ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

Кафедра
«Криптология и кибербезопасность»

Отчет

о научно-исследовательской работе

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

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

Научный руководитель: Еремин А.В.
подпись, дата

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

Москва – 2020

РЕФЕРАТ

Отчёт 79 с., 20 рис., 6 табл., 74 источн., 4 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 неявной аутентификации пользователя мобильного устройства на основе анализа нескольких поведенческих характеристик.

В 2020 году мобильные устройства используются повсеместно, поэтому актуальна проблема обеспечения защиты устройств от несанкционированного доступа. В работе рассмотрены различные подходы к аутентификации, а также методы поведенческой биометрии и предложена многомодульная система аутентификации.

В ходе работы было разработано мобильное приложение под операционную систему Android для сбора данных поведения пользователей.

На основе собранных данных проведено формирование признаков для нескольких модулей системы и тестирование моделей машинного обучения на сформированных выборках.

Были получены значения метрик качества для нескольких алгоритмов классификации. Лучшие значения показал алгоритм случайного леса, для него F-мера составила 0.916-0.993. Также было проведено сравнение метрик, полученных для разных модулей при разных способах формирования признаков.

Полученные результаты будут использованы в дальнейшей работе по созданию непрерывного метода аутентификации пользователей на основе нескольких поведенческих характеристик.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

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

API — программный интерфейс приложения (application programming interface).

AUC-ROC — площадь под ROC-кривой (area under the curve).

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 предложена метод непрерывной аутентификации на основе биометрического профиля пользователя, основанного на данных о сетях WiFi и устройствах Bluetooth, находящихся вокруг устройства, на данных о геопозиции пользователя и статистике использования мобильных приложений. Также предложена архитектура и сценарии работы мобильного приложения, реализующего предложенную систему.

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 приведены результаты тестирования нескольких моделей машинного обучения, использованных для решения задачи классификации пользователей на легальных и несанкционированных.

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.

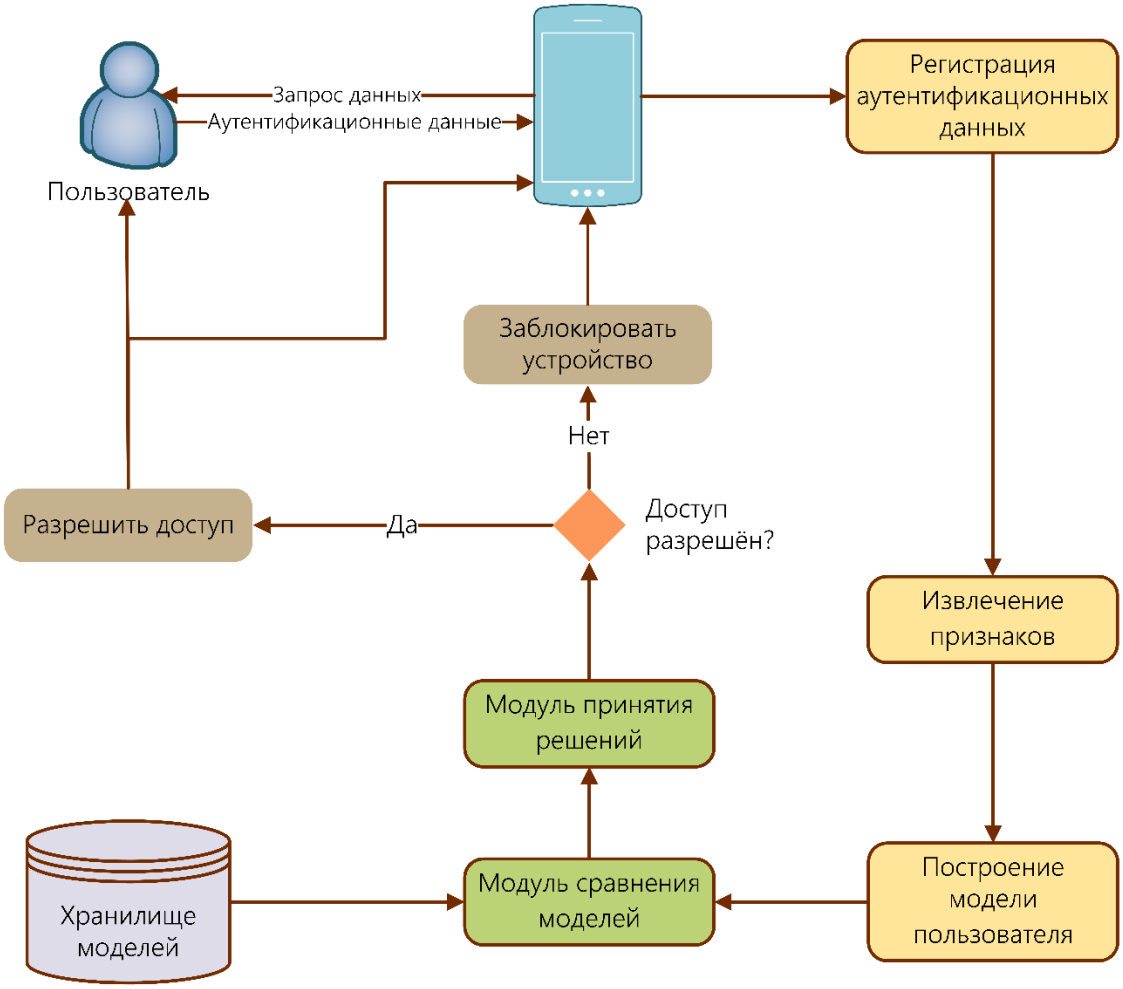

Рисунок 1 — Схема проведения биометрической аутентификации пользователя мобильного устройства

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.

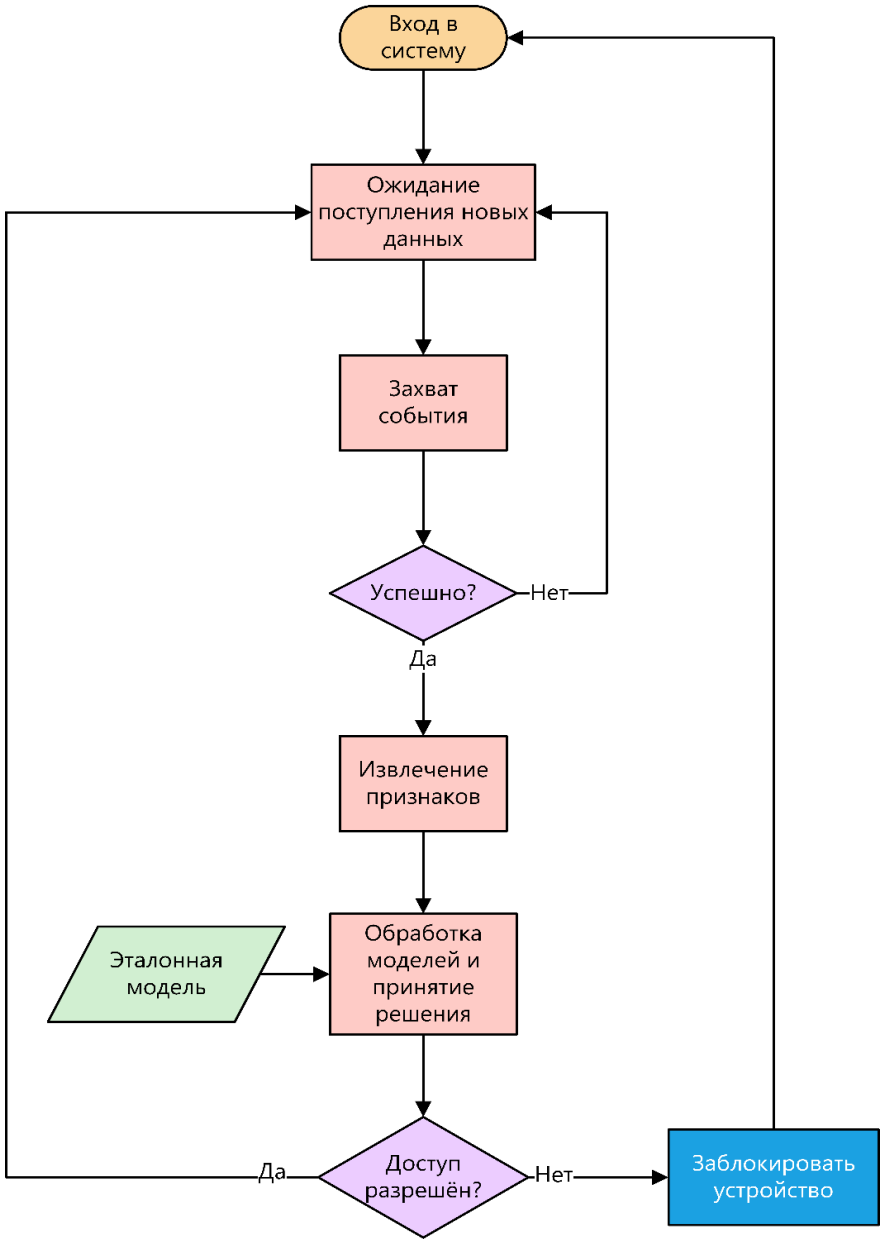

Рисунок 2 — Схема процесса непрерывной аутентификации по поведенческой биометрии на устройстве

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).

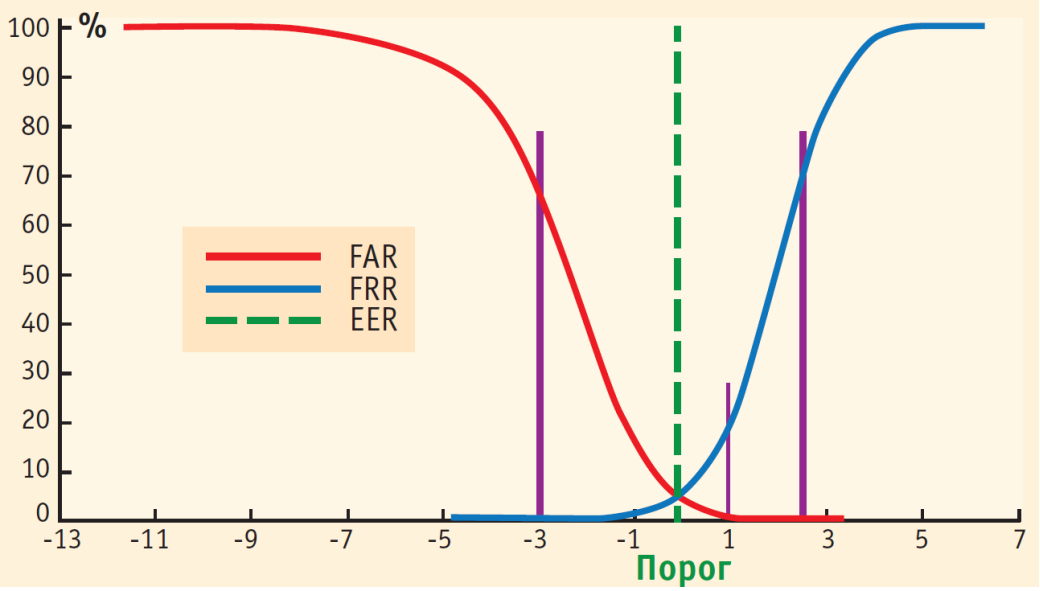

Рисунок 3 — Графики FAR и FRR [14]

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.

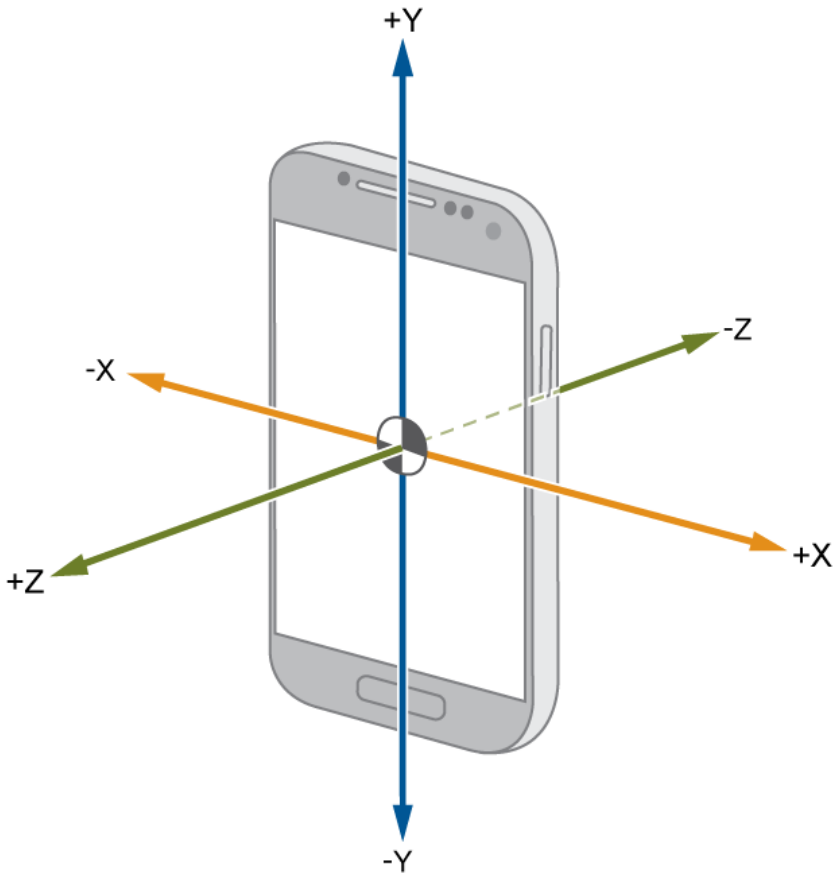

Рисунок 4 — Оси X, Y и Z относительно мобильного телефона [29]

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%.

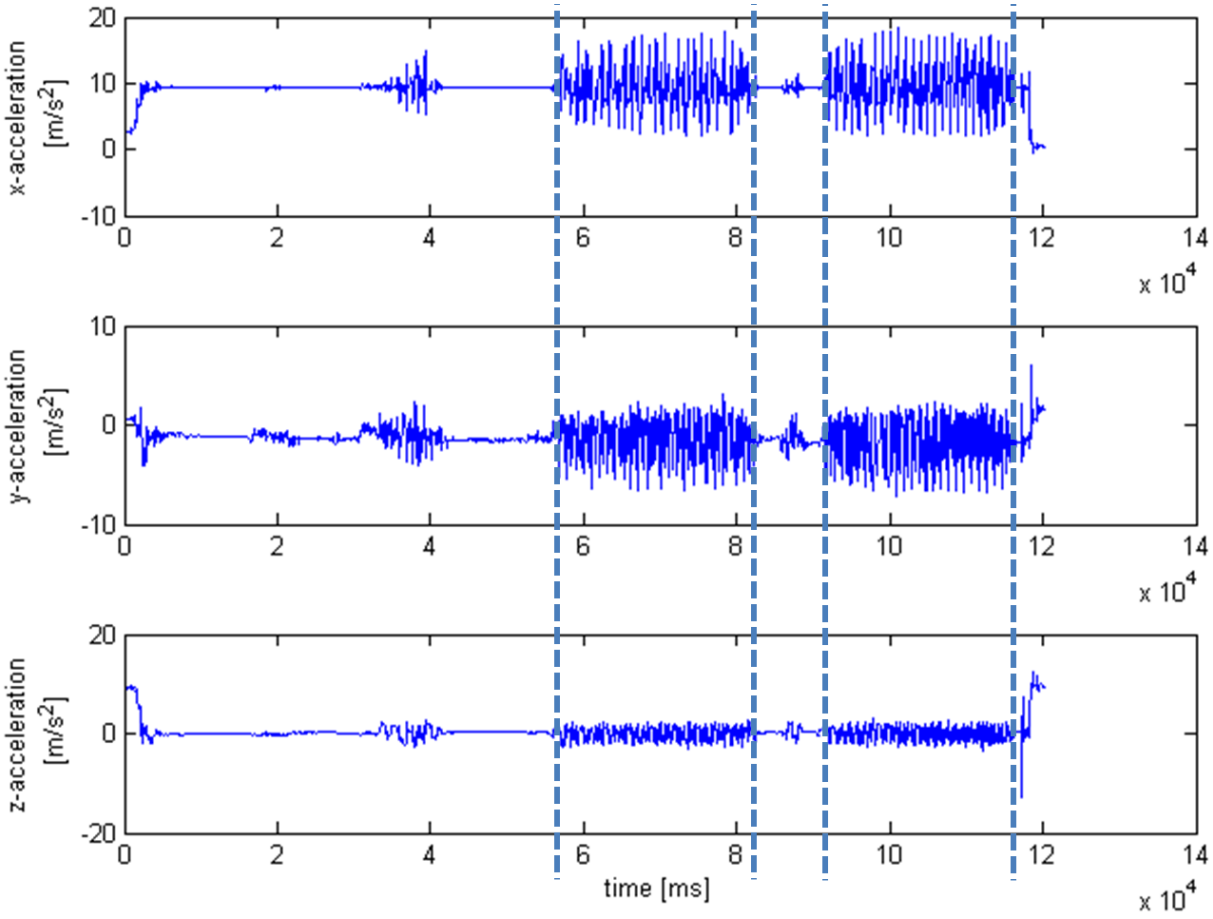

Рисунок 5 — Показания акселерометра для каждой оси [28]

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.

