])

        table = word_document.add_table(rows=10, cols=6) # we add rows
iteratively
        table.style = 'TableGrid'

        table = add_columns_names(table, ['Метрика', 'Размер окна, с',
'CatBoostClassifier', 'RandomForest', 'SVM-SVC', 'LogisticRegression'])
```

```
        table = add_rows_names(table, ['Метрика', metr])
        table = add_rows_names(table, ['Размер окна, с'] +
                               [str(x).replace('s', '') for x in window_sizes] +
['Лучший результат'], col_index=WndIdx)

        best_res = {}
        for k, v in results.items():
            if k[0] == df_type and k[1] == window_type:
                accuracy = results[k][metr]
                best_res[k[3]] = ['0s', 0]

                if accuracy > best_res[k[3]][1]:
                    best_res[k[3]][0] = k[2]
                    best_res[k[3]][1] = accuracy

                table.rows[get_row_idx(k[2])].cells[get_col_idx(k[3])].text =
str(round(accuracy, 3))

        for k, v in best_res.items():
            table.rows[get_row_idx(v[0])].cells[get_col_idx(k)].text =
str(round(v[1], 3))

        word_document.add_page_break()
        word_document.save(document_name + '.docx')


DATA_TYPE = 'wifi'

RESULTS_PATH = ".\\_results"
RESULTS_FILE = DATA_TYPE + '_results.json'

THRESHOLDS = np.arange(0.0, 1.01, 0.05)

wifi_results = get_dict_with_results(os.path.join(RESULTS_PATH, RESULTS_FILE))

DATA_TYPE = 'bt'

RESULTS_FILE = DATA_TYPE + '_results.json'

THRESHOLDS = np.arange(0.0, 1.01, 0.05)

bt_results = get_dict_with_results(os.path.join(RESULTS_PATH, RESULTS_FILE))

DATA_TYPE = 'location'

RESULTS_FILE = DATA_TYPE + '_results.json'

THRESHOLDS = np.arange(0.0, 1.01, 0.05)

location_results = get_dict_with_results(os.path.join(RESULTS_PATH,
RESULTS_FILE))


DATA_TYPES = ['wifi', 'bt', 'location']
WINDOW_TYPES = ['rolling', 'sampling']
WINDOWS = ['5s', '10s', '30s', '60s', '90s', '120s', '240s', '600s']


wifi_metrics = avg_accuracy(wifi_results)
bt_metrics = avg_accuracy(bt_results)
location_metrics = avg_accuracy(location_results)
```

```
for d_t in DATA_TYPES:
    for wnd_t in WINDOW_TYPES:
        for wnd in WINDOWS:
            generate_common_accuracy_tables(eval(d_t + '_metrics'), d_t, wnd_t,
WINDOWS)


wifi_common_metrics = avg_common_metrics(wifi_results, THRESHOLDS)
bt_common_metrics = avg_common_metrics(bt_results, THRESHOLDS)
location_common_metrics = avg_common_metrics(location_results, THRESHOLDS)