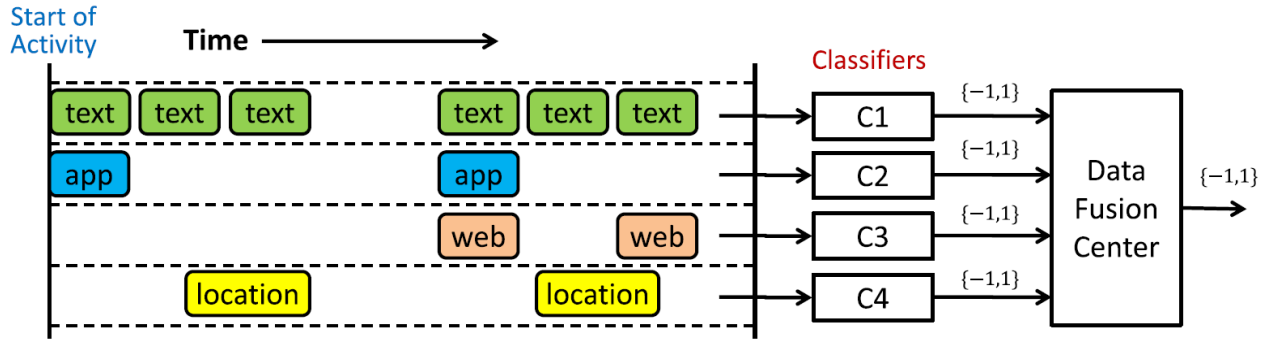

Рисунок 6 — Схема обработки событий классификаторами и принятия решения DFC [10]

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.

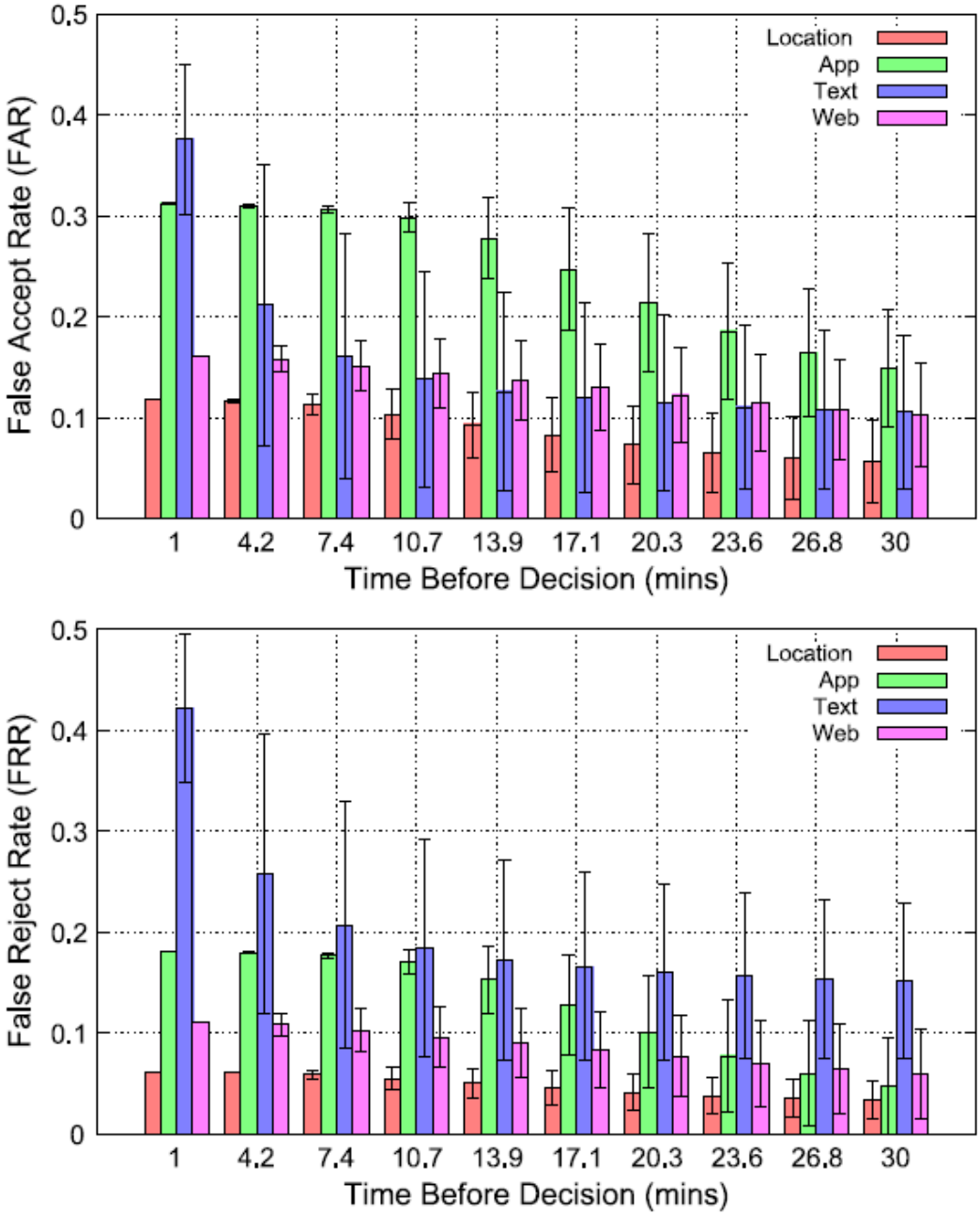

Рисунок 7 — Зависимость FAR и FRR для каждого модуля от временного интервала, отведенного на сбор данных [10]

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 система непрерывной аутентификации, состоящаю из нескольких модулей:

- LOCATION — модуль регистрации местоположения пользователя;
- APP — модуль регистрации статистики использования пользователем приложений на устройстве;
- WIFI — слежение за WiFi-сетями;
- BT — слежение 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.

APP фиксирует статистику взаимодействия пользователя с установленными на устройстве приложениями.

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

BT фиксирует окружение из устройств Bluetooth.

Каждый модуль формирует признаки, которые будут использоваться классификаторами, реализованными с помощью алгоритмов машинного обучения или нейронных сетей. При этом решение о допуске пользователя в систему может приниматься на основе вычислений путём 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.

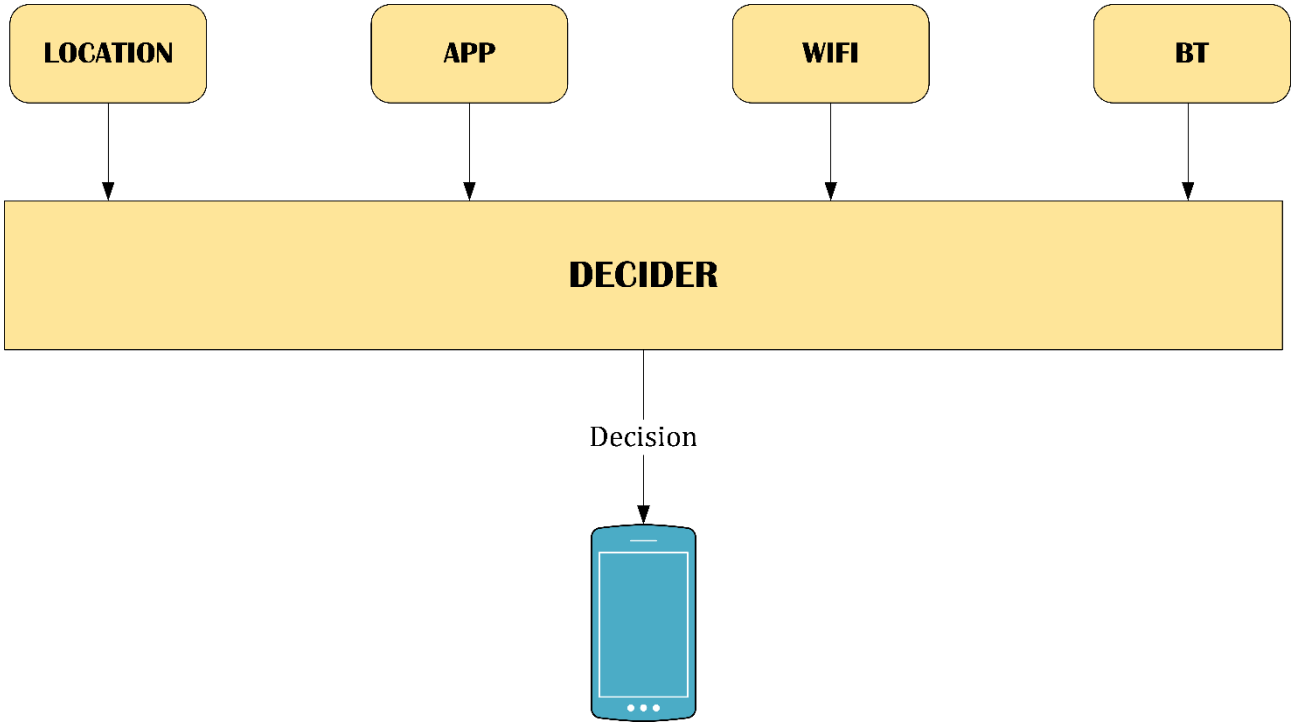

Рисунок 8 — Схема многомодульной системы аутентификации

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).

В большинстве проанализированных работ использовались методы бинарной классификации. Однако, они могут быть неприменимы для системы аутентификации на мобильном устройстве, так как система может не иметь данных о потенциальном злоумышленнике. На устройстве может храниться информация исключительно о поведенческих параметрах одного зарегистрированного пользователя. Поэтому предложено рассмотреть методы одноклассовой классификации (обнаружения аномалий) [34] и оценить их эффективность.

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 системы

Рассматриваемая система может быть использована двумя различными способами.

В первом случае система 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 важное 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 месторасположения факторов, то она потенциально сможет функционировать как в первом, так и во втором сценарии.

3.2 Архитектура мобильного приложения, реализующего метод аутентификации по поведенческому профилю пользователя

Для исследования возможностей аутентификации с помощью предложенной системы на мобильном устройстве под управлением операционной системы Android разработана архитектура приложения, которое будет собирать данные для аутентификации и аутентифицировать пользователя. Таким образом, приложение может иметь два режима функционирования:

- режим накопления данных;
- режим активной аутентификации.

На рисунке 9 представлена схема работы устройства, когда приложение находится в режиме сбора данных. Существуют несколько открытых наборов данных, однако они не всегда включают в себя именно ту информацию, которая необходима для функционирования предложенной системы аутентификации. А так как для функционирования системы в любом случае необходима возможность сбора данных, решено вынести это в отдельный режим. Данный режим будет задействован в двух сценариях. В первом случае — для продолжительного сбора данных без обработки (рис. 9). Такой сценарий будет использован в случае, если всё-таки будет необходимо создавать набор данных самостоятельно. Во втором случае режим сбора данных является одной из

составляющих режима активной аутентификации. Здесь данные собираются за короткие интервалы времени, чтобы проводить непрерывную аутентификацию.

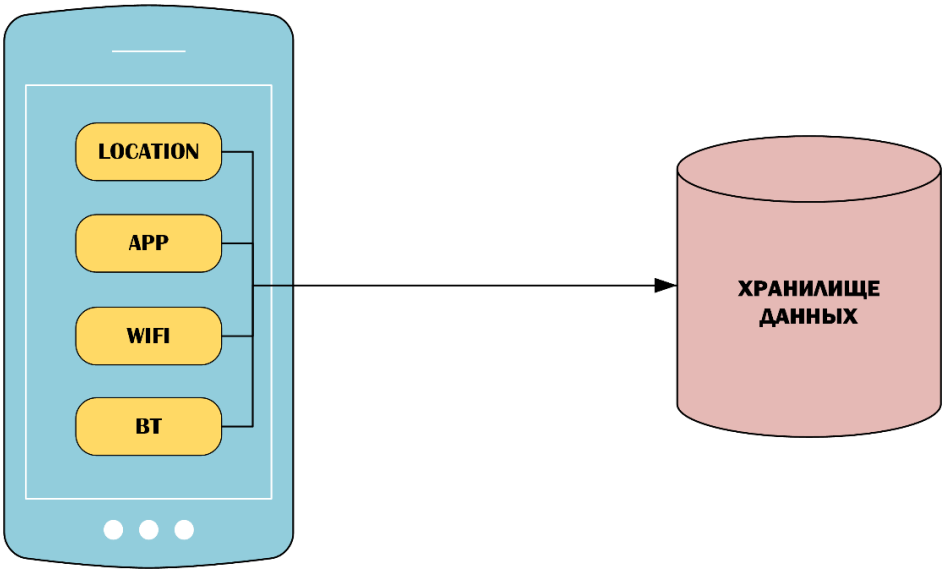

Рисунок 9 — Режим накопления данных

На рисунке 10 изображена схема функционирования устройства в режиме активной аутентификации. Модуль приложения, ответственный за аутентификацию (AUTH MODULE), управляет остальными модулями. Выполняется сбор данных в заданные короткие окна времени, после чего происходит обработка признаков, построение модели и классификация. Вердикт классификатора обрабатывается снова AUTH MODULE, после чего устройство блокируется или остаётся доступным пользователю.

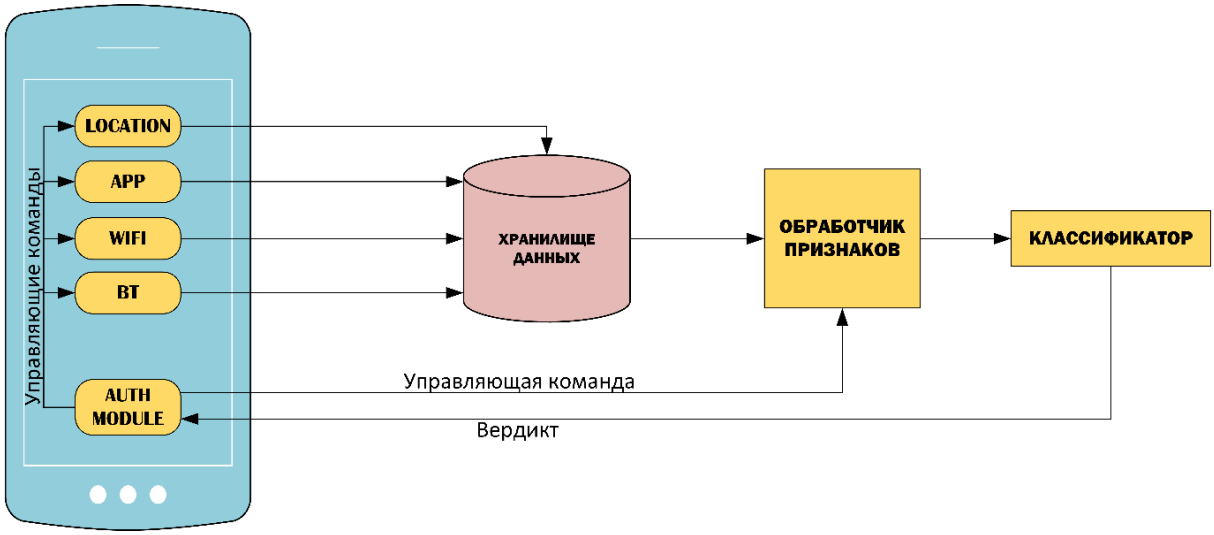

Рисунок 10 — Режим активной аутентификации

Приложение может иметь клиент-серверную архитектуру. На сервере могут храниться данные, а также может выполняться извлечение признаков и обучение моделей машинного обучения. В дальнейших работах предполагается исследовать влияния предложенного метода аутентификации на производительность и энергопотребление устройства с целью уточнить целесообразность создания клиент-серверной архитектуры.

С помощью приложения возможно будет оценить следующие параметры и характеристики предложенной системы аутентификации:

- FAR, FRR, EER и другие метрики;
- влияние на производительность мобильного устройства;
- влияние на время жизни устройства от одного заряда аккумулятора;
- оптимальный размер временного окна сбора данных от пользователя;
- удобство использования системы.

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 многомодульной системы аутентификации. При нехватке данных для одного из модулей становится невозможным проводить исследование системы. Поэтому было принято решение реализовать мобильное приложение, которое собирает данные пользователей для исследования.

На основе предложенной ранее архитектуры было разработано мобильное приложение под операционную систему Android, реализующее возможность сбора данных для дальнейшего исследования метода аутентификации. Приложение было разработано 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 11 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».

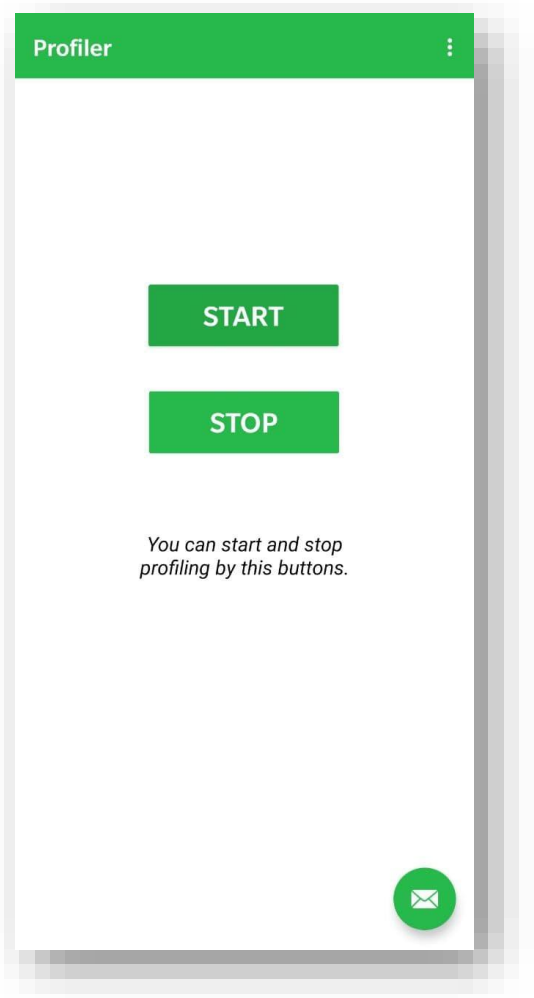

Рисунок 11 — Главный экран приложения

Служба управляет модулями WIFI, BT, LOCATION и APP. 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.

4.2 Собираемые данные

Каждый модуль приложения записывает получаемую информацию в отдельный файл CSV. Всего формируется 4 файла: app.data, 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.

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

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

Модуль APP собирает статистику использования приложений с помощью специального класса UsageStatsManager, предоставляющего информацию о статистике использования всех приложений, которую собирает операционная система.

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 3 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. Всего было собрано 8 ГБ данных поведения пользователей формата CSV. В процессе сбора данных была обнаружена проблема, заключающаяся в том, что операционная система могла остановить работу приложения в зависимости от настроенного профиля энергопотребления. Поэтому для 2 участников пришлось продлить участие в исследовании на сутки после изменения настроек системы.

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 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.

5.1 Формирование признаков

Полученные данные представляют собой упорядоченный во времени массив событий. Поэтому было решено 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.

Для вычисления среднего арифметического использовалась формула (1):

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

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 (2):

$$\begin{gathered} F_2 = [net_0, net_1, \ldots, net_n], \\ F_3 = [rssi_0, rssi_1, \ldots, rssi_n], \\ net_i = \begin{cases} 1, \textit{ если } i \in G_j \\ 0, \textit{ если } i \notin G_j \end{cases}, \\ rssi_i = \begin{cases} rssi_i^j, \textit{ если } i \in G_j \\ 0, \textit{ если } i \notin G_j \end{cases}, \end{gathered} \qquad (2)$$

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 (3):

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

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 (4):

$$disappeared\_nets\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m}\frac{|G_{j-k} \setminus (G_j \bigcap G_{j-k})|}{|G_j \bigcup G_{j-k}|}, \quad (4)$$

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 (5):

$$J\_dist\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m}\left(1-\frac{|G_j \bigcap G_{j-k}|}{|G_j \bigcup G_{j-k}|}\right), \quad (5)$$

5.1.1.4 Изменение вектора сетей между сканированиями

Евклидово расстояние между двумя векторами F_2 групп, усреднённое по окну. Использовалась формула (6):

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

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 (7):

$$rssi\_dist\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m}\sqrt{\sum_{i=0}^{n-1}(x_j^i - x_{j-k}^i)^2}, \quad (7)$$

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 (8):

$$conn\_rssi\_chg\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m} x_k - x_{j-k}, \quad (8)$$

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 (9):

$$count\_avg = \frac{1}{m}\sum_{\substack{k=1 \\ k<j}}^{m} count_k - count_{j-k}, \quad (9)$$

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 12, цветными рамками изображены группы элементарных векторов при размере окна в 20 секунд);

- путём разбиения с помощью скользящего окна (изображено в правой части рисунка 12, 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 12 — 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 по формуле (1);
- несмещённая выборочная дисперсия по формуле (10):

$$S^2 = \frac{1}{n-1}\sum_{i=0}^{n-1}(x_i - \overline{x})^2, \qquad (10)$$

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

- медиана;
- несмещённый выборочный коэффициент асимметрии [54] по формуле (11):

$$skew = \frac{\sqrt{n(n-1)}}{n-2} \frac{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^3}{\left(\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^2\right)^{\frac{3}{2}}}; \quad (11)$$

- несмещённый выборочный коэффициент эксцесса по формуле (12):

$$kurtosis = \frac{n^2 - 1}{(n-2)(n-3)}\left(\frac{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^4}{\left(\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^2\right)^2} - 3 + \frac{6}{n+1}\right). \quad (12)$$

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

- выборочное среднее по формуле (1);
- несмещённый выборочный коэффициент асимметрии (11);
- среднее абсолютное отклонение по формуле (13):

$$MAD = \frac{1}{n}\sum_{i=0}^{n-1}|x_i - \bar{x}|. \quad (13)$$

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) по формуле (1);
- 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 (3);
- число исчезнувших устройств (4);
- расстояние Жаккара (5);

- изменение вектора устройств между группами (6);
- изменение количества устройств в группе (9).

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

- выборочное среднее по формуле (1);
- несмещённая выборочная дисперсия (10);
- медиана;
- несмещённый выборочный коэффициент асимметрии (11);
- несмещённый выборочный коэффициент эксцесса (12).

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

- выборочное среднее (1);
- несмещённый выборочный коэффициент асимметрии (11);
- среднее абсолютное отклонение (13).

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 (1);
- несмещённая выборочная дисперсия (10);

- медиана;
- несмещённый выборочный коэффициент асимметрии (11);
- несмещённый выборочный коэффициент эксцесса (12);
- стандартное отклонение по формуле (14):

$$S = \sqrt{\frac{\sum_{i=0}^{n-1}(x_i - \overline{x})^2}{n-1}}, \tag{14}$$

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 (15):

$$speed = \frac{dist}{time}, \tag{15}$$

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 (16):

$$alt\_speed = \frac{altitude_{curr} - altitude_{prev}}{time}, \tag{16}$$

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

5.1.3.3 Ускорение

Ускорение рассчитывалось на основе рассчитанной скорости по формуле (17):

$$acceleration = \frac{speed_{curr} - speed_{prev}}{time}, \quad (17)$$

где $speed_{curr}$ и $speed_{prev}$ — скорость, рассчитанная ранее для текущего и для предшествующего события соответственно.

5.1.3.4 Ускорение для скорости изменения высоты

Ускорение рассчитывалось на основе рассчитанной скорости изменения высоты по формуле (18):

$$alt\_acceleration = \frac{alt\_speed_{curr} - alt\_speed_{prev}}{time}. \quad (18)$$

5.2 Подготовка выборок для обучения моделей

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 (19, 20):

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

$$x_{\max} = \max(x_0^j, ..., x_{n-1}^j) \tag{20}$$

После чего, каждое значение *j*-ого признака на *i*-ом объекте выборки вычисляется следующим образом (21):

$$x_j^i = \frac{x_j^i - x_{\min}}{x_{\max} - x_{\min}}, \tag{21}$$

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].

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 (22):

$$z = \frac{x - \overline{x}}{S}, \tag{22}$$

где $\bar{x}$ — выборочное среднее, рассчитывается по формуле (1), S — стандартное отклонение, рассчитывается по формуле (23):

$$S = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^2}\,. \tag{23}$$

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 рисунках 13 и 14 изображены диаграммы рассеяния, на которых по осям расположены значения некоторых признаков, полученных для модуля BT.

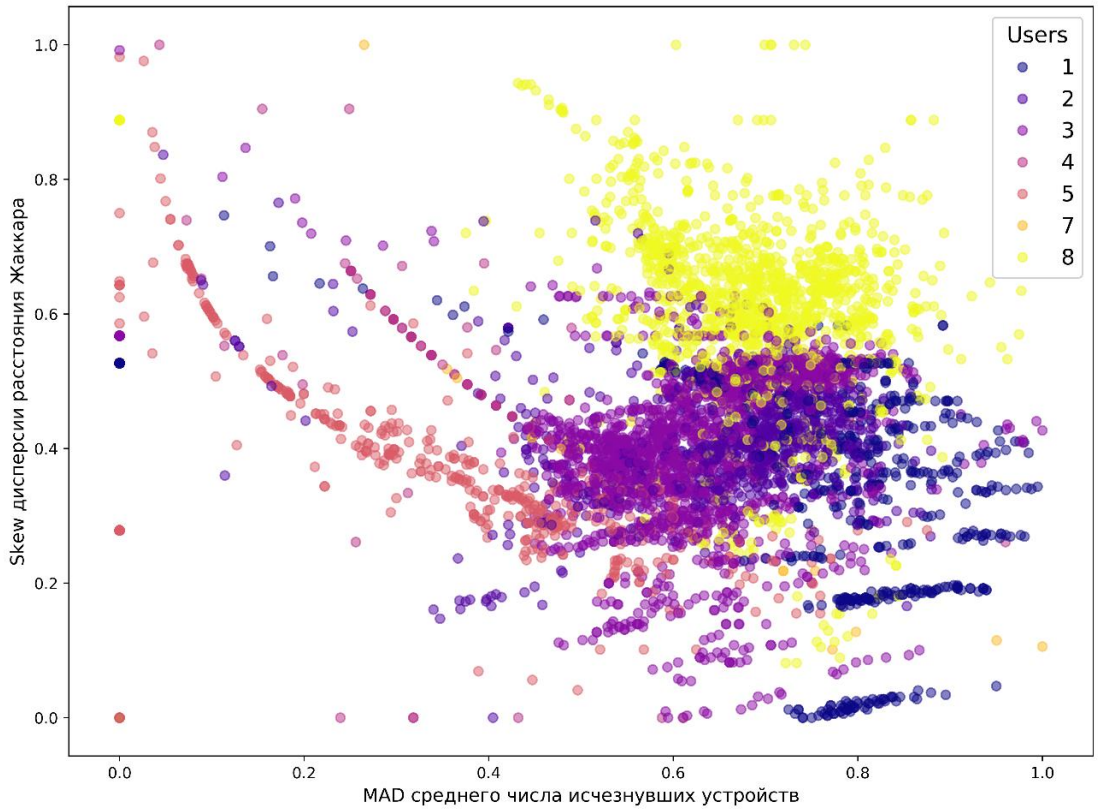

Рисунок 13 — Диаграмма рассеяния для каждого пользователя до масштабирования признаков

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. На рисунке 13 представлены данные до удаления выбросов, а на рисунке 14 — после. Видно, что значения признаков после удаления выбросов лежат в более узком интервале значений, а также уменьшилось количество удалённых отдельно лежащих точек.

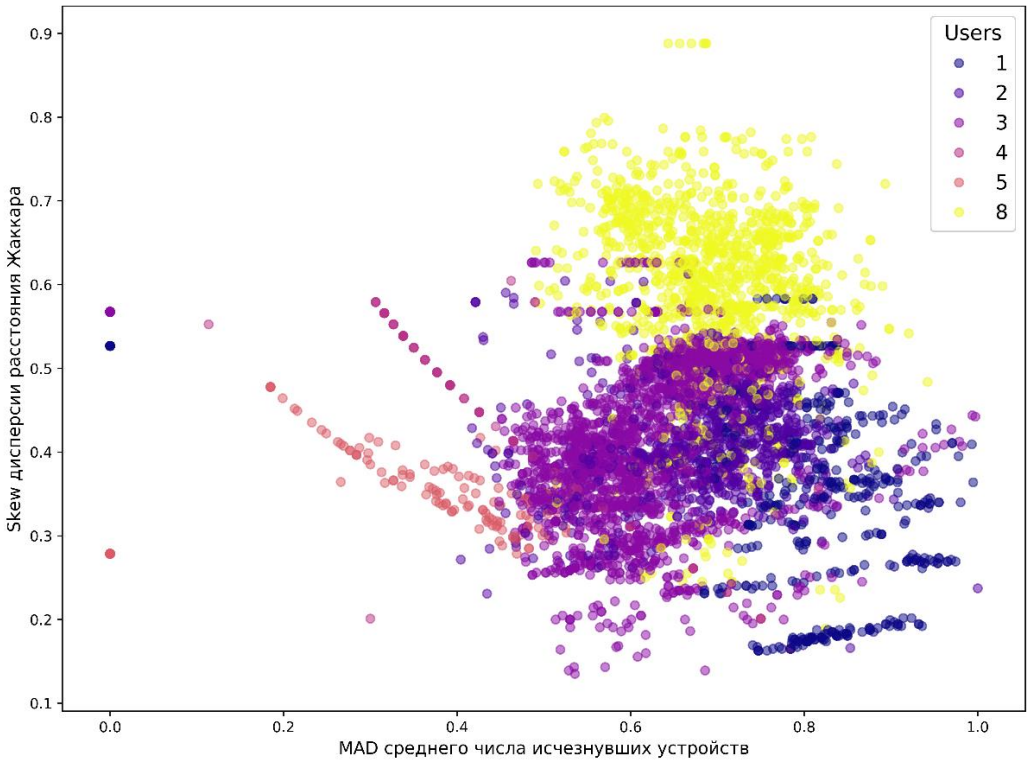

Рисунок 14 — Диаграмма рассеяния для каждого пользователя после масштабирования признаков

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. Он рассчитывается следующим образом (24):

$$r_{xy} = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n-1}(x_i - \bar{x})^2 \sum_{i=0}^{n-1}(y_i - \bar{y})^2}}, \quad (24)$$

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, вычисляются по формуле (1).

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 15 представлена матрица коэффициентов корреляции признаков для модуля LOCATION до удаления сильно коррелирующих, а на рисунке 16 — после удаления.

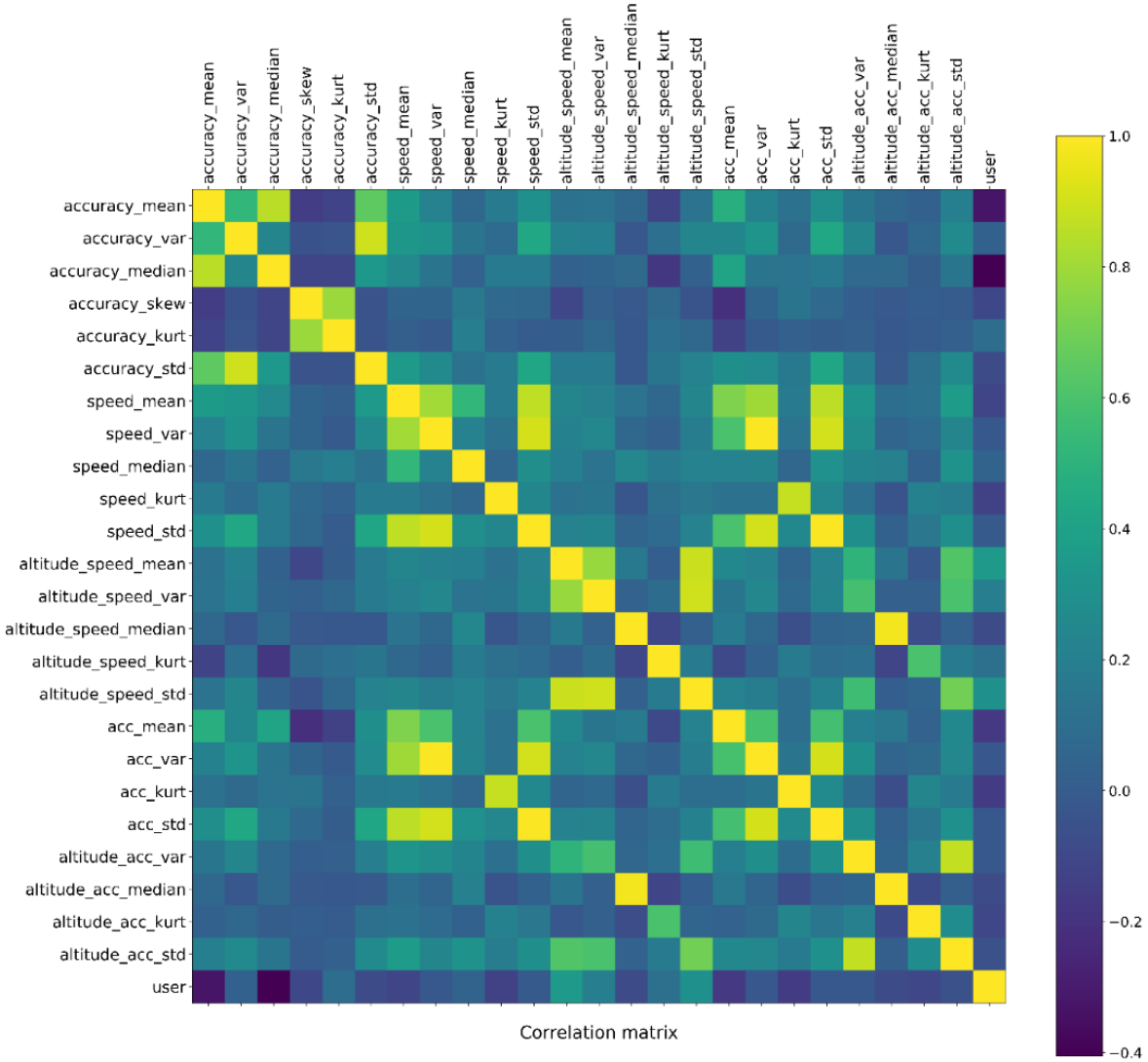

Рисунок 15 — Матрица корреляций признаков между собой до удаления сильно коррелирующих признаков для модуля LOCATION