for d_t in DATA_TYPES:
    for wnd_t in WINDOW_TYPES:
        for wnd in WINDOWS:
            generate_common_metrics_tables(eval(d_t + '_common_metrics'), d_t,
wnd_t, WINDOWS)
```

Приложение Г

Результаты тестирования моделей машинного обучения

В приложении представлены таблицы, содержащие результаты тестирования нескольких обученных моделей. В таблицах В.1, В.2, В.3, В.4, В.5, В.6 представлены результаты, полученные на этапе кросс-валидации, а в таблицах В.7, В.8, В.9, В.10, В.11, В.12 — на этапе финального тестирования.

Таблица В.1 — Усреднённые значения метрики accuracy для модуля ВТ на этапе кросс-валидации для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
Accuracy	5	0.879	0.865	0.862	0.871
	10	0.885	0.868	0.859	0.881
	30	0.892	0.879	0.881	0.898
	60	0.903	0.883	0.879	0.929
	90	0.904	0.891	0.892	0.934
	120	0.904	0.901	0.899	0.942
	240	0.905	0.904	0.901	0.938
	600	0.910	0.907	0.906	0.943

Таблица В.2 — Усреднённые значения метрики accuracy для модуля ВТ на этапе кросс-валидации для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
Accuracy	5	0.863	0.851	0.849	0.879
	10	0.874	0.863	0.865	0.892
	30	0.888	0.871	0.868	0.903
	60	0.896	0.874	0.867	0.921
	90	0.901	0.884	0.881	0.922
	120	0.902	0.881	0.878	0.929
	240	0.907	0.892	0.894	0.934
	600	0.911	0.902	0.901	0.937

Таблица В.3 — Усреднённые значения метрики accuracy для модуля WIFI на этапе кросс-валидации для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
Accuracy	5	0.912	0.922	0.924	0.
	10	0.925	0.933	0.929	0.
	30	0.932	0.941	0.942	0.
	60	0.936	0.947	0.949	0.912
	90	0.937	0.942	0.937	0.908
	120	0.932	0.951	0.950	0.904
	240	0.939	0.953	0.948	0.909
	600	0.940	0.956	0.952	0.915

Таблица В.4 — Усреднённые значения метрики accuracy для модуля WIFI на этапе кросс-валидации для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
Accuracy	5	0.893	0.910	0.898	0.855
	10	0.906	0.918	0.909	0.862
	30	0.911	0.922	0.912	0.864
	60	0.915	0.927	0.916	0.878
	90	0.914	0.930	0.931	0.869
	120	0.921	0.944	0.938	0.877
	240	0.932	0.946	0.950	0.879
	600	0.934	0.951	0.947	0.883

Таблица В.5 — Усреднённые значения метрики accuracy для модуля LOCATION на этапе кросс-валидации для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
Accuracy	5	0.932	0.930	0.930	0.964
	10	0.940	0.939	0.939	0.972
	30	0.943	0.945	0.945	0.979
	60	0.951	0.948	0.948	0.987
	90	0.951	0.949	0.949	0.987
	120	0.958	0.954	0.954	0.991
	240	0.959	0.957	0.957	0.992
	600	0.962	0.965	0.965	0.993

Таблица В.6 — Усреднённые значения метрики accuracy для модуля LOCATION на этапе кросс-валидации для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
Accuracy	5	0.922	0.923	0.922	0.932
	10	0.929	0.924	0.928	0.937
	30	0.932	0.935	0.938	0.942
	60	0.938	0.938	0.929	0.961
	90	0.938	0.939	0.931	0.952
	120	0.937	0.939	0.933	0.949
	240	0.941	0.943	0.938	0.953
	600	0.945	0.946	0.952	0.963

Таблица В.7 — Усреднённые значения метрик AUC-ROC и EER для модуля BT на этапе финального тестирования для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
AUC-ROC	5	0.798	0.799	0.795	0.699
	10	0.807	0.804	0.803	0.706
	30	0.812	0.805	0.807	0.711
	60	0.819	0.815	0.813	0.717
	90	0.841	0.832	0.824	0.721
	120	0.862	0.880	0.882	0.717
	240	0.873	0.898	0.893	0.723
	600	0.881	0.912	0.899	0.739
EER	5	0.269	0.256	0.257	0.271
	10	0.271	0.248	0.240	0.237
	30	0.263	0.233	0.235	0.207
	60	0.200	0.226	0.232	0.192
	90	0.259	0.216	0.223	0.209
	120	0.234	0.178	0.175	0.305
	240	0.212	0.163	0.171	0.243
	600	0.174	0.152	0.168	0.184

Таблица В.8 — Усреднённые значения метрик AUC-ROC и EER для модуля BT на этапе финального тестирования для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
AUC-ROC	5	0.832	0.889	0.884	0.698
	10	0.843	0.901	0.912	0.706
	30	0.861	0.911	0.910	0.717
	60	0.863	0.916	0.909	0.719
	90	0.895	0.947	0.929	0.752
	120	0.905	0.946	0.938	0.749
	240	0.911	0.948	0.951	0.754
	600	0.923	0.947	0.934	0.763
EER	5	0.271	0.157	0.159	0.458
	10	0.259	0.141	0.137	0.442
	30	0.254	0.131	0.138	0.437
	60	0.242	0.128	0.132	0.428
	90	0.18	0.103	0.103	0.406
	120	0.163	0.104	0.104	0.399
	240	0.161	0.102	0.102	0.382
	600	0.132	0.093	0.093	0.361

Таблица В.9 — Усреднённые значения метрик AUC-ROC и EER для модуля WIFI на этапе финального тестирования для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
AUC-ROC	5	0.904	0.921	0.902	0.864
	10	0.907	0.925	0.931	0.869
	30	0.916	0.934	0.932	0.881
	60	0.923	0.940	0.933	0.884
	90	0.932	0.945	0.921	0.888
	120	0.941	0.950	0.942	0.891
	240	0.945	0.952	0.947	0.899
	600	0.947	0.957	0.949	0.902
EER	5	0.155	0.173	0.154	0.184
	10	0.147	0.123	0.132	0.171
	30	0.142	0.118	0.134	0.167
	60	0.135	0.114	0.123	0.164
	90	0.127	0.108	0.115	0.156
	120	0.116	0.105	0.100	0.156
	240	0.107	0.990	0.980	0.153
	600	0.098	0.920	0.970	0.143

Таблица B.10 — Усреднённые значения метрик AUC-ROC и EER для модуля WIFI на этапе финального тестирования для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	Logistic Regression
AUC-ROC	5	0.929	0.943	0.934	0.855
	10	0.933	0.962	0.922	0.872
	30	0.942	0.967	0.943	0.899
	60	0.958	0.971	0.953	0.898
	90	0.964	0.977	0.978	0.907
	120	0.964	0.981	0.963	0.909
	240	0.967	0.983	0.973	0.913
	600	0.972	0.988	0.964	0.922
EER	5	0.113	0.102	0.112	0.169
	10	0.095	0.089	0.084	0.157
	30	0.091	0.073	0.078	0.143
	60	0.082	0.065	0.056	0.137
	90	0.069	0.047	0.053	0.133
	120	0.072	0.043	0.052	0.128
	240	0.068	0.038	0.065	0.136
	600	0.059	0.029	0.058	0.114

Таблица B.11 — Усреднённые значения метрик AUC-ROC и EER для модуля LOCATION на этапе финального тестирования для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
AUC-ROC	5	0.911	0.898	0.901	0.852
	10	0.913	0.909	0.914	0.872
	30	0.910	0.918	0.910	0.864
	60	0.929	0.940	0.948	0.889
	90	0.942	0.943	0.952	0.871
	120	0.944	0.952	0.959	0.883
	240	0.953	0.948	0.958	0.894
	600	0.951	0.954	0.959	0.900
EER	5	0.143	0.189	0.164	0.193
	10	0.144	0.175	0.178	0.182
	30	0.139	0.156	0.159	0.174
	60	0.130	0.132	0.143	0.167
	90	0.131	0.122	0.124	0.161
	120	0.123	0.107	0.111	0.155
	240	0.117	0.108	0.106	0.156
	600	0.104	0.990	0.102	0.134

Таблица В.12 — Усреднённые значения метрик AUC-ROC и EER для модуля LOCATION на этапе финального тестирования для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	LogisticRegression
AUC-ROC	5	0.922	0.939	0.928	0.823
	10	0.928	0.942	0.938	0.845
	30	0.929	0.956	0.944	0.867
	60	0.937	0.968	0.959	0.877
	90	0.948	0.978	0.967	0.898
	120	0.956	0.985	0.971	0.907
	240	0.964	0.984	0.968	0.917
	600	0.972	0.988	0.979	0.922
EER	5	0.113	0.109	0.116	0.187
	10	0.097	0.112	0.098	0.169
	30	0.098	0.069	0.089	0.155
	60	0.087	0.063	0.074	0.159
	90	0.074	0.056	0.066	0.137
	120	0.081	0.049	0.042	0.119
	240	0.063	0.045	0.054	0.139
	600	0.065	0.037	0.044	0.123

Приложение Д

Исходный код сценариев, использованных для автоматизированного формирования тестовых выборок элементарных событий

В данном приложении представлен исходный код сценариев на языке Python, которые использовались для формирования выборок для тестирования предложенного в работе метода аутентификации.

Сценарий для случайного выбора и генерации наборов событий продолжительностью 15 минут events_generator.py.