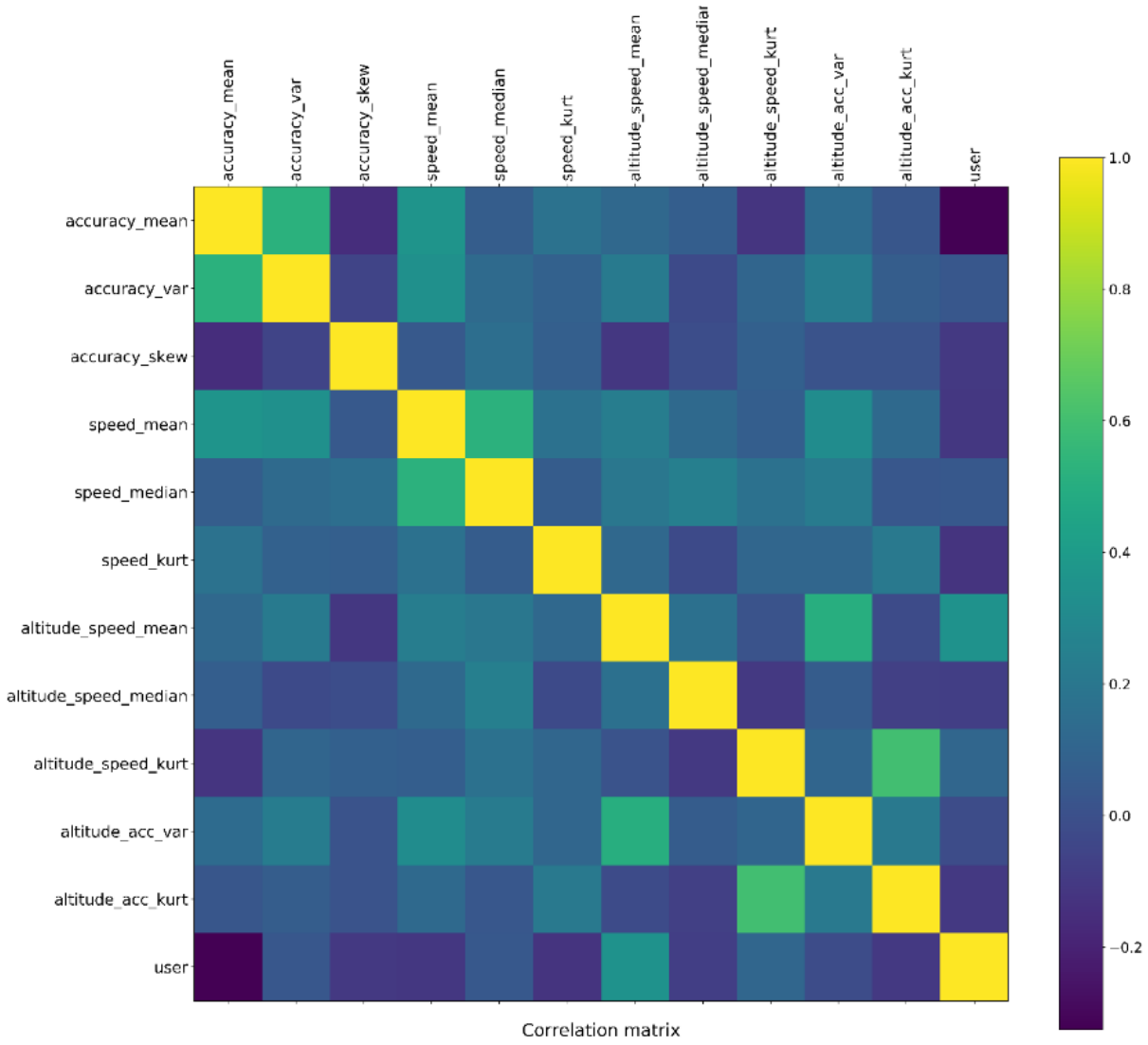

Рисунок 16 — Матрица корреляций признаков между собой после удаления сильно коррелирующих признаков для модуля LOCATION

Можно видеть, что признаки, связанные со значениями ускорения достаточно сильно коррелировали с признаками, характеризующими скорость. Это объясняется тем, что ускорение вычислялось по достаточно простой формуле (16) на основе значений скорости.

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 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] (использована реализация 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 (25):

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \quad (25)$$

Точность (precision) — доля истинно-положительных предсказаний алгоритма среди всех положительных предсказаний. Вычисляется по формуле (26):

$$precision = \frac{TP}{TP + FP}. \quad (26)$$

Полнота (recall, true positive rate, TPR) — отношение числа истинно-положительных вердиктов к количеству элементов в положительном классе. Рассчитывается по формуле (27):

$$recall = \frac{TP}{TP + FN}. \quad (27)$$

FRR — доля легальных пользователей, отвергнутых системой по ошибке (ошибка первого рода) [71]. Рассчитывается по формуле (28):

$$FRR = FNR = 1 - recall = \frac{FN}{TP + FN}. \quad (28)$$

FAR — доля нелегальных пользователей, пропущенных системой по ошибке (ошибка второго рода) [71]. Рассчитывается по формуле (29):

$$FAR = FPR = \frac{FP}{TN + FP}. \quad (29)$$

EER рассчитывается по формуле (30):

$$EER = \frac{FAR + FRR}{2}. \quad (30)$$

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.

F-мера — является обобщением precision и recall. Для расчёта используется формула (31):

$$F = \frac{2 \times precision \times recall}{precision + recall}. \quad (31)$$

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 дублирования случайных векторов (oversampling) [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 неизвестном ей ранее пользователе.

Для того, чтобы объединить полученные результаты кросс-валидации для каждого алгоритма было рассчитано среднее значение метрики accuracy следующим образом (32):

$$cv\_score = \frac{1}{n(n-1)} \sum_{\substack{1 \le i, j \le n \\ i \ne j}}^{n} score_i^j \,, \tag{32}$$

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, которая выполнялась на первом этапе. Ввиду того, что в тестовую выборку теперь были включены данные легального пользователя, можно рассчитать метрики AUC-ROC и F-меру (31).

Для вычисления метрик использовались функции из библиотеки sklearn.metrics.

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, 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	RandomForest	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 В для каждого модуля. В случае таблиц В.7, В.8, В.9, В.10, В.11, В.12 учитывались значения F-меры.

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

Модуль	CatBoostClassifier	RandomForest	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 (RandomForest) 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 17 визуализировано значение метрики accuracy 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 увеличивается с ростом размера окна в секундах.

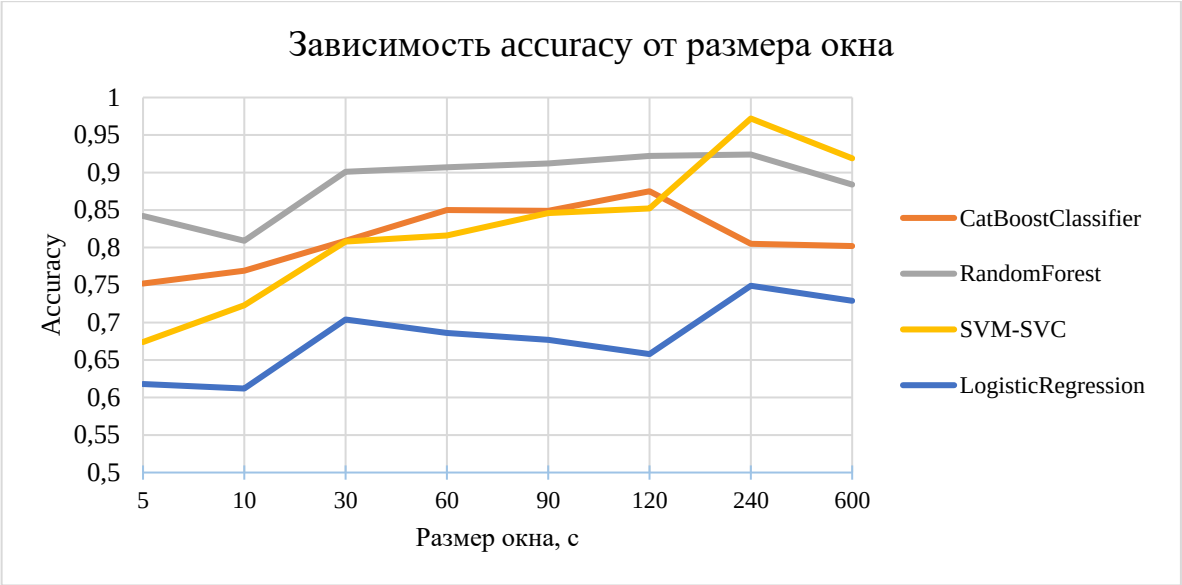

Рисунок 17 — График зависимости значения метрики accuracy на этапе кросс-валидации для модуля BT (стационарное окно)

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 18 представлена диаграмма, на которой можно заметить, что при росте размера окна значения F-меры как правило растут, хоть иногда и незначительно.

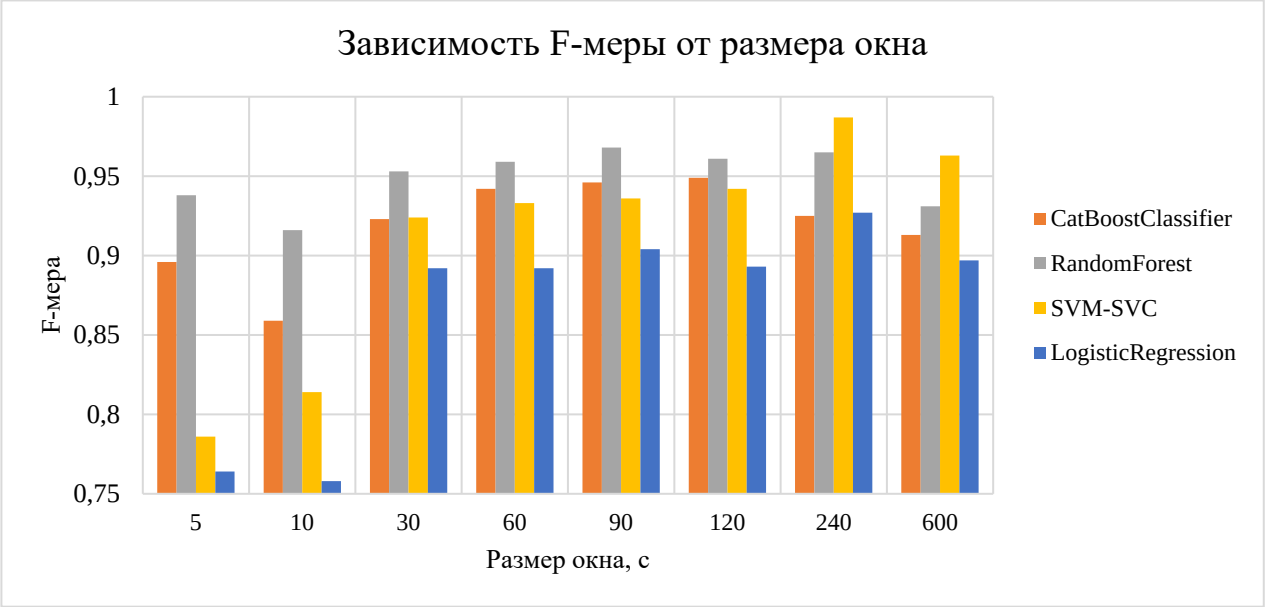

Рисунок 18 — Диаграмма зависимости значения метрики от размера стационарного окна для модуля BT на этапе финального тестирования

На рисунке 19 представлена диаграмма, визуализирующая различия в средних значениях F-меры для двух различных способов формирования признаков — с помощью стационарного и скользящего окна. Можно заметить,

что для модуля WIFI лучшие показатели были у скользящего окна, а у модуля BT — наоборот. Из этого следует, что для разных модулей следует детальнее проанализировать значения метрик при использовании разных типов окна, чтобы выбрать приемлемый вариант.

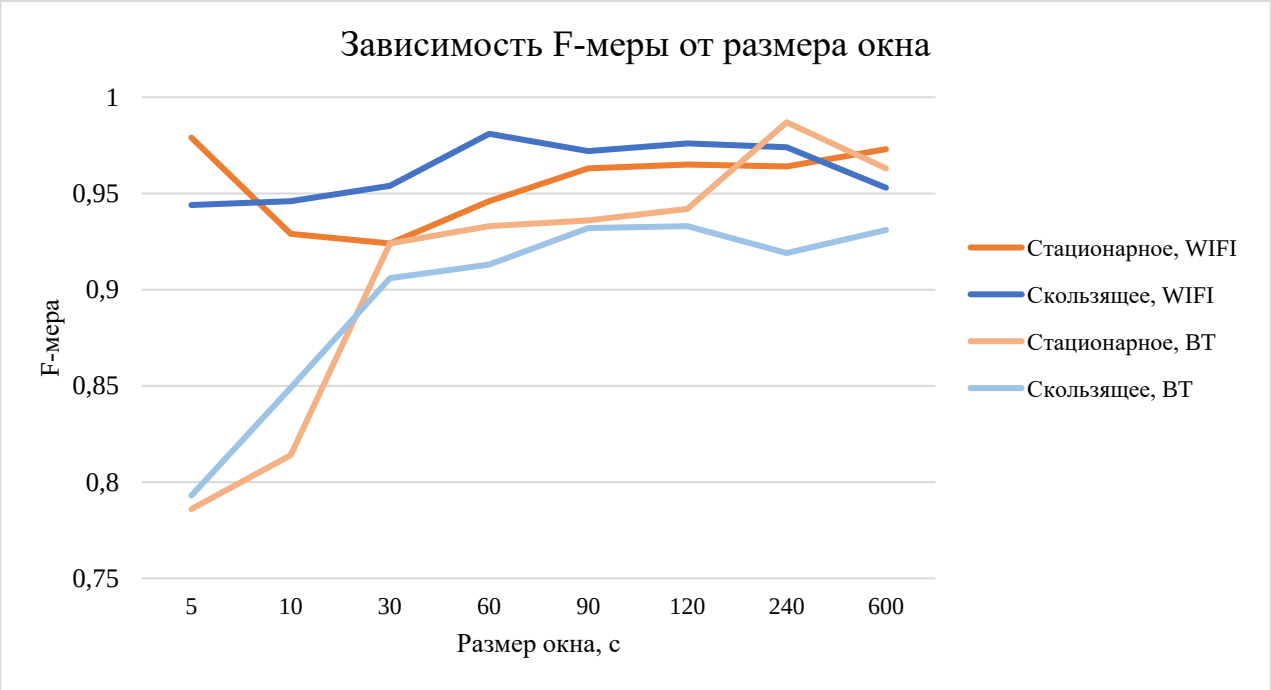

Рисунок 19 — График зависимости значения F-меры от размера окна для модуля BT и WIFI на этапе финального тестирования алгоритма RandomForest

На рисунке 20 представлена диаграмма, по которой можно сравнить значения F-меры 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 модели, кроме логистической регрессии. Лучшие значения метрик качества наблюдались при работе алгоритма RandomForest, для этого алгоритма получены значения F-меры в промежутке между от 0.916 до 0.993 на этапе финального тестирования. SVM-SVC также продемонстрировал высокие значения F-меры: от 0.786 до 0.987. Для дальнейшего исследования системы аутентификации решено исключить логистическую регрессию из перечня тестируемых алгоритмов.

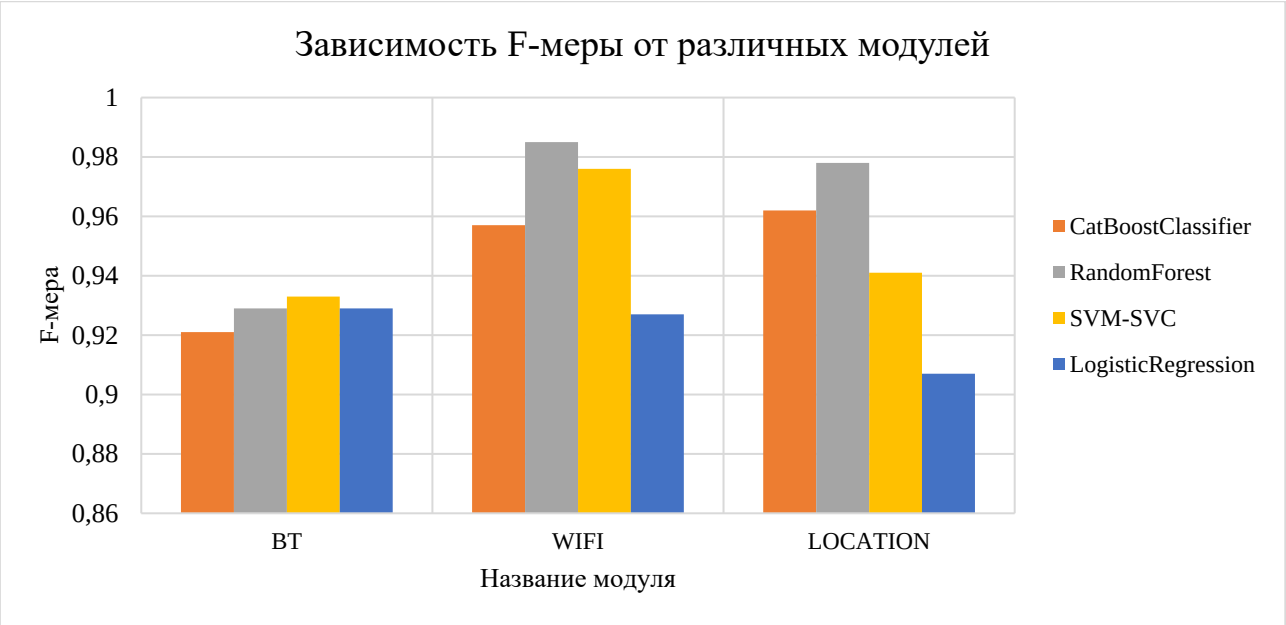

Рисунок 20 — Диаграмма зависимости значения F-меры от модуля при размере скользящего окна 120 секунд

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 выбора между скользящим и стационарным окном следует провести дополнительные исследования для каждого модуля в отдельности. Стоит, однако, отметить, что модели, обученные на признаках, сформированных с помощью скользящего окна, достигали пиковых значений AUC-ROC (0.999) раньше на 2-4 значения окна, чем при использовании стационарного окна.

Среди всех модулей лучшие значения метрик были получены при тестировании обученных моделей на данных от WIFI и LOCATION. Но отставание модуля BT по F-мере составило всего от 0.02 до 0.07.

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 тестирование моделей проводилось на выборке, состоящей из данных пользователя, которые отсутствовали в обучающей выборке.

ЗАКЛЮЧЕНИЕ

В данной работе предложен метод аутентификации по поведенческому профилю пользователя, основанный на данных, получаемых с помощью датчиков WiFi, Bluetooth, GPS, а также на основе статистики использования приложений.

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.

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

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

Лучшие значения F-меры были получены при тестировании алгоритмов случайного леса (0.916-0.987) и при использовании метода опорных векторов (0.786-0.987). Для модулей LOCATION и WIFI значения F-меры в среднем оказывались на 0.02-0.07 больше, чем для BT. При этом значения метрик на более длительных окнах (120 с, 240 с), использованных при формировании признаков, оказывались в среднем выше, чем на более коротких.

В дальнейшей работе будут сформированы признаки и проведено тестирование моделей для модуля APP. Также предполагается исследовать эффективность алгоритмов кластеризации и обнаружения аномалий,

приложенных к решению задачи аутентификации для многомодульной системы. Будет проведено объединение модулей в рамках единого метода. Построенная модель будет принимать решение об аутентификации, анализируя предсказания для каждого модуля. Заключительным шагом будущей работы является моделирование потока событий, приходящих на вход моделям, и оценка эффективности такого метода.

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.

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.app.AppOpsManager;
import android.bluetooth.BluetoothAdapter;
import android.content.Context;
import android.content.Intent;
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.Menu;
import android.view.MenuItem;
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 REQUEST_ENABLE_BT = 1;
    private final static int PERMISSIONS_REQUEST_ID = 1001;

    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(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!mIsPermissionsGranted) {
                    showLongToast("Grant permission please!");
                    requestPermissions();
                    return;
                }

                if (!isUsageStatsPermissionsGranted()) {
                    Intent intent = new
Intent(Settings.ACTION_USAGE_ACCESS_SETTINGS);
                    showLongToast("Grant action usage permission please!");
                    startActivityForResult(intent, 1);
                    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(new View.OnClickListener() {
            @Override
            public void onClick(View 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(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                sendZipProgressBar.setVisibility(View.VISIBLE);
                emailSendButton.setEnabled(false);
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        onSendButtonClick();
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                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();
    }

    @Override
    protected void onStop() {
        super.onStop();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    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() {
        return ProfilingService.isRunning;
    }

    private boolean isUsageStatsPermissionsGranted() {
        boolean granted = false;
        AppOpsManager appOps = (AppOpsManager)
getSystemService(APP_OPS_SERVICE);
        int mode = appOps.checkOpNoThrow(AppOpsManager.OPSTR_GET_USAGE_STATS,
android.os.Process.myUid(), getPackageName());

        if (mode == AppOpsManager.MODE_DEFAULT) {
            granted =
(checkCallingOrSelfPermission(android.Manifest.permission.PACKAGE_USAGE_STATS)
== PackageManager.PERMISSION_GRANTED);
        } else {
            granted = (mode == AppOpsManager.MODE_ALLOWED);
        }

        return granted;
    }

    void requestPermissions() {
        requestPermissions(new String[]{
                Manifest.permission.ACCESS_COARSE_LOCATION,
                Manifest.permission.ACCESS_FINE_LOCATION,
                Manifest.permission.ACCESS_WIFI_STATE,
                Manifest.permission.BLUETOOTH,
                Manifest.permission.BLUETOOTH_ADMIN,
                Manifest.permission.CHANGE_WIFI_STATE,
                Manifest.permission.FOREGROUND_SERVICE,
                Manifest.permission.PACKAGE_USAGE_STATS
        }, PERMISSIONS_REQUEST_ID);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
        if (requestCode == PERMISSIONS_REQUEST_ID && grantResults.length > 0 &&
grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            mIsPermissionsGranted = true;
        } else {
            mIsPermissionsGranted = false;
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
```

```
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    private void showLongToast(String text) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                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();
            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());

            MutexHolder.getMutex().lock();
            try {
                moveDataFilesToTempDirectory(ProfilingService.STAT_FILE_NAMES);
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                MutexHolder.getMutex().unlock();
            }

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

            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);
                }

                startActivity(chooser);
            }
        } catch (Exception ex) {
```

```
            Toast.makeText(getApplicationContext(), "Sending failed! Try
again!", Toast.LENGTH_LONG);
            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.BluetoothDevice;
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.Message;
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.ExistingWorkPolicy;
import androidx.work.OneTimeWorkRequest;
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.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_WIFI_SCAN_WORK_TAG =
"com.yerseg.profiler.WIFI_SCAN_WORK";
    public static final String PUSH_BT_SCAN_WORK_TAG =
"com.yerseg.profiler.BT_SCAN_WORK";
```

```
    public static final String PUSH_APP_STAT_SCAN_WORK_TAG =
"com.yerseg.profiler.APP_STAT_SCAN_WORK";

    public static final int WIFI_STATS_UPDATE_FREQ = 5000;
    public static final int BLUETOOTH_STATS_UPDATE_FREQ = 5000;
    public static final int APP_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 APP_STATS_FILE_NAME = "app.data";
    public static final String BLUETOOTH_STATS_FILE_NAME = "bt.data";
    public static final String LOCATION_STATS_FILE_NAME = "location.data";
    public static final String SCREEN_STATE_STATS_FILE_NAME = "screen.data";
    public static final String WIFI_STATS_FILE_NAME = "wifi.data";

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

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

    private Looper mServiceLooper;
    private ServiceHandler mServiceHandler;

    private BroadcastReceiver mScreenStatusBroadcastReceiver;
    private BroadcastReceiver mWifiScanReceiver;
    private BroadcastReceiver mBluetoothBroadcastReceiver;

    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;
        }

        HandlerThread thread = new HandlerThread("LocationThread",
Process.THREAD_PRIORITY_FOREGROUND);
        thread.start();

        mServiceLooper = thread.getLooper();
        mServiceHandler = new ServiceHandler(mServiceLooper);

        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()));

        startScreenStateTracking();
        startLocationTracking();
        startWifiTracking();
        startBluetoothTracking();
        startApplicationsStatisticTracking();

        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        synchronized (this) {
            isStopping = true;
        }

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

        stopScreenStateTracking();
        stopLocationTracking();
        stopWifiTracking();
        stopBluetoothTracking();
        stopApplicationsStatisticTracking();

        NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.cancel(1);

        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.setOngoing(true)
                .setContentTitle("Profiler App is running")
                .setPriority(NotificationManager.IMPORTANCE_MAX)
                .setCategory(Notification.CATEGORY_SERVICE)
                .build();

        startForeground(1, notification);
    }

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

        FusedLocationProviderClient fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
        if (fusedLocationProviderClient != null) {
            int permission = ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION);
            if (permission == PackageManager.PERMISSION_GRANTED) {
                mLocationCallback = new LocationCallback() {
                    @Override
                    public void onLocationResult(LocationResult result) {
                        Log.d("Profiler [LocationStat]",
String.format(Locale.getDefault(), "\t%d\tonLocationResult()",
Process.myTid()));
                        Location location = result.getLastLocation();
                        String locationStats =
String.format(Locale.getDefault(), "%s,%f,%f,%f,%f,%s\n",
                                Utils.GetTimeStamp(System.currentTimeMillis()),
                                location.getAccuracy(),
                                location.getAltitude(),
                                location.getLatitude(),
                                location.getLongitude(),
                                location.getProvider()
                        );

Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
LOCATION_STATS_FILE_NAME, locationStats);
                    }
                };

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

    private void startWifiTracking() {
        final WifiManager wifiManager = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);

        mWifiScanReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context c, Intent intent) {
                if (intent.getBooleanExtra(WifiManager.EXTRA_RESULTS_UPDATED,
false)) {
                    new Thread(new Runnable() {
                        @Override
                        public void run() {
```

```
                            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,%s,%s,%d,%d,%d,%d,%d,%s,%d,%s,%b,%b\n",
                                        timestamp,
                                        statResponseId,
                                        result.BSSID,
                                        result.SSID,
                                        result.capabilities,
                                        result.centerFreq0,
                                        result.centerFreq1,
                                        result.channelWidth,
                                        result.frequency,
                                        result.level,
                                        result.operatorFriendlyName.length(),
                                        result.timestamp,
                                        result.venueName,
                                        result.is80211mcResponder(),
                                        result.isPasspointNetwork());


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
WIFI_STATS_FILE_NAME, wifiStats);
                            }
                        }
                    }).start();
                }
            }
        };

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

        OneTimeWorkRequest refreshWork = new
OneTimeWorkRequest.Builder(WifiProfilingWorker.class).build();

WorkManager.getInstance(getApplicationContext()).enqueueUniqueWork(PUSH_WIFI_SCA
N_WORK_TAG, ExistingWorkPolicy.KEEP, refreshWork);
    }

    private void startBluetoothTracking() {
        mBluetoothBroadcastReceiver = new BroadcastReceiver() {
            public void onReceive(Context context, Intent intent) {
                String action = intent.getAction();
                if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                    BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                    new Thread(new Runnable() {
                        @Override
                        public void run() {
```

```
                                String bluetoothStats =
String.format(Locale.getDefault(), "%s,%s,%s,%d,%d,%d,%d\n",

Utils.GetTimeStamp(System.currentTimeMillis()),
                                        device.getName(),
                                        device.getAddress(),

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


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
BLUETOOTH_STATS_FILE_NAME, bluetoothStats);
                            }
                        }).start();
                    }
                }
            };

            IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
            registerReceiver(mBluetoothBroadcastReceiver, filter);

            OneTimeWorkRequest refreshWork = new
OneTimeWorkRequest.Builder(BluetoothProfilerWorker.class).build();

WorkManager.getInstance(getApplicationContext()).enqueueUniqueWork(PUSH_BT_SCAN_
WORK_TAG, ExistingWorkPolicy.KEEP, refreshWork);
        }

        private void startApplicationsStatisticTracking() {
            OneTimeWorkRequest refreshWork = new
OneTimeWorkRequest.Builder(ApplicationsProfilerWorker.class).build();

WorkManager.getInstance(getApplicationContext()).enqueueUniqueWork(PUSH_APP_STAT
_SCAN_WORK_TAG, ExistingWorkPolicy.KEEP, refreshWork);
        }

        private void startScreenStateTracking() {
            mScreenStatusBroadcastReceiver = new BroadcastReceiver() {
                @Override
                public void onReceive(Context context, Intent intent) {
                    String intentActionName = "";

                    if (intent.getAction().equals(Intent.ACTION_USER_PRESENT)) {
                        intentActionName = "ACTION_USER_PRESENT";
                    } else if (intent.getAction().equals(Intent.ACTION_SHUTDOWN)) {
                        intentActionName = "ACTION_SHUTDOWN";
                    } else if
(intent.getAction().equals(Intent.ACTION_DREAMING_STARTED)) {
                        intentActionName = "ACTION_DREAMING_STARTED";
                    } else if
(intent.getAction().equals(Intent.ACTION_DREAMING_STOPPED)) {
                        intentActionName = "ACTION_DREAMING_STOPPED";
                    } else if (intent.getAction().equals(Intent.ACTION_REBOOT)) {
                        intentActionName = "ACTION_REBOOT";
                    } else if (intent.getAction().equals(Intent.ACTION_SCREEN_OFF))
{
                        intentActionName = "ACTION_SCREEN_OFF";
                    } else if (intent.getAction().equals(Intent.ACTION_SCREEN_ON)) {
                        intentActionName = "ACTION_SCREEN_ON";
```

```
                } else if
(intent.getAction().equals(Intent.ACTION_USER_UNLOCKED)) {
                    intentActionName = "ACTION_USER_UNLOCKED";
                }

                final String finalIntentActionName = intentActionName;
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        String screenStats = String.format(Locale.getDefault(),
"%s,%s\n", Utils.GetTimeStamp(System.currentTimeMillis()),
finalIntentActionName);

Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
SCREEN_STATE_STATS_FILE_NAME, screenStats);
                    }
                }).run();
            }
        };

        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(Intent.ACTION_USER_PRESENT);
        intentFilter.addAction(Intent.ACTION_SHUTDOWN);
        intentFilter.addAction(Intent.ACTION_DREAMING_STARTED);
        intentFilter.addAction(Intent.ACTION_DREAMING_STOPPED);
        intentFilter.addAction(Intent.ACTION_REBOOT);
        intentFilter.addAction(Intent.ACTION_SCREEN_OFF);
        intentFilter.addAction(Intent.ACTION_SCREEN_ON);
        intentFilter.addAction(Intent.ACTION_USER_UNLOCKED);

        registerReceiver(mScreenStatusBroadcastReceiver, intentFilter);
    }

    void stopScreenStateTracking() {
        if (mScreenStatusBroadcastReceiver != null)
            unregisterReceiver(mScreenStatusBroadcastReceiver);
    }

    void stopLocationTracking() {
        FusedLocationProviderClient fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
        if (fusedLocationProviderClient != null) {

fusedLocationProviderClient.removeLocationUpdates(mLocationCallback);
        }

        mServiceHandler.removeCallbacksAndMessages(null);
        mServiceLooper.quitSafely();
    }

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


WorkManager.getInstance(getApplicationContext()).cancelUniqueWork(PUSH_APP_STAT_
SCAN_WORK_TAG);
    }

    void stopBluetoothTracking() {
        if (mBluetoothBroadcastReceiver != null)
            unregisterReceiver(mBluetoothBroadcastReceiver);
```

```
WorkManager.getInstance(getApplicationContext()).cancelUniqueWork(PUSH_BT_SCAN_W
ORK_TAG);
    }

    void stopApplicationsStatisticTracking() {

WorkManager.getInstance(getApplicationContext()).cancelUniqueWork(PUSH_APP_STAT_
SCAN_WORK_TAG);
    }

    private final class ServiceHandler extends Handler {
        public ServiceHandler(Looper looper) {
            super(looper);
        }