```
import pandas as pd
import random as rnd
import os
from datetime import datetime as dt, timedelta as td
import argparse

POWER_EVENTS_FILE_NAME = "power.data"
GENERATED_FLOW_NAME = "flow"

SAMPLES_COUNT = 10
DURATION = 15


def convert_timestamp_for_file(file_path):
    print("\tConvert: ", file_path)

    with open(file_path, 'r') as f:
        lines = f.readlines()

    if len(lines) == 0:
        return

    timestamps = [dt.strptime(x.split(';')[0], '%d.%m.%Y_%H:%M:%S.%f') for x in
lines]
    users = [int(x.split(';')[-1].replace('\n', '')) for x in lines]

    df = pd.DataFrame({'timestamp': timestamps, 'user': users})
    df.index = pd.DatetimeIndex(df.timestamp)

    df['delta'] = (df.timestamp.shift(-1) - df.timestamp).shift(1)

    idx = df.index[df['user'].diff() != 0][-1]
    df.at[idx, 'delta'] = pd.Timedelta(0)

    time_array = []
    current_timestamp = [df.timestamp[0]]

    def add_value_to_array(x, time_array, current_timestamp):
        delta = x
        if pd.isnull(x):
            delta = pd.Timedelta(0)

        current_timestamp[0] += delta
        time_array.append(current_timestamp[0])
```

```
    _ = df.delta.apply(lambda x: add_value_to_array(x, time_array,
current_timestamp))

    df.timestamp = time_array
    df = df.drop(['delta'], axis=1)

    ts = [x.strftime('%d.%m.%Y_%H:%M:%S.%f') for x in df['timestamp']]
    new_lines = []
    for line, timestamp in zip(lines, ts):
        new_line = line.split(';')
        new_line[0] = timestamp
        new_lines.append(';'.join(new_line))

    with open(file_path, 'w') as of:
        of.writelines(new_lines)


def convert_timestamps(path):
    for time_dir in os.listdir(path):
        dir = os.path.join(path, time_dir)
        for user_path in os.listdir(dir):
            flow_path = os.path.join(dir, user_path, GENERATED_FLOW_NAME)
            print("Convert timestamps: ", flow_path)

            for file in os.listdir(flow_path):
                file_path = os.path.join(flow_path, file)
                if os.path.isfile(file_path):
                    convert_timestamp_for_file(file_path)


def merge_different_users_samples(path, samples_count):
    for time_dir in os.listdir(path):
        dir = os.path.join(path, time_dir)
        for user_path in os.listdir(dir):
            print("Merge samples for ", user_path)

            src_user_path = os.path.join(dir, user_path)
            out_user_path = os.path.join(src_user_path, GENERATED_FLOW_NAME)

            if os.path.exists(out_user_path) is False:
                os.makedirs(out_user_path)

            data_files = []
            for file in os.listdir(src_user_path):
                file_path = os.path.join(src_user_path, file)
                if os.path.isfile(file_path):
                    data_files.append(file_path)

            for file in data_files:
                for other_user_path in os.listdir(dir):
                    if other_user_path != user_path:
                        other_path = os.path.join(dir, other_user_path)
                        other_data_files = []
                        for f in os.listdir(other_path):
                            other_file_path = os.path.join(other_path, f)
                            if os.path.isfile(other_file_path):
                                other_data_files.append(other_file_path)

                        for other_file in other_data_files:
                            if os.path.basename(other_file).split('_')[0] ==
os.path.basename(file).split('_')[0]:
                                print("\tFile: ", file)
```

```
                                print("\tOther file: ", other_file)

                                valid_user =
os.path.basename(src_user_path).split('_')[1]
                                intruder =
os.path.basename(other_user_path).split('_')[1]

                                with open(file, 'r') as ff:
                                    valid_lines = ff.readlines()

                                with open(other_file, 'r') as ff:
                                    intruder_lines = ff.readlines()

                                valid_lines = [x.replace('\n', ';' + valid_user)
+ '\n' for x in valid_lines]
                                intruder_lines = [x.replace('\n', ';' +
intruder) + '\n' for x in intruder_lines]

                                base_name = os.path.basename(file).split('_')[0]
                                index_1 =
os.path.basename(file).split('_')[1].replace('.data', '')
                                index_2 =
os.path.basename(other_file).split('_')[1].replace('.data', '')
                                file_name = '_'.join([base_name, index_1,
index_2, intruder]) + ".data"
                                with open(os.path.join(out_user_path,
file_name), 'w') as of:
                                    of.writelines(valid_lines)
                                    of.writelines(intruder_lines)


def generate_samples_for_each_user(src_path, dst_path, samples_count, duration):
    seed = int(rnd.SystemRandom().random() * 10000)
    rnd.seed(a=seed)

    for user_data_dir in os.listdir(src_path):
        print("Generate samples for ", user_data_dir)

        src_user_path = os.path.join(src_path, user_data_dir)
        out_user_path = os.path.join(dst_path, user_data_dir)

        if os.path.exists(out_user_path) is False:
            os.makedirs(out_user_path)

        data_files = []
        for file in os.listdir(src_user_path):
            file_path = os.path.join(src_user_path, file)
            if os.path.isfile(file_path) and file != POWER_EVENTS_FILE_NAME:
                data_files.append(file_path)

        for file in data_files:
            print("\tFile: ", file)
            df = pd.read_csv(file, sep='\n', index_col=False, header=None,
low_memory=False)

            df['timestamp'] = df[0].apply(lambda x: x.split(';')[0])
            df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))

            df.index = pd.DatetimeIndex(df.timestamp)
            df = df.sort_index()
```

```
            df = df.drop(['timestamp'], axis=1)

            for i in range(samples_count):
                print("\t\tSample ", i)
                chosen_file_name = data_files[rnd.randrange(len(data_files))]
                with open(chosen_file_name) as f:
                    lines = f.readlines()

                index = rnd.randrange(len(lines) // 5, 4 * (len(lines) // 5))
                begin_timestamp = dt.strptime(lines[index].split(';')[0],
'%d.%m.%Y_%H:%M:%S.%f')
                end_timestamp = begin_timestamp + td(minutes=duration)

                sample = df[begin_timestamp: end_timestamp]

                p = os.path.join(out_user_path,
os.path.basename(file).replace(".data", "_" + str(i) + ".data"))
                sample.to_csv(p, sep=';', header=False, index=False)

                with open(p, 'r') as f:
                    lines = f.readlines()

                lines = [x[1:-2] + '\n' for x in lines]
                with open(p, 'w') as f:
                    f.writelines(lines)


def generate_events(src_path, dst_path, samples_count, duration):
    generate_samples_for_each_user(src_path, os.path.join(dst_path,
str(duration) + "min"), samples_count, duration)


def main():
    parser = argparse.ArgumentParser(description='Events flow generator')

    parser.add_argument("--src", default=None, type=str, help="Folder with data
after filtering")
    parser.add_argument("--dst", default=None, type=str, help="Destination
folder")

    args = parser.parse_args()
    src_folder = args.src
    dst_folder = args.dst

    generate_events(src_folder, dst_folder, SAMPLES_COUNT, DURATION)
    merge_different_users_samples(dst_folder, SAMPLES_COUNT)
    convert_timestamps(dst_folder)


if __name__ == '__main__':
    main()
```

Сценарий для формирования признаков на сгенерированных выборках events_to_features.py.

```
import pandas as pd
import numpy as np
from datetime import datetime as dt
import scipy.stats as stats
from scipy.spatial import distance
```

```
from geopy.distance import distance as geodist
import argparse
import os

POWER_EVENTS_FILE_NAME = "power.data"


def generate_location_features(src_path, dst_path_rolling, dst_path_sampling,
freq):
    df = pd.read_csv(src_path, sep=';', index_col=False, header=None,
low_memory=False,
                     names=['timestamp', 'accuracy', 'altitude', 'latitude',
'longitude', 'user'])

    if len(df) == 0:
        return

    df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))

    df.index = pd.DatetimeIndex(df.timestamp)
    df = df.sort_index()

    VALID_USER = df.iloc[0]['user']
    df['events_count'] = 1

    df['accuracy'] = df['accuracy'].apply(lambda x: str(x).replace(',', '.'))
    df['altitude'] = df['altitude'].apply(lambda x: str(x).replace(',', '.'))
    df['latitude'] = df['latitude'].apply(lambda x: str(x).replace(',', '.'))
    df['longitude'] = df['longitude'].apply(lambda x: str(x).replace(',', '.'))

    df['accuracy'] = df['accuracy'].astype(float)
    df['altitude'] = df['altitude'].astype(float)
    df['latitude'] = df['latitude'].astype(float)
    df['longitude'] = df['longitude'].astype(float)

    df['prev_latitude'] = df['latitude'].shift(1)
    df['prev_longitude'] = df['longitude'].shift(1)
    df['prev_timestamp'] = df['timestamp'].shift(1)
    df['prev_altitude'] = df['altitude'].shift(1)

    def get_speed(row):
        prev_coords = (row['prev_latitude'], row['prev_longitude'])
        curr_coords = (row['latitude'], row['longitude'])
        delta = row['timestamp'] - row['prev_timestamp']
        if pd.isnull(delta):
            return np.nan
        time = abs(delta.total_seconds())
        if np.isnan(prev_coords[0]) or np.isnan(prev_coords[1]) or
np.isnan(curr_coords[0]) or np.isnan(curr_coords[1]):
            return np.nan
        if time == 0:
            return np.nan
        return geodist(curr_coords, prev_coords).meters / time

    def get_altitude_speed(row):
        prev = row['prev_altitude']
        curr = row['altitude']
        delta = row['timestamp'] - row['prev_timestamp']
        if pd.isnull(delta):
            return np.nan
        time = abs(delta.total_seconds())
```

```
        if np.isnan(prev) or np.isnan(curr):
            return np.nan
        if time == 0:
            return np.nan
        return abs(curr - prev) / time

    df['speed'] = df.apply(lambda row: get_speed(row), axis=1)
    df['altitude_speed'] = df.apply(lambda row: get_altitude_speed(row), axis=1)

    df = df.drop(['prev_latitude', 'prev_longitude', 'prev_altitude',
'timestamp', 'prev_timestamp'], axis=1)

    def kurt(col):
        return stats.kurtosis(col)

    def user_agg(col):
        if (col == VALID_USER).all():
            return 1
        else:
            return 0

    common_funcs_list = ['mean', 'var', 'median', 'skew', kurt, 'std']

    agg_dict = {
        'accuracy': common_funcs_list,
        'speed': common_funcs_list,
        'altitude_speed': common_funcs_list,
        'events_count': 'sum',
        'user': user_agg
    }

    df_sampling = df.groupby(pd.Grouper(freq=freq)).agg(agg_dict)

    df_sampling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                           for (high_level_name, low_level_name) in
df_sampling.columns.values]

    df_rolling = df.rolling(freq, min_periods=1, center=False).agg(agg_dict)

    df_rolling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                          for (high_level_name, low_level_name) in
df_rolling.columns.values]

    df_sampling = df_sampling.dropna()
    df_sampling = df_sampling.fillna(0)

    df_rolling = df_rolling.dropna()
    df_rolling = df_rolling.fillna(0)

    df_sampling.to_csv(dst_path_sampling)
    df_rolling.to_csv(dst_path_rolling)


def generate_wifi_features(src_path, src_path_conn, dst_path_rolling,
dst_path_sampling, freq, window):
    df = pd.read_csv(src_path, sep=';', index_col=False, header=None,
low_memory=False,
                     names=['timestamp', 'uuid', 'bssid', 'chwidth', 'freq',
'level', 'user'])

    if len(df) == 0:
        return
```

```
    df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))
    df.index = pd.DatetimeIndex(df.timestamp)
    df = df.sort_index()