        @Override
        public void handleMessage(@NonNull Message msg) {
            super.handleMessage(msg);
        }
    }
}
```

ApplicationProfilerWorker.java

```
package com.yerseg.profiler;

import android.app.usage.ConfigurationStats;
import android.app.usage.EventStats;
import android.app.usage.UsageStats;
import android.app.usage.UsageStatsManager;
import android.content.Context;
import android.content.res.Configuration;
import android.os.SystemClock;

import androidx.annotation.NonNull;
import androidx.work.ExistingWorkPolicy;
import androidx.work.OneTimeWorkRequest;
import androidx.work.WorkManager;
import androidx.work.Worker;
import androidx.work.WorkerParameters;

import java.util.List;
import java.util.Locale;
import java.util.UUID;
import java.util.concurrent.CancellationException;

public class ApplicationsProfilerWorker extends Worker {
    Context mContext;

    public ApplicationsProfilerWorker(@NonNull Context context, @NonNull
WorkerParameters workerParams) {
        super(context, workerParams);
        mContext = context;
    }

    @NonNull
    @Override
    public Result doWork() {
        try {
```

```
            Thread.sleep(ProfilingService.APP_STATS_UPDATE_FREQ);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        doActualWork();
        return Result.success();
    }

    private void doActualWork() {


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
ProfilingService.APP_STATS_FILE_NAME, getStatisticsForWritingToFile());

        if (!ProfilingService.isStopping) {
            try {
                OneTimeWorkRequest refreshWork = new
OneTimeWorkRequest.Builder(ApplicationsProfilerWorker.class).build();

WorkManager.getInstance(mContext).enqueueUniqueWork(ProfilingService.PUSH_APP_ST
AT_SCAN_WORK_TAG, ExistingWorkPolicy.REPLACE, refreshWork);
            } catch (CancellationException ex) {
                ex.printStackTrace();
            }
        }
    }

    private String getStatisticsForWritingToFile() {
        long beginTime = java.lang.System.currentTimeMillis() -
SystemClock.elapsedRealtime();
        long endTime = java.lang.System.currentTimeMillis();

        UsageStatsManager usageStatsManager = (UsageStatsManager)
mContext.getSystemService(Context.USAGE_STATS_SERVICE);

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

        StringBuilder statistic = new StringBuilder();

        List<UsageStats> usageStatsList =
usageStatsManager.queryUsageStats(UsageStatsManager.INTERVAL_DAILY, beginTime,
endTime);

        for (UsageStats usageStats : usageStatsList) {
            statistic.append(String.format(Locale.getDefault(),
"UsageStats,%s,%s,%s,%d,%d,%d,%d\n",
                    timestamp,
                    statResponseId,
                    usageStats.getPackageName(),
                    usageStats.getFirstTimeStamp(),
                    usageStats.getLastTimeStamp(),
                    usageStats.getLastTimeUsed(),
                    usageStats.getTotalTimeInForeground()));
        }

        List<ConfigurationStats> configurationStatsList =
usageStatsManager.queryConfigurations(UsageStatsManager.INTERVAL_DAILY,
beginTime, endTime);

        for (ConfigurationStats configurationStats : configurationStatsList) {
            Configuration configuration = configurationStats.getConfiguration();
```

```
            statistic.append(String.format(Locale.getDefault(),
"ConfigStats,%s,%s,%d,%d,%d,%d,%d,%d,%d,%f,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%
d,%d,%d,%s,%b,%b,%b\n",
                    timestamp,
                    statResponseId,
                    configurationStats.getActivationCount(),
                    configurationStats.getFirstTimeStamp(),
                    configurationStats.getLastTimeActive(),
                    configurationStats.getLastTimeStamp(),
                    configurationStats.getTotalTimeActive(),
                    configuration.colorMode,
                    configuration.densityDpi,
                    configuration.fontScale,
                    configuration.hardKeyboardHidden,
                    configuration.keyboard,
                    configuration.keyboardHidden,
                    configuration.mcc,
                    configuration.mnc,
                    configuration.navigation,
                    configuration.navigationHidden,
                    configuration.orientation,
                    configuration.screenHeightDp,
                    configuration.screenLayout,
                    configuration.screenWidthDp,
                    configuration.smallestScreenWidthDp,
                    configuration.touchscreen,
                    configuration.uiMode,
                    configuration.getLayoutDirection(),
                    configuration.getLocales().toLanguageTags(),
                    configuration.isScreenHdr(),
                    configuration.isScreenRound(),
                    configuration.isScreenWideColorGamut()));
        }

        List<EventStats> eventStatsList =
usageStatsManager.queryEventStats(UsageStatsManager.INTERVAL_DAILY, beginTime,
endTime);

        for (EventStats eventStat : eventStatsList) {
            statistic.append(String.format(Locale.getDefault(),
"EventStats,%s,%s,%d,%d,%d,%d,%d,%d\n",
                    timestamp,
                    statResponseId,
                    eventStat.getCount(),
                    eventStat.getEventType(),
                    eventStat.getFirstTimeStamp(),
                    eventStat.getLastTimeStamp(),
                    eventStat.getLastEventTime(),
                    eventStat.getTotalTime()));
        }

        return statistic.toString();
    }
}
```

BluetoothProfilerWorker.java

```
package com.yerseg.profiler;

import android.bluetooth.BluetoothAdapter;
```

```
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothManager;
import android.bluetooth.BluetoothProfile;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanCallback;
import android.bluetooth.le.ScanResult;
import android.content.Context;

import androidx.annotation.NonNull;
import androidx.work.ExistingWorkPolicy;
import androidx.work.OneTimeWorkRequest;
import androidx.work.WorkManager;
import androidx.work.Worker;
import androidx.work.WorkerParameters;

import java.util.List;
import java.util.Locale;
import java.util.Set;
import java.util.UUID;
import java.util.concurrent.CancellationException;

public class BluetoothProfilerWorker extends Worker {

    Context mContext;

    public BluetoothProfilerWorker(@NonNull Context context, @NonNull
WorkerParameters workerParams) {
        super(context, workerParams);
        mContext = context;
    }

    @NonNull
    @Override
    public Result doWork() {
        try {
            Thread.sleep(ProfilingService.BLUETOOTH_STATS_UPDATE_FREQ);

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        doActualWork();
        return Result.success();
    }

    private void doActualWork() {
        String statResponseId = UUID.randomUUID().toString();
        final BluetoothAdapter bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
        if (bluetoothAdapter != null) {
            if (!bluetoothAdapter.isDiscovering())
                BluetoothAdapter.getDefaultAdapter().startDiscovery();

            String timestamp = Utils.GetTimeStamp(System.currentTimeMillis());
            Set<BluetoothDevice> bondedDevices =
bluetoothAdapter.getBondedDevices();

            for (BluetoothDevice device : bondedDevices) {
                String bluetoothStats = String.format(Locale.getDefault(),
"%s,%s,BONDED,%s,%s,%d,%d,%d,%d\n",
                        timestamp,
                        statResponseId,
                        device.getName(),
```

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


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
ProfilingService.BLUETOOTH_STATS_FILE_NAME, bluetoothStats);
            }
        }

        final BluetoothManager bluetoothManager = (BluetoothManager)
getApplicationContext().getSystemService(Context.BLUETOOTH_SERVICE);
        if (bluetoothManager != null) {
            String timestamp = Utils.GetTimeStamp(System.currentTimeMillis());
            List<BluetoothDevice> gattDevices =
bluetoothManager.getConnectedDevices(BluetoothProfile.GATT);

            for (BluetoothDevice device : gattDevices) {
                String bluetoothStats = String.format(Locale.getDefault(),
"%s,%s,GATT,%s,%s,%d,%d,%d,%d\n",
                        timestamp,
                        statResponseId,
                        device.getName(),
                        device.getAddress(),
                        device.getBluetoothClass().getMajorDeviceClass(),
                        device.getBluetoothClass().getDeviceClass(),
                        device.getBondState(),
                        device.getType());


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
ProfilingService.BLUETOOTH_STATS_FILE_NAME, bluetoothStats);
            }

            timestamp = Utils.GetTimeStamp(System.currentTimeMillis());
            List<BluetoothDevice> gattServerDevices =
bluetoothManager.getConnectedDevices(BluetoothProfile.GATT_SERVER);

            for (BluetoothDevice device : gattServerDevices) {
                String bluetoothStats = String.format(Locale.getDefault(),
"%s,%s,GATT_SERVER,%s,%s,%d,%d,%d,%d\n",
                        timestamp,
                        statResponseId,
                        device.getName(),
                        device.getAddress(),
                        device.getBluetoothClass().getMajorDeviceClass(),
                        device.getBluetoothClass().getDeviceClass(),
                        device.getBondState(),
                        device.getType());


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
ProfilingService.BLUETOOTH_STATS_FILE_NAME, bluetoothStats);
            }
        }

        BluetoothLeScanner btScanner =
BluetoothAdapter.getDefaultAdapter().getBluetoothLeScanner();
        btScanner.startScan(new ScanCallback() {
            @Override
```

```
            public void onScanResult(int callbackType, ScanResult result) {
                super.onScanResult(callbackType, result);

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

        if (!ProfilingService.isStopping) {
            try {
                OneTimeWorkRequest refreshWork = new
OneTimeWorkRequest.Builder(BluetoothProfilerWorker.class).build();

WorkManager.getInstance(mContext).enqueueUniqueWork(ProfilingService.PUSH_BT_SCA
N_WORK_TAG, ExistingWorkPolicy.REPLACE, refreshWork);
            } catch (CancellationException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

MutexHolder.java

```
package com.yerseg.profiler;

import java.util.concurrent.locks.ReentrantLock;

public class MutexHolder {
    private static volatile ReentrantLock mutex;

    public static ReentrantLock getMutex() {
        ReentrantLock localInstance = mutex;
        if (localInstance == null) {
            synchronized (ReentrantLock.class) {
                localInstance = mutex;
                if (localInstance == null) {
                    mutex = localInstance = new ReentrantLock();
                }
            }
        }
        return localInstance;
    }
}
```

Utils.java

```
package com.yerseg.profiler;
```

```
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.io.IOException;
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 {

    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 void moveFile(File src, File dst) throws IOException {
        FileChannel inChannel = new FileInputStream(src).getChannel();
        FileChannel outChannel = new FileOutputStream(dst).getChannel();

        try {
            inChannel.transferTo(0, inChannel.size(), outChannel);

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

```
            ex.printStackTrace();
        } finally {
            if (inChannel != null)
                inChannel.close();
            if (outChannel != null)
                outChannel.close();
        }
    }

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

            try {
                MutexHolder.getMutex().lock();

                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();
            } finally {
                MutexHolder.getMutex().unlock();
            }
        }
    }

    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 {
                        origin.close();
                    }
                }

                out.finish();
                out.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

WifiProfilingWorker.java

```
package com.yerseg.profiler;

import android.content.Context;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;

import androidx.annotation.NonNull;
import androidx.work.ExistingWorkPolicy;
import androidx.work.OneTimeWorkRequest;
import androidx.work.WorkManager;
import androidx.work.Worker;
import androidx.work.WorkerParameters;

import java.util.List;
import java.util.Locale;
import java.util.UUID;
import java.util.concurrent.CancellationException;

public class WifiProfilingWorker extends Worker {
```

```
    Context mContext;

    public WifiProfilingWorker(@NonNull Context context, @NonNull
WorkerParameters workerParams) {
        super(context, workerParams);
        mContext = context;
    }

    @NonNull
    @Override
    public Result doWork() {
        try {
            Thread.sleep(ProfilingService.WIFI_STATS_UPDATE_FREQ);

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        doActualWork();
        return Result.success();
    }

    private void doActualWork() {
        WifiManager wifiManager = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        wifiManager.startScan();

        String statResponseId = UUID.randomUUID().toString();
        String timestamp = Utils.GetTimeStamp(System.currentTimeMillis());
        WifiInfo currentInfo = wifiManager.getConnectionInfo();

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


Utils.FileWriter.writeFile(Utils.getProfilingFilesDir(getApplicationContext()),
ProfilingService.WIFI_STATS_FILE_NAME, connectionInfo);

        if (!ProfilingService.isStopping) {
            try {
                OneTimeWorkRequest refreshWork = new
OneTimeWorkRequest.Builder(WifiProfilingWorker.class).build();

WorkManager.getInstance(mContext).enqueueUniqueWork(ProfilingService.PUSH_WIFI_S
CAN_WORK_TAG, ExistingWorkPolicy.REPLACE, refreshWork);
            } catch (CancellationException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

Приложение Б

Исходный код сценариев, используемых для автоматизированной обработки данных

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.

```
import os
import zipfile

tmp_dir_name = "tmp"

if os.path.exists(os.path.join(os.getcwd(), tmp_dir_name)) is False:
    os.mkdir(os.path.join(os.getcwd(), tmp_dir_name))

tmp_dir_path = os.path.join(os.getcwd(), tmp_dir_name)

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

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):
    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_path):
            (category, file) = generate_name(filename, i)
            if os.path.exists(os.path.join(os.getcwd(), path, category)) is
False:
                os.mkdir(os.path.join(os.getcwd(), path, category))
            os.rename(os.path.join(tmp_dir_path, filename),
os.path.join(os.getcwd(), path, category, file))

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

def merge_files_in_folder(folder_path, bins = 1):
    try:
        log = ''
        out_file_name = ".".join([os.path.basename(folder_path), 'data'])
        with open(os.path.join(folder_path, out_file_name), 'w+', encoding='utf-
8') as out_file:
            for file, i in zip(os.listdir(folder_path),
range(len(os.listdir(folder_path)))):
                if file != out_file_name:
```

```
                    with open(os.path.join(folder_path, file), 'r',
encoding='utf-8') as f:
                        out_file.writelines(f.readlines())
                    log += '...' + file
    except Exception as ex:
        print(ex)
    finally:
        print(log)


process_user_data(['user_1', 'user_2', 'user_3', 'user_4', 'user_5', 'user_6',
'user_7', 'user_8'])

for i in range(1, 9):
    merge_files_in_folder(".\\user_" + str(i) + "\\location")
    merge_files_in_folder(".\\user_" + str(i) + "\\bt")
    merge_files_in_folder(".\\user_" + str(i) + "\\wifi")
    merge_files_in_folder(".\\user_" + str(i) + "\\app")
```

Сценарий для предварительной обработки данных и формирования признаков для модуля WIFI.