    VALID_USER = df.iloc[0]['user']
    df['events_count'] = 1

    df = df.drop(['timestamp', 'chwidth'], axis=1)

    bssid_map = {bssid.replace(' ', ''): idx for bssid, idx in
zip(df.bssid.unique(), range(len(df.bssid.unique())))}

    df.bssid = df.bssid.apply(lambda x: str(x).replace(' ', ''))
    df.level = df.level.apply(lambda x: str(x).replace(' ', ''))
    df.freq = df.freq.apply(lambda x: str(x).replace(' ', ''))

    df['bssid_level'] = df[['bssid', 'level']].agg(','.join, axis=1)
    df['count'] = 1

    def user_agg(col):
        if (col == VALID_USER).all():
            return 1
        else:
            return 0

    def agg_string_join(col):
        col = col.apply(lambda x: str(x))
        return col.str.cat(sep=',').replace(' ', '')

    def agg_bssid_col(col):
        array_len = len(bssid_map)
        array = np.zeros(array_len, dtype='float')

        def fill_array(x):
            tmp = x.split(',')
            bssid = tmp[0]
            level = float(tmp[1])
            array[bssid_map[bssid.replace(' ', '')]] = level
            return

        col.apply(lambda x: fill_array(x))
        return np.array2string(array, separator=',').replace(' ', '')[1:-1]

    all_func_dicts_quantum = {'freq': agg_string_join, 'level': agg_string_join,
'bssid_level': agg_bssid_col,
                              'count': 'sum', 'events_count': 'sum', 'user':
user_agg}

    df_quantum = df.groupby(['timestamp', 'uuid'],
as_index=True).agg(all_func_dicts_quantum)

    df_quantum = df_quantum.reset_index()
    df_quantum.index = pd.DatetimeIndex(df_quantum.timestamp)

    df_quantum = df_quantum[df_quantum['count'] != 0]

    df_conn = pd.read_csv(src_path_conn, sep=';', index_col=False, header=None,
low_memory=False,
                          names=['timestamp', '1', 'bssid', '2', '3', '4', '5',
'level', '6'])
```

```
    if len(df_conn) != 0:
        df_conn['timestamp'] = df_conn['timestamp'].apply(lambda x:
dt.strptime(x, '%d.%m.%Y_%H:%M:%S.%f'))
        df_conn.index = pd.DatetimeIndex(df_conn.timestamp)
        df_conn = df_conn.sort_index()

        def get_level_from_row(row):
            bssid = df_conn.iloc[df_conn.index.get_loc(row.name,
method='nearest')]['bssid']
            if str(bssid) == 'nan' or str(bssid) == 'null' or str(bssid) == '':
                return 0

            level = df_conn.iloc[df_conn.index.get_loc(row.name,
method='nearest')]['level']
            time = df_conn.iloc[df_conn.index.get_loc(row.name,
method='nearest')]['timestamp']
            return level if abs((time - row.name).total_seconds()) <= 10 else 0

        df_conn = df_conn.loc[~df_conn.index.duplicated(keep='first')]
        df_quantum['conn_level'] = df_quantum.apply(lambda row:
get_level_from_row(row), axis=1)
    else:
        df_quantum['conn_level'] = 0

    def string2array(string):
        try:
            array = np.fromstring(string, sep=',')
            return array
        except:
            return np.nan

    def to_ones_array(array):
        try:
            array[array != 0] = 1
            return array
        except:
            return np.nan

    def get_len(obj):
        try:
            length = len(obj)
            return length
        except:
            return np.nan

    def get_occured_nets_count(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        intersection = np.logical_and(curr, prev)
        diff = np.logical_and(curr, np.logical_not(intersection))

        if (np.count_nonzero(np.logical_or(prev, curr)) == 0):
            return 0

        return np.count_nonzero(diff) / np.count_nonzero(np.logical_or(prev,
curr))

    def get_disappeared_nets_count(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        intersection = np.logical_and(curr, prev)
```

```
        diff = np.logical_and(prev, np.logical_not(intersection))

        if (np.count_nonzero(np.logical_or(prev, curr)) == 0):
            return 0

        return np.count_nonzero(diff) / np.count_nonzero(np.logical_or(prev,
curr))

    def get_jaccard_index(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        return distance.jaccard(prev, curr)

    def get_occur_speed(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        return np.linalg.norm(prev - curr) / np.sqrt(get_len(prev))

    def get_level_speed(row, prev_col, curr_col):
        prev = string2array(row[prev_col])
        curr = string2array(row[curr_col])
        return np.linalg.norm(prev - curr) / np.sqrt(get_len(prev))

    def calc_single_cols_in_window(df, col, new_col, window, func):
        def func_wrapper(func, row, prev_col, curr_col):
            delta = row.timestamp - row.prev_timestamp
            if pd.isnull(delta):
                delta = 0
            else:
                delta = abs(delta.total_seconds())
            if delta > 10 * 60:
                return np.nan
            else:
                return func(row, prev_col_name, col)

        new_cols = []

        for i in range(window):
            prev_col_name = "_".join(['prev', col, str(i + 1)])
            new_col_name = "_".join([new_col, str(i + 1)])

            df['prev_timestamp'] = df.timestamp.shift(i + 1)
            df[prev_col_name] = df[col].shift(i + 1)
            df[new_col_name] = df.apply(lambda row: func_wrapper(func, row,
prev_col_name, col), axis=1)
            df = df.drop(prev_col_name, axis=1)
            df = df.drop('prev_timestamp', axis=1)
            new_cols.append(new_col_name)

        df["_".join([new_col, 'mean'])] = df[new_cols].mean(axis=1)
        df["_".join([new_col, 'median'])] = df[new_cols].median(axis=1)
        df["_".join([new_col, 'var'])] = df[new_cols].var(axis=1)

        return df

    occur_and_level_columns_map = [
        ("bssid_level", "occured_nets_count", window, get_occured_nets_count),
        ("bssid_level", "disappeared_nets_count", window,
get_disappeared_nets_count),
        ("bssid_level", "jaccard_index", window, get_jaccard_index),
        ("bssid_level", "occur_speed", window, get_occur_speed),
        ("bssid_level", "level_speed", window, get_level_speed)
```

```
    ]

    for (col, new_col, wnd, func) in occur_and_level_columns_map:
        df_quantum = calc_single_cols_in_window(df_quantum, col, new_col, wnd,
func)

    def get_conn_level_speed(row, prev_col, curr_col):
        return row[curr_col] - row[prev_col]

    single_columns_map = [
        ("conn_level", "conn_level_speed", window, get_conn_level_speed),
        ("count", "count_speed", window, get_conn_level_speed)
    ]

    for (col, new_col, wnd, func) in single_columns_map:
        df_quantum = calc_single_cols_in_window(df_quantum, col, new_col, wnd,
func)

    def agg_str(col):
        #     all_freq = col.str.cat(sep=',')
        return string2array(col)

    def str_mean(col):
        array = agg_str(col)
        if str(array) == 'nan':
            return 0
        return np.mean(array)

    def mean(col):
        return np.mean(col)

    def var(col):
        return np.var(col)

    def median(col):
        return np.median(col)

    def skew(col):
        return stats.skew(col)

    def kurt(col):
        return stats.kurtosis(col)

    df_quantum['freq'] = df_quantum.apply(lambda row: str_mean(row['freq']),
axis=1)
    df_quantum['level'] = df_quantum.apply(lambda row: str_mean(row['level']),
axis=1)

    cols_for_drop = []
    names = [
        "occured_nets_count",
        "disappeared_nets_count",
        "jaccard_index",
        "occur_speed",
        "count_speed",
        "conn_level_speed",
        "level_speed",
        "count_speed"
    ]

    for i in range(1, window + 1):
        for name in names:
```

```
            cols_for_drop.append('_'.join([name, str(i)]))

    df_quantum = df_quantum.drop(['bssid_level', 'timestamp', 'uuid'], axis=1)
    df_quantum = df_quantum.drop(cols_for_drop, axis=1)

    def user_agg(col):
        if (col == 1).all():
            return 1
        else:
            return 0

    common_cols = [x for x in df_quantum.columns[0:6] if x != 'user' and x !=
'events_count']
    speed_acc_cols = df_quantum.columns[6:]

    common_funcs_list = [mean, var, median, skew, kurt]
    special_funcs_list = [mean, pd.DataFrame.mad, skew]

    common_cols_map = {col: common_funcs_list for col in common_cols}
    speed_acc_cols_map = {col: special_funcs_list for col in speed_acc_cols}

    additional = {'user': user_agg, 'events_count': 'sum'}

    agg_dict = common_cols_map
    agg_dict.update(speed_acc_cols_map)
    agg_dict.update(additional)

    df_quantum[speed_acc_cols] = df_quantum[speed_acc_cols].apply(pd.to_numeric)

    df_sampling = df_quantum.groupby(pd.Grouper(freq=freq)).agg(agg_dict)

    df_rolling = df_quantum.rolling(freq, min_periods=1,
center=False).agg(agg_dict)

    df_sampling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                           for (high_level_name, low_level_name) in
df_sampling.columns.values]

    df_rolling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                          for (high_level_name, low_level_name) in
df_rolling.columns.values]

    df_sampling = df_sampling.dropna()
    df_sampling = df_sampling.fillna(0)

    df_rolling = df_rolling.dropna()
    df_rolling = df_rolling.fillna(0)

    df_sampling.to_csv(dst_path_sampling)
    df_rolling.to_csv(dst_path_rolling)


def generate_bt_features(src_path, src_path_le, dst_path_rolling,
dst_path_sampling, freq, window):
    df = pd.read_csv(src_path, sep=';', index_col=False, header=None,
low_memory=False,
                     names=['timestamp', 'action', 'bssid', 'major_class',
'class', 'bond_state', 'type', 'user'])

    if len(df) == 0:
        return
```

```
    df = df[df['action'] == 'android.bluetooth.device.action.FOUND']

    df['timestamp'] = df['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))
    df.index = pd.DatetimeIndex(df.timestamp)
    df = df.sort_index()

    VALID_USER = df.iloc[0]['user']
    df['events_count'] = 1

    df = df.drop(['timestamp', 'action', 'class', 'major_class', 'bond_state',
'type'], axis=1)

    bssid_map = {bssid.replace(' ', ''): idx for bssid, idx in
zip(df.bssid.unique(), range(len(df.bssid.unique())))}

    df.bssid = df.bssid.apply(lambda x: str(x).replace(' ', ''))
    df['count'] = 1

    def user_agg(col):
        if (col == VALID_USER).all():
            return 1
        else:
            return 0

    def agg_string_join(col):
        col = col.apply(lambda x: str(x))
        return col.str.cat(sep=',').replace(' ', '')

    def agg_bssid_col(col):
        array_len = len(bssid_map)
        array = np.zeros(array_len, dtype='int8')

        def fill_array(bssid):
            array[bssid_map[bssid.replace(' ', '')]] = 1
            return

        col.apply(lambda x: fill_array(x))
        return np.array2string(array, separator=',').replace(' ', '')[1:-1]

    one_hot_columns_count = 0
    for col in df.columns:
        if col.find('one_hot') != -1:
            one_hot_columns_count += 1

    cat_columns = df.columns[1:1 + one_hot_columns_count]
    cat_columns_map = {col: 'mean' for col in cat_columns}

    all_func_dicts_quantum = {'bssid': agg_bssid_col, 'count': 'sum', 'user':
user_agg, 'events_count': 'sum'}
    all_func_dicts_quantum.update(cat_columns_map)

    df_quantum = df.groupby(pd.Grouper(freq='5s'),
as_index=True).agg(all_func_dicts_quantum)

    df_quantum = df_quantum.reset_index()
    df_quantum.index = pd.DatetimeIndex(df_quantum.timestamp)

    df_quantum = df_quantum.dropna()

    df_le = pd.read_csv(src_path_le, sep=';', index_col=False, header=None,
low_memory=False,
```