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

def replace_commas_in_line(line):
    pos = line.find(',CONN,')
    if pos == -1:
        return line[:79] + line[79 : line.find('[') - 1].replace(',', '_') +
line[line.find('[') - 1 :]
    else:
        begin_pos = line.find('\"')
        return line[:begin_pos] + line[begin_pos:].replace(',',
'_').replace('\"', '')

def preprocess_lines(lines):
    new_lines = []
    for line in lines:
        sublines = re.split(r'([0-3][0-9].[0-1][0-9].2020_[0-2][0-9]:[0-6][0-
9]:[0-6][0-9].[0-9][0-9][0-9])+', line)
        if (sublines[0] == ''):
            sublines = sublines[1:]

        if len(sublines) > 2:
            for i in range(0, len(sublines), 2):
                tmp_line = ""
                tmp_line += sublines[i] + sublines[i + 1] + '\n'
                new_lines.append(replace_commas_in_line(tmp_line))
        else:
            new_lines.append(replace_commas_in_line(line))

    return new_lines

def wifi_file_process(filepath):
```

```
    new_files_dir = "_generated"

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

    lines = preprocess_lines(lines)

    with open(filepath, 'w', encoding = 'utf-8') as f:
        f.writelines(lines)

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

    new_lines = []
    for line in lines:
        new_lines.append(replace_commas_in_line(line))

    lines = new_lines

    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))

    new_file_path = os.path.join(os.getcwd(), new_files_dir,
os.path.basename(filepath))
    new_conn_file_path = os.path.join(os.getcwd(), new_files_dir,
"_".join(["conn", os.path.basename(filepath)]))

    with open(new_file_path, 'w+', encoding='utf-8') as f:
        for line, i in zip(lines, range(len(lines))):
            if line.find(',CONN,') == -1:
                f.write(line)

    with open(new_conn_file_path, 'w+', encoding='utf-8') as f:
        for line, i in zip(lines, range(len(lines))):
            if line.find(',CONN,') != -1:
                f.write(line)

    return {'BASE': new_file_path, 'CONN': new_conn_file_path}

def wifi_make_dfs(file_path, conn_file_path, sampling_freq, rolling = False,
sampling = True):
    df = pd.read_csv(file_path, index_col = False, header = None, low_memory =
False,                      names = ['timestamp', 'uuid', 'bssid', 'ssid',
'capabilities',                                  'freq0', 'freq1', 'chwidth',
'freq', 'level', 'operator',                                    'time', 'venue',
'802', 'passpoint'])

    new_files_dir = "_datasets"
    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))

    new_files_dir += "\\" + sampling_freq
    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))

    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', 'operator', 'venue', '802', 'time',
'ssid', 'capabilities',                    'passpoint', 'freq0', 'freq1'], 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(conn_file_path, index_col = False, header = None,
low_memory = False,                          names = ['timestamp', 'uuid',
'stamp', 'bssid', '1', '2', '3', '4', '5', '6', 'level', '8'])

    df_conn = df_conn.drop(df_conn.columns.difference(['bssid','timestamp',
'level']), axis = 1)
    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

    WINDOW_SIZE = 3

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

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

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

    WINDOW_SIZE = 3

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

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

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

    WINDOW_SIZE = 3

    multi_speed_cols = ["occured_nets_count_mean", "jaccard_index_mean",
"occur_speed_mean", "disappeared_nets_count_mean",
                        "conn_level_speed_mean", "count_speed_mean"]

    for col in multi_speed_cols:
        df_quantum = calc_single_cols_in_window(df_quantum, col,
"_".join(["acceleration", col]), window, func)

    def agg_str(col):
        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)

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

    common_cols = df_quantum.columns[1:5]
    speed_acc_cols = df_quantum.columns[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 }

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

    index = os.path.basename(file_path).split('_')[-1][0]

    df_sampling.to_csv(new_files_dir + "\\wifi_sampling_dataset_" + index +
".csv")

    df_rolling = df_quantum.rolling(sampling_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_rolling = df_rolling.dropna()
    df_rolling = df_rolling.fillna(0)

    df_rolling.to_csv(new_files_dir + "\\wifi_rolling_dataset_" + index +
".csv")

def wifi_pipeline(files, sampling_freq):
    logs = []
    for file in files:
        logs.append(wifi_file_process(file))
    for t in logs:
        print(t['BASE'], sampling_freq)
        wifi_make_dfs(t['BASE'], t['CONN'], sampling_freq, True, True)

SAMPLING_FREQs = ['5s', '10s', '30s', '60s', '90s', '120s', '240s', '600s']
data_list = [
    ".\\raw_data\\wifi_1.data",
    ".\\raw_data\\wifi_2.data",
    ".\\raw_data\\wifi_3.data",
    ".\\raw_data\\wifi_4.data",
    ".\\raw_data\\wifi_5.data",
    ".\\raw_data\\wifi_6.data",
    ".\\raw_data\\wifi_7.data",
    ".\\raw_data\\wifi_8.data"
]

for freq in SAMPLING_FREQs:
    wifi_pipeline(data_list, freq)
```

Сценарий для предварительной обработки данных и формирования признаков для модуля BT.

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

```
from geopy.distance import distance as geodist

def replace_commas_in_line(line):
    if line.find(',CONN,') != -1:
        return ''
    if line.find(',BONDED,') != -1:
        pos = line.find(',BONDED,') + 1
        pos = line.find(',', pos)
        end_pos = line.find(':', pos + 1)
        comma_pos = line.find(',', pos + 1, end_pos)
        if comma_pos == -1:
            end_pos = line.find(',', end_pos + 1) + 3
        pos += 1
        end_pos -= 3
        return line[:pos] + line[pos:end_pos].replace(',', '_') + line[end_pos:]
    if line.find(',GATT,') != -1:
        pos = line.find(',GATT,') + 1
        pos = line.find(',', pos)
        end_pos = line.find(':', pos + 1)
        comma_pos = line.find(',', pos + 1, end_pos)
        if comma_pos == -1:
            end_pos = line.find(',', end_pos + 1) + 3
        pos += 1
        end_pos -= 3
        return line[:pos] + line[pos:end_pos].replace(',', '_') + line[end_pos:]
    if line.find(',GATT_SERVER,') != -1:
        pos = line.find(',GATT_SERVER,') + 1
        pos = line.find(',', pos)
        end_pos = line.find(':', pos + 1)
        comma_pos = line.find(',', pos + 1, end_pos)
        if comma_pos == -1:
            end_pos = line.find(',', end_pos + 1) + 3
        pos += 1
        end_pos -= 3
        return line[:pos] + line[pos:end_pos].replace(',', '_') + line[end_pos:]
    if line.find(',LE,') != -1:
        return line

    pos = line.find(',')
    end_pos = line.find(':', pos + 1)
    comma_pos = line.find(',', pos + 1, end_pos)
    if comma_pos == -1:
        end_pos = line.find(',', end_pos + 1) + 3
    pos += 1
    end_pos -= 3
    return line[:pos] + line[pos:end_pos].replace(',', '_') + line[end_pos:]

def preprocess_lines(lines):
    new_lines = []
    for line in lines:
        sublines = re.split(r'([0-3][0-9].[0-1][0-9].2020_[0-2][0-9]:[0-6][0-
9]:[0-6][0-9].[0-9][0-9][0-9])+', line)
        if (sublines[0] == ''):
            sublines = sublines[1:]

        if len(sublines) > 2:
            for i in range(0, len(sublines), 2):
                tmp_line = ""
                tmp_line += sublines[i] + sublines[i + 1] + '\n'
                new_lines.append(tmp_line)
        else:
            new_lines.append(line)
```

```
    return new_lines

def bt_file_process(filepath):

    new_files_dir = "_generated"

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

    lines = preprocess_lines(lines)

    with open(filepath, 'w', encoding = 'utf-8') as f:
        f.writelines(lines)

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

    new_lines = []
    for line in lines:
        new_lines.append(replace_commas_in_line(line))

    lines = new_lines

    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))

    new_file_path = os.path.join(os.getcwd(), new_files_dir,
os.path.basename(filepath))
    new_bonded_file_path = os.path.join(os.getcwd(), new_files_dir,
"_".join(["bonded", os.path.basename(filepath)]))
    new_le_file_path = os.path.join(os.getcwd(), new_files_dir, "_".join(["le",
os.path.basename(filepath)]))
    new_gatt_file_path = os.path.join(os.getcwd(), new_files_dir,
"_".join(["gatt", os.path.basename(filepath)]))
    new_gatt_server_file_path = os.path.join(os.getcwd(), new_files_dir,
"_".join(["gatt_server", os.path.basename(filepath)]))

    with open(new_file_path, 'w+', encoding='utf-8') as f:
        for line, i in zip(lines, range(len(lines))):
            if line.find(',BONDED,') == -1 and line.find(',GATT,') == -1 and
line.find(',GATT_SERVER,') == -1 and line.find(',LE,') == -1:
                f.write(line)

    with open(new_le_file_path, 'w+', encoding='utf-8') as f:
        for line, i in zip(lines, range(len(lines))):
            if line.find(',LE,') != -1:
                f.write(line)

    return {'BASE': new_file_path, 'LE': new_le_file_path}

def bt_make_dataframes(file_path, le_file_path, sampling_freq, rolling = False,
sampling = True):
    df = pd.read_csv(file_path, index_col = False, header = None, low_memory =
False,                          names = ['timestamp', 'ssid', 'bssid',
'major_class', 'class',                                         'bond_state',
'type'])

    new_files_dir = "_datasets"
    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))
```

```
    new_files_dir += "\\" + sampling_freq
    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))

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

    one_hot_maj_class = pd.get_dummies(df['major_class'], prefix='one_hot_c')
    df = df.join(one_hot_maj_class)

    one_hot_bond = pd.get_dummies(df['bond_state'], prefix='one_hot_b')
    df = df.join(one_hot_bond)

    one_hot_type = pd.get_dummies(df['type'], prefix='one_hot_t')
    df = df.join(one_hot_type)

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

    df = df.drop(['timestamp', 'ssid', '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(le_file_path, index_col = False, header = None,
low_memory = False,                          names = ['timestamp', 'stamp', '1',
'2', 'level', '3', 'connectable', '8'])

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

    WINDOW_SIZE = 3

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

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

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

    WINDOW_SIZE = 3

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

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

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

    WINDOW_SIZE = 3

    multi_speed_cols = ["occured_devices_count_mean", "jaccard_index_mean",
"occur_speed_mean", "disappeared_devices_count_mean",
                        "count_speed_mean"]

    for col in multi_speed_cols:
        df_quantum = calc_single_cols_in_window(df_quantum, col,
"_".join(["acceleration", col]), window, func)

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

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

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

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

    def str_skew(col):
        return stats.skew(agg_str(col))
```

```
    def str_kurt(col):
        return stats.kurtosis(agg_str(col))

    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 = df_quantum.drop(['bssid', 'timestamp'], 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 =
sampling_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_sampling = df_sampling.dropna()
    df_sampling = df_sampling.fillna(0)

    index = os.path.basename(file_path).split('_')[-1][0]

    df_sampling.to_csv(new_files_dir + "\\bt_sampling_dataset_" + index +
".csv")

    df_rolling = df_quantum.rolling(sampling_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_rolling = df_rolling.dropna()
    df_rolling = df_rolling.fillna(0)

    df_rolling.to_csv(new_files_dir + "\\bt_rolling_dataset_" + index + ".csv")

def bt_pipeline(files, sampling_freq):
    logs = []
    for file in files:
```

```
        logs.append(bt_file_process(file))
    for t in logs:
        print(t['BASE'], sampling_freq)
        bt_make_dataframes(t['BASE'], t['LE'], sampling_freq, True, True)

SAMPLING_FREQs = ['5s', '10s', '30s', '60s', '90s', '120s', '240s', '600s']
data_list = [
    ".\\raw_data\\bt_1.data",
    ".\\raw_data\\bt_2.data",
    ".\\raw_data\\bt_3.data",
    ".\\raw_data\\bt_4.data",
    ".\\raw_data\\bt_5.data",
    ".\\raw_data\\bt_6.data",
    ".\\raw_data\\bt_7.data",
    ".\\raw_data\\bt_8.data"
]

for freq in SAMPLING_FREQs:
    bt_pipeline(data_list, freq)
```

Сценарий для предварительной обработки данных и формирования признаков для модуля LOCATION.

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

REGEX_LOG_STRING_PATTERN =     r'([0-3]\d.[0-1]\d.2020_[0-2]\d:[0-6]\d:[0-
6]\d.\d{3}),(-?\d+,\d+),(-?\d+,\d+),(-?\d+,\d+),(-?\d+,\d+),fused'

def replace_commas_in_line(line):
    matches = re.findall(REGEX_LOG_STRING_PATTERN, line)
    if len(matches) > 0:
        matches = matches[0]
    else:
        return line
    matches = [x for x in map(lambda x: x.replace(',','.'), matches)]
    return ",".join(matches) + '\n'

def preprocess_lines(lines):
    new_lines = []
    for line in lines:
        sublines = re.split(r'([0-3]\d.[0-1]\d.2020_[0-2]\d:[0-6]\d:[0-
6]\d.\d{3})+', line)
        if (sublines[0] == ''):
            sublines = sublines[1:]

        if len(sublines) > 2:
            for i in range(0, len(sublines), 2):
                tmp_line = ""
                tmp_line += sublines[i] + sublines[i + 1] + '\n'
                new_lines.append(tmp_line)
        else:
            new_lines.append(line)
```

```
    return new_lines

def location_file_process(filepath):

    new_files_dir = "_generated"

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

    lines = preprocess_lines(lines)

    with open(filepath, 'w', encoding = 'utf-8') as f:
        f.writelines(lines)

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

    new_lines = []
    for line in lines:
        new_lines.append(replace_commas_in_line(line))

    lines = new_lines

    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))

    new_file_path = os.path.join(os.getcwd(), new_files_dir,
os.path.basename(filepath))

    with open(new_file_path, 'w+', encoding='utf-8') as f:
        for line, i in zip(lines, range(len(lines))):
            f.write(line)

    return {'BASE': new_file_path}

def location_make_dataframes(file_path, sampling_freq, rolling = False, sampling
= True):
    df = pd.read_csv(file_path, index_col = False, header = None, low_memory =
False,                      names = ['timestamp', 'accuracy', 'altitude',
'latitude', 'longitude'])

    new_files_dir = "_datasets"
    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))

    new_files_dir += "\\" + sampling_freq
    if os.path.exists(os.path.join(os.getcwd(), new_files_dir)) is False:
        os.mkdir(os.path.join(os.getcwd(), new_files_dir))

    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['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'], axis=1)

    df['prev_speed'] = df['speed'].shift(1)
    df['prev_altitude_speed'] = df['altitude_speed'].shift(1)

    def get_acceleration(row):
        prev_speed = row['prev_speed']
        curr_speed = row['speed']
        delta = row['timestamp'] - row['prev_timestamp']
        if pd.isnull(delta):
            return np.nan
        time = abs(delta.total_seconds())
        if np.isnan(prev_speed) or np.isnan(curr_speed):
            return np.nan
        if time == 0:
            return np.nan
        return curr_speed - prev_speed / time

    def get_altitude_acceleration(row):
        prev_speed = row['prev_altitude_speed']
        curr_speed = row['altitude_speed']
        delta = row['timestamp'] - row['prev_timestamp']
        if pd.isnull(delta):
            return np.nan
        time = abs(delta.total_seconds())
        if np.isnan(prev_speed) or np.isnan(curr_speed):
            return np.nan
        if time == 0:
            return np.nan
        return curr_speed - prev_speed / time

    df['acc'] = df.apply(lambda row: get_acceleration(row), axis=1)
    df['altitude_acc'] = df.apply(lambda row: get_altitude_acceleration(row),
axis=1)
```

```
    df = df.drop(['prev_altitude_speed', 'prev_speed', '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,
        'acc': common_funcs_list,
        'altitude_acc': common_funcs_list
    }

    df_sampling = df.groupby(pd.Grouper(freq = sampling_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_sampling = df_sampling.dropna()
    df_sampling = df_sampling.fillna(0)

    index = os.path.basename(file_path).split('_')[-1][0]

    df_sampling.to_csv(new_files_dir + "\\location_sampling_dataset_" + index +
".csv")

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

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

    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_rolling.to_csv(new_files_dir + "\\location_rolling_dataset_" + index +
".csv")

def location_pipeline(files, sampling_freq):
    logs = []
    for file in files:
        logs.append(location_file_process(file))
    for t in logs:
        print(t['BASE'], sampling_freq)
        location_make_dataframes(t['BASE'], sampling_freq, True, True)

SAMPLING_FREQs = ['5s', '10s', '30s', '60s', '90s', '120s', '240s', '600s']
data_list = [
    ".\\raw_data\\location_1.data",
    ".\\raw_data\\location_2.data",
    ".\\raw_data\\location_3.data",
    ".\\raw_data\\location_4.data",
    ".\\raw_data\\location_5.data", ".\\raw_data\\location_6.data",
    ".\\raw_data\\location_7.data", ".\\raw_data\\location_8.data",
]

for freq in SAMPLING_FREQs:
    location_pipeline(data_list, freq)
```