```
                        names=['timestamp', '1', '2', '3', 'level', '3',
'connectable', '4'])

    if len(df_le) != 0:
        df_le['timestamp'] = df_le['timestamp'].apply(lambda x: dt.strptime(x,
'%d.%m.%Y_%H:%M:%S.%f'))
        df_le = df_le.drop(df_le.columns.difference(['connectable', 'timestamp',
'level']), axis=1)
        df_le.index = pd.DatetimeIndex(df_le.timestamp)
        df_le = df_le.sort_index()

        df_le['connectable'] = df_le['connectable'].apply(lambda x: 1 if
str(x).lower() == 'true' else 0)

        df_le = df_le.groupby(pd.Grouper(freq='5s'),
as_index=True).agg({'level': 'mean', 'connectable': 'mean'})
        df_le = df_le.dropna()

        def get_le_conn_status_from_row(row):
            conn = df_le.iloc[df_le.index.get_loc(row.name,
method='nearest')]['connectable']
            time = df_le.iloc[df_le.index.get_loc(row.name,
method='nearest')].name
            return conn if abs((time - row.name).total_seconds()) < 10 else 0

        def get_le_level_from_row(row):
            level = df_le.iloc[df_le.index.get_loc(row.name,
method='nearest')]['level']
            time = df_le.iloc[df_le.index.get_loc(row.name,
method='nearest')].name
            return level if abs((time - row.name).total_seconds()) < 10 else 0

        df_le = df_le.loc[~df_le.index.duplicated(keep='first')]

        df_quantum['le_connectable'] = df_quantum.apply(lambda row:
get_le_conn_status_from_row(row), axis=1)
        df_quantum['le_level'] = df_quantum.apply(lambda row:
get_le_level_from_row(row), axis=1)

    else:
        df_quantum['le_connectable'] = 0
        df_quantum['le_level'] = 0

    def string2array(string):
        try:
            array = np.fromstring(string, sep=',')
            return array
        except:
            return np.nan

    def to_ones_array(array):
        try:
            array[array != 0] = 1
            return array
        except:
            return np.nan

    def get_len(obj):
        try:
            length = len(obj)
            return length
        except:
```

```
            return np.nan

    def get_occured_nets_count(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        intersection = np.logical_and(curr, prev)
        diff = np.logical_and(curr, np.logical_not(intersection))

        if (np.count_nonzero(np.logical_or(prev, curr)) == 0):
            return 0

        return np.count_nonzero(diff) / np.count_nonzero(np.logical_or(prev,
curr))

    def get_disappeared_nets_count(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        intersection = np.logical_and(curr, prev)
        diff = np.logical_and(prev, np.logical_not(intersection))

        if (np.count_nonzero(np.logical_or(prev, curr)) == 0):
            return 0

        return np.count_nonzero(diff) / np.count_nonzero(np.logical_or(prev,
curr))

    def get_jaccard_index(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        return distance.jaccard(prev, curr)

    def get_occur_speed(row, prev_col, curr_col):
        prev = to_ones_array(string2array(row[prev_col]))
        curr = to_ones_array(string2array(row[curr_col]))
        return np.linalg.norm(prev - curr) / np.sqrt(get_len(prev))

    def calc_single_cols_in_window(df, col, new_col, window, func):
        def func_wrapper(func, row, prev_col, curr_col):
            delta = row.timestamp - row.prev_timestamp
            if pd.isnull(delta):
                delta = 0
            else:
                delta = abs(delta.total_seconds())
            if delta > 10 * 60:
                return np.nan
            else:
                return func(row, prev_col_name, col)

        new_cols = []

        for i in range(window):
            prev_col_name = "_".join(['prev', col, str(i + 1)])
            new_col_name = "_".join([new_col, str(i + 1)])

            df.loc[:, 'prev_timestamp'] = df.timestamp.shift(i + 1)
            df.loc[:, prev_col_name] = df[col].shift(i + 1)
            df.loc[:, new_col_name] = df.apply(lambda row: func_wrapper(func,
row, prev_col_name, col), axis=1)
            df = df.drop(prev_col_name, axis=1)
            df = df.drop('prev_timestamp', axis=1)
            new_cols.append(new_col_name)
```

```
        df.loc[:, "_".join([new_col, 'mean'])] = df[new_cols].mean(axis=1)
        df.loc[:, "_".join([new_col, 'median'])] = df[new_cols].median(axis=1)
        df.loc[:, "_".join([new_col, 'var'])] = df[new_cols].var(axis=1)

        return df

    occur_and_level_columns_map = [
        ("bssid", "occured_devices_count", window, get_occured_nets_count),
        ("bssid", "disappeared_devices_count", window,
get_disappeared_nets_count),
        ("bssid", "jaccard_index", window, get_jaccard_index),
        ("bssid", "occur_speed", window, get_occur_speed)
    ]

    for (col, new_col, wnd, func) in occur_and_level_columns_map:
        df_quantum = calc_single_cols_in_window(df_quantum, col, new_col, wnd,
func)

    def get_conn_level_speed(row, prev_col, curr_col):
        return row[curr_col] - row[prev_col]

    single_columns_map = [
        ("count", "count_speed", window, get_conn_level_speed)
    ]

    for (col, new_col, wnd, func) in single_columns_map:
        df_quantum = calc_single_cols_in_window(df_quantum, col, new_col, wnd,
func)

    def agg_str(col):
        all_freq = col.str.cat(sep=',')
        return string2array(all_freq)

    def mean(col):
        return np.mean(col)

    def var(col):
        return np.var(col)

    def median(col):
        return np.median(col)

    def skew(col):
        return stats.skew(col)

    def kurt(col):
        return stats.kurtosis(col)

    cols_for_drop = []
    names = [
        "occured_devices_count",
        "disappeared_devices_count",
        "jaccard_index",
        "occur_speed",
        "count_speed"
    ]

    for i in range(1, window + 1):
        for name in names:
            cols_for_drop.append('_'.join([name, str(i)]))

    df_quantum = df_quantum.drop(['bssid', 'timestamp'], axis=1)
```

```
    df_quantum = df_quantum.drop(cols_for_drop, axis=1)

    def user_agg(col):
        if (col == 1).all():
            return 1
        else:
            return 0

    common_cols = [x for x in df_quantum.columns[:one_hot_columns_count + 5] if
x != 'user' and x != 'events_count']
    speed_acc_cols = df_quantum.columns[one_hot_columns_count + 5:]

    common_funcs_list = [mean, var, median, skew, kurt]
    special_funcs_list = [mean, pd.DataFrame.mad, skew]

    common_cols_map = {col: common_funcs_list for col in common_cols}
    speed_acc_cols_map = {col: special_funcs_list for col in speed_acc_cols}

    additional = {'user': user_agg, 'events_count': 'sum'}

    agg_dict = common_cols_map
    agg_dict.update(speed_acc_cols_map)
    agg_dict.update(additional)

    df_quantum[speed_acc_cols] = df_quantum[speed_acc_cols].apply(pd.to_numeric)

    df_sampling = df_quantum.groupby(pd.Grouper(freq=freq)).agg(agg_dict)

    df_rolling = df_quantum.rolling(freq, min_periods=1,
center=False).agg(agg_dict)

    df_sampling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                           for (high_level_name, low_level_name) in
df_sampling.columns.values]

    df_rolling.columns = ["_".join([str(high_level_name), str(low_level_name)])
                          for (high_level_name, low_level_name) in
df_rolling.columns.values]

    df_sampling = df_sampling.dropna()
    df_sampling = df_sampling.fillna(0)

    df_rolling = df_rolling.dropna()
    df_rolling = df_rolling.fillna(0)

    df_sampling.to_csv(dst_path_sampling)
    df_rolling.to_csv(dst_path_rolling)


def generate_features(src, dst, freq, window):
    for time_dir in os.listdir(src):
        dir = os.path.join(src, time_dir)
        for user_path in os.listdir(dir):
            src_user_path = os.path.join(dir, user_path, "flow")
            out_user_sampling_path = os.path.join(dst, time_dir, "sampling",
freq, user_path)
            out_user_rolling_path = os.path.join(dst, time_dir, "rolling", freq,
user_path)

            if os.path.exists(out_user_sampling_path) is False:
                os.makedirs(out_user_sampling_path)
```

```
            if os.path.exists(out_user_rolling_path) is False:
                os.makedirs(out_user_rolling_path)

            print("Generate features for ", user_path)

            for file in os.listdir(src_user_path):
                if os.path.isfile(os.path.join(src_user_path, file)) and
file.find(POWER_EVENTS_FILE_NAME) == -1:
                    prefix = file[:-11]
                    postfix = file[-11:]

                    # # # # # # # # # # # #
                    # if postfix[1] == '0' and postfix[3] == '0':
                    # # # # # # # # # # # #

                    if prefix == "base_wifi":
                        wifi_path = os.path.join(src_user_path, prefix +
postfix)
                        wifi_conn = os.path.join(src_user_path, "conn_wifi" +
postfix)
                        wifi_sampling_out = os.path.join(out_user_sampling_path,
"wifi" + postfix + ".csv")
                        wifi_rolling_out = os.path.join(out_user_rolling_path,
"wifi" + postfix + ".csv")

                        print("\tGenerate WIFI: ", wifi_path)
                        generate_wifi_features(wifi_path, wifi_conn,
wifi_rolling_out, wifi_sampling_out, freq, window)

                    if prefix == "base_bt":
                        bt_path = os.path.join(src_user_path, prefix + postfix)
                        bt_le = os.path.join(src_user_path, "le_bt" + postfix)
                        bt_sampling_out = os.path.join(out_user_sampling_path,
"bt" + postfix + ".csv")
                        bt_rolling_out = os.path.join(out_user_rolling_path,
"bt" + postfix + ".csv")

                        print("\tGenerate BT: ", bt_path)
                        generate_bt_features(bt_path, bt_le, bt_rolling_out,
bt_sampling_out, freq, window)

                    if prefix == "location":
                        location_path = os.path.join(src_user_path, prefix +
postfix)
                        location_sampling_out =
os.path.join(out_user_sampling_path, "location" + postfix + ".csv")
                        location_rolling_out =
os.path.join(out_user_rolling_path, "location" + postfix + ".csv")

                        print("\tGenerate LOCATION: ", location_path)
                        generate_location_features(location_path,
location_rolling_out, location_sampling_out, freq)


def main():
    parser = argparse.ArgumentParser(description='Features generator for
events')

    parser.add_argument("--src", default=None, type=str, help="Folder with
data")
    parser.add_argument("--dst", default=None, type=str, help="Destination
folder")
```

```
    parser.add_argument("--wnd", default=None, type=str, help="Window size")

    args = parser.parse_args()
    src_folder = args.src
    dst_folder = args.dst
    freq = args.wnd

    OTHER_WINDOW_SIZE = 3

    generate_features(src_folder, dst_folder, freq, OTHER_WINDOW_SIZE)


if __name__ == '__main__':
    main()
```