Приложение В

Исходный код сценариев, использованных для обучения моделей

В данном приложении приведён исходный код на языке Python, с помощью которого проводилось обучение и тестирование моделей. Ввиду идентичности алгоритмов кросс-валидации и финальной проверки для всех модулей, приложена только часть, использованная для тестирования моделей на данных, полученных с модуля LOCATION.

```
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, roc_auc_score, f1_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

def make_dataframe_impl(df_count, rolling=True):
    dfs_list = []
    dfs_rows_len_list = []

    file_name = ""
    if rolling is True:
        file_name = ".\\_datasets\\60s\\location_rolling_dataset_"
    else:
        file_name = ".\\_datasets\\60s\\location_sampling_dataset_"

    for i in range(1, df_count + 1):
        df = pd.read_csv(file_name + str(i) + ".csv")
        df = df.drop(["timestamp"], axis=1)

        df = (df - df.min()) / (df.max() - df.min())

        df["user"] = i
        dfs_list.append(df)
        dfs_rows_len_list.append(df.shape[0])

    df = pd.concat(dfs_list, ignore_index=True)
    return df

def make_common_rolling_dataframe(df_count):
    return make_dataframe_impl(df_count, True)


def make_common_sampling_dataframe(df_count):
    return make_dataframe_impl(df_count, False)

def drop_bad_rows(df):
    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])) > 2.5
                    ].index
                ):
                    bad_rows.add(x)

            for x in list(df[col][np.abs(stats.zscore(df[col])) > 2.5].index):
                bad_rows.add(x)

    df.drop(list(bad_rows), axis=0, inplace=True)

    return df

def drop_bad_cols(df):
    bad_cols = set()
    for col in df.columns:
        if col != "user":
            if df[df[col] != df[col].mean()].shape[0] < 0.1 * 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]
                    < 0.1 * df.loc[df.user == user, :].shape[0]
                ):
                    bad_cols.add(col)

                elif (
                    np.sum(np.abs(stats.zscore(df.loc[df.user == user, col])) <
2.5)
                    < 0.9 * df.loc[df.user == user, col].shape[0]
                ):
                    bad_cols.add(col)

    df.drop(list(bad_cols), axis=1, inplace=True)

    return df

def resample(df):
    sampling_dfs = []
    need_count = 0

    for label, count in zip(df.user.value_counts().index,
df.user.value_counts().values):
        if need_count == 0:
            need_count = count
            df_ = df[df.user == label]
            sampling_dfs.append(df_)
        else:
            df_ = df[df.user == label]
            df_over = df_.sample(need_count, replace=True, random_state=42)
            sampling_dfs.append(df_over)

    new_df = pd.concat(sampling_dfs)
    new_df = new_df.reset_index()
```

```
    return pd.concat(sampling_dfs)

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

df = make_common_rolling_dataframe(8)
df.columns.to_list()
features = [
    "accuracy_mean",
    "accuracy_var",
    "accuracy_median",
    "accuracy_skew",
    "accuracy_kurt",
    "accuracy_std",
    "speed_mean",
    "speed_var",
    "speed_median",
    'speed_skew',
    "speed_kurt",
    "speed_std",
    "altitude_speed_mean",
    "altitude_speed_var",
    "altitude_speed_median",
    'altitude_speed_skew',
    "altitude_speed_kurt",
    "altitude_speed_std",
    "acc_mean",
    "acc_var",
    'acc_median',
    'acc_skew',
    "acc_kurt",
    "acc_std",
    'altitude_acc_mean',
    "altitude_acc_var",
    "altitude_acc_median",
    'altitude_acc_skew',
    "altitude_acc_kurt",
    "altitude_acc_std",
```

```
    "user",
]

df = df.drop(df.columns.difference(features), axis=1)
df = df.dropna()

df = drop_bad_cols(df)
df = drop_bad_rows(df)

fig = plt.figure(figsize=(25, 25))
plt.matshow(df.corr(), fignum=fig.number)
plt.xticks(range(df.shape[1]), df.columns, fontsize=18, rotation=90)
plt.yticks(range(df.shape[1]), df.columns, fontsize=18)
cb = plt.colorbar()
cb.ax.tick_params(labelsize=16)
plt.title("Correlation matrix", fontsize=20, y=-0.03)

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]) > 0.7) and column != "user"]
df = df.drop(corr_cols, axis=1)

sample = df.sample(10000)
plt.figure(figsize=(16, 10))
plt.scatter(x=sample['altitude_speed_mean'], y=sample['accuracy_kurt'],
alpha=0.5, c=sample.user, cmap='magma')
plt.colorbar()
plt.show()

# for user in df.user.unique():
#     for valid_user in df.user.unique():
#         if user != valid_user:
#             print('-----------------------------------------------------------------
----------------')
#             print('Valid user: ', valid_user, 'Extracted user: ', user)
#             print('-----------------------------------------------------------------
----------------')
#             df1, df_ = extract_delayed_user(df.copy(), user)
#             df1['user'] = 0
#             df_ = split_users_into_two_classes(df_.copy(), valid_user)
#             df_ = resample(df_)

#             dataset = df_.to_numpy()
#             X = dataset[:, :-1]
#             y = dataset[:, -1]

#             X_test = df1.to_numpy()[:, :-1]
#             y_test = df1.to_numpy()[:, -1]

#             model.fit(X, y, verbose=False)

#             preds_class = model.predict(X_test)
#             print('Accuracy: ', accuracy_score(preds_class, y_test))

#             sum_ = 0
#             imp = [ (x, i) for x, i in zip(model.feature_importances_,
range(len(model.feature_importances_)) )]
#             sorted_ = sorted(imp, key=lambda tup: tup[0])
#             for i in range(len(sorted_)):
#                 if sorted_[i][0] > 5:
```

```
#                     print(sorted_[i][1], ': ', df_.columns[sorted_[i][1]], ' -
', sorted_[i][0])

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

# ## CatBoostClassifier CV

iterations = 100
depth = 6
loss_function = 'Logloss'
l2_leaf_reg = 1
leaf_estimation_iterations = 5
logging_level = 'Silent'

df['labels'] = df['user']

CV_CATBOOST_BIG_DICT = {}

for user in df.labels.unique():
    print("Valid User: ", user)
    print("-------------------------------------------------------------------
-----------")

    df_ = resample(df.copy())
    df_ = split_users_into_two_classes(df_.copy(), user)
    df_ = resample(df_)

    group_labels = df_.labels.to_numpy().copy()
    df_ = df_.drop('labels', axis=1)

    dataset = df_.to_numpy().copy()
    X = dataset[:, :-1]
    y = dataset[:, -1]

    cv_split = PredefinedSplit(test_fold=get_cv_split(X, y, group_labels, user))
    scoring = ('accuracy', 'balanced_accuracy')

    model = CatBoostClassifier(iterations=iterations, depth=depth,
loss_function=loss_function,
l2_leaf_reg=l2_leaf_reg, leaf_estimation_iterations=leaf_estimation_iterations,
logging_level=logging_level)

    cv_results = cross_validate(model, X, y, scoring=scoring, cv=cv_split,
n_jobs=-1)
    accuracy = cv_results['test_accuracy']

    CV_CATBOOST_BIG_DICT[str(user)] = {}
    CV_CATBOOST_BIG_DICT[str(user)]["accuracy"] = accuracy.copy()
    CV_CATBOOST_BIG_DICT[str(user)]["mean_accuracy"] = np.mean(accuracy).copy()
    CV_CATBOOST_BIG_DICT[str(user)]["max_accuracy"] = np.max(accuracy).copy()
    CV_CATBOOST_BIG_DICT[str(user)]["min_accuracy"] = np.min(accuracy).copy()

    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("-------------------------------------------------------------------
-----------")
```

```
# ## CatBoostClassifier Final Validation

iterations = 100
depth = 6
loss_function = 'Logloss'
l2_leaf_reg = 1
leaf_estimation_iterations = 5
logging_level = 'Silent'

df["labels"] = df["user"]

VALIDATION_CATBOOST_BIG_DICT = {}

for user in df.labels.unique():
    print("Valid User: ", user)
    print("----------------------------------------------------------------------
-----------")

    VALIDATION_CATBOOST_BIG_DICT[str(user)] = {}

    for ex_user in df.labels.unique():
        if ex_user != user:

            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_ = resample(df_.copy())
            df_ = split_users_into_two_classes(df_.copy(), user)
            df_ = resample(df_)

            df_ = df_.drop("labels", axis=1)

            dataset = df_.to_numpy().copy()
            np.random.shuffle(dataset)

            X = dataset[:, :-1]
            y = dataset[:, -1]

            model = CatBoostClassifier(iterations=iterations, depth=depth,
loss_function=loss_function,
l2_leaf_reg=l2_leaf_reg, leaf_estimation_iterations=leaf_estimation_iterations,
logging_level=logging_level)
            model.fit(X, y)

            # Testing

            test_df = pd.concat(df_for_test)
```

```
            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]

            min_others_test_count = min(others_in_test_count)

            is_important_min = True
            if min_others_test_count <= ex_user_in_test_count and
min_others_test_count <= valid_user_in_test_count:
                is_important_min = False

            new_df_parts = []
            if is_important_min is True:
                part_size = min(valid_user_in_test_count, ex_user_in_test_count)
                other_sample_size = part_size // len(others_in_test_count) + 1

            else:
                part_size_can_be = min_others_test_count *
len(others_in_test_count)

                if part_size_can_be > min(valid_user_in_test_count,
ex_user_in_test_count):
                    part_size = min(valid_user_in_test_count,
ex_user_in_test_count)
                    other_sample_size = part_size // len(others_in_test_count) +
1
                else:
                    part_size = part_size_can_be
                    other_sample_size = min_others_test_count

            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())

            for x in test_df.labels.unique():
                if x != user and x != ex_user:
                    new_df_parts.append(test_df[test_df.labels ==
x].sample(other_sample_size).copy())

            test_df = pd.concat(new_df_parts)

            test_df.loc[test_df.labels != user, "user"] = 0
            test_df.loc[test_df.labels == user, "user"] = 1

            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()

            VALIDATION_CATBOOST_BIG_DICT[str(user)][ex_user] = {}
            VALIDATION_CATBOOST_BIG_DICT[str(user)][ex_user]["y_test"] =
y_test.copy()
```

```
            VALIDATION_CATBOOST_BIG_DICT[str(user)][ex_user]["y_predict"] =
model.predict(X_test).copy()
            VALIDATION_CATBOOST_BIG_DICT[str(user)][ex_user]["y_proba"] =
model.predict_proba(X_test).copy()

            print("Valid user = ", user, ", Extracted user = ", ex_user,
"accuracy = ",

accuracy_score(VALIDATION_CATBOOST_BIG_DICT[str(user)][ex_user]["y_test"],

VALIDATION_CATBOOST_BIG_DICT[str(user)][ex_user]["y_predict"])
                )

            print("----------------------------------------------------------------
--------------------")


# ## RandomForestClassifier CV

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'

df['labels'] = df['user']

CV_RFC_BIG_DICT = {}

for user in df.labels.unique():
    print("Valid User: ", user)
    print("----------------------------------------------------------------------
-----------")

    df_ = resample(df.copy())
    df_ = split_users_into_two_classes(df_.copy(), user)
    df_ = resample(df_)

    group_labels = df_.labels.to_numpy().copy()
    df_ = df_.drop('labels', axis=1)

    dataset = df_.to_numpy().copy()
    X = dataset[:, :-1]
    y = dataset[:, -1]

    cv_split = PredefinedSplit(test_fold=get_cv_split(X, y, group_labels, user))
    scoring = ('accuracy', 'balanced_accuracy')

    model = RandomForestClassifier(n_estimators=n_estimators,
criterion=criterion, max_depth=max_depth,
min_samples_split=min_samples_split, min_samples_leaf=min_samples_leaf,
max_features=max_features, n_jobs=n_jobs, class_weight=class_weight)

    cv_results = cross_validate(model, X, y, scoring=scoring, cv=cv_split,
n_jobs=-1)
    accuracy = cv_results['test_accuracy']

    CV_RFC_BIG_DICT[str(user)] = {}
    CV_RFC_BIG_DICT[str(user)]["accuracy"] = accuracy.copy()
```

```
    CV_RFC_BIG_DICT[str(user)]["mean_accuracy"] = np.mean(accuracy).copy()
    CV_RFC_BIG_DICT[str(user)]["max_accuracy"] = np.max(accuracy).copy()
    CV_RFC_BIG_DICT[str(user)]["min_accuracy"] = np.min(accuracy).copy()

    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("------------------------------------------------------------------------
-----------")


# ## RandomForestClassifier Final Validation

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'

df["labels"] = df["user"]

VALIDATION_RFC_BIG_DICT = {}

for user in df.labels.unique():
    print("Valid User: ", user)
    print("------------------------------------------------------------------------
-----------")

    VALIDATION_RFC_BIG_DICT[str(user)] = {}

    for ex_user in df.labels.unique():
        if ex_user != user:

            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_ = resample(df_.copy())
            df_ = split_users_into_two_classes(df_.copy(), user)
            df_ = resample(df_)

            df_ = df_.drop("labels", axis=1)

            dataset = df_.to_numpy().copy()
            np.random.shuffle(dataset)
```

```
            X = dataset[:, :-1]
            y = dataset[:, -1]

            model = RandomForestClassifier(n_estimators=n_estimators,
criterion=criterion, max_depth=max_depth,
min_samples_split=min_samples_split, min_samples_leaf=min_samples_leaf,
max_features=max_features, n_jobs=n_jobs, class_weight=class_weight)
            model.fit(X, y)

            # Testing

            test_df = pd.concat(df_for_test)

            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]

            min_others_test_count = min(others_in_test_count)

            is_important_min = True
            if min_others_test_count <= ex_user_in_test_count and
min_others_test_count <= valid_user_in_test_count:
                is_important_min = False

            new_df_parts = []
            if is_important_min is True:
                part_size = min(valid_user_in_test_count, ex_user_in_test_count)
                other_sample_size = part_size // len(others_in_test_count) + 1

            else:
                part_size_can_be = min_others_test_count *
len(others_in_test_count)

                if part_size_can_be > min(valid_user_in_test_count,
ex_user_in_test_count):
                    part_size = min(valid_user_in_test_count,
ex_user_in_test_count)
                    other_sample_size = part_size // len(others_in_test_count) +
1
                else:
                    part_size = part_size_can_be
                    other_sample_size = min_others_test_count

            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())

            for x in test_df.labels.unique():
                if x != user and x != ex_user:
                    new_df_parts.append(test_df[test_df.labels ==
x].sample(other_sample_size).copy())

            test_df = pd.concat(new_df_parts)

            test_df.loc[test_df.labels != user, "user"] = 0
            test_df.loc[test_df.labels == user, "user"] = 1

            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()

            VALIDATION_RFC_BIG_DICT[str(user)][ex_user] = {}
            VALIDATION_RFC_BIG_DICT[str(user)][ex_user]["y_test"] =
y_test.copy()
            VALIDATION_RFC_BIG_DICT[str(user)][ex_user]["y_predict"] =
model.predict(X_test).copy()
            VALIDATION_RFC_BIG_DICT[str(user)][ex_user]["y_proba"] =
model.predict_proba(X_test).copy()

            print("Valid user = ", user, ", Extracted user = ", ex_user,
"accuracy = ",

accuracy_score(VALIDATION_RFC_BIG_DICT[str(user)][ex_user]["y_test"],

VALIDATION_RFC_BIG_DICT[str(user)][ex_user]["y_predict"])
                  )

            print("-------------------------------------------------------------
-------------------")


# ## SVC CV

C = 10
kernel = 'rbf'
degree = 1
gamma = 5

df['labels'] = df['user']

CV_SVC_BIG_DICT = {}

for user in df.labels.unique():
    print("Valid User: ", user)
    print("-------------------------------------------------------------------
-----------")

    df_ = resample(df.copy())
    df_ = split_users_into_two_classes(df_.copy(), user)
    df_ = resample(df_)

    df_.loc[df_.user == 0, 'user'] = -1

    group_labels = df_.labels.to_numpy().copy()
    df_ = df_.drop('labels', axis=1)

    dataset = df_.to_numpy().copy()
    X = dataset[:, :-1]
    y = dataset[:, -1]

    cv_split = PredefinedSplit(test_fold=get_cv_split(X, y, group_labels, user))
    scoring = ('accuracy', 'balanced_accuracy')

    model = SVC(C=C, kernel=kernel, degree=degree, gamma=gamma)

    cv_results = cross_validate(model, X, y, scoring=scoring, cv=cv_split,
n_jobs=-1)
    accuracy = cv_results['test_accuracy']
```

```
    CV_SVC_BIG_DICT[str(user)] = {}
    CV_SVC_BIG_DICT[str(user)]["accuracy"] = accuracy.copy()
    CV_SVC_BIG_DICT[str(user)]["mean_accuracy"] = np.mean(accuracy).copy()
    CV_SVC_BIG_DICT[str(user)]["max_accuracy"] = np.max(accuracy).copy()
    CV_SVC_BIG_DICT[str(user)]["min_accuracy"] = np.min(accuracy).copy()

    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("------------------------------------------------------------------------
-----------")


# ## SVC Final Validation

C = 10
kernel = 'rbf'
degree = 1
gamma = 5

df["labels"] = df["user"]

VALIDATION_SVC_BIG_DICT = {}

for user in df.labels.unique():
    print("Valid User: ", user)
    print("------------------------------------------------------------------------
-----------")

    VALIDATION_SVC_BIG_DICT[str(user)] = {}

    for ex_user in df.labels.unique():
        if ex_user != user:

            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_ = resample(df_.copy())
            df_ = split_users_into_two_classes(df_.copy(), user)
            df_ = resample(df_)

            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]

            model = SVC(C=C, kernel=kernel, degree=degree, gamma=gamma)
            model.fit(X, y)

            # Testing

            test_df = pd.concat(df_for_test)

            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]

            min_others_test_count = min(others_in_test_count)

            is_important_min = True
            if min_others_test_count <= ex_user_in_test_count and
min_others_test_count <= valid_user_in_test_count:
                is_important_min = False

            new_df_parts = []
            if is_important_min is True:
                part_size = min(valid_user_in_test_count, ex_user_in_test_count)
                other_sample_size = part_size // len(others_in_test_count) + 1

            else:
                part_size_can_be = min_others_test_count *
len(others_in_test_count)

                if part_size_can_be > min(valid_user_in_test_count,
ex_user_in_test_count):
                    part_size = min(valid_user_in_test_count,
ex_user_in_test_count)
                    other_sample_size = part_size // len(others_in_test_count) +
1
                else:
                    part_size = part_size_can_be
                    other_sample_size = min_others_test_count

            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())

            for x in test_df.labels.unique():
                if x != user and x != ex_user:
                    new_df_parts.append(test_df[test_df.labels ==
x].sample(other_sample_size).copy())

            test_df = pd.concat(new_df_parts)

            test_df.loc[test_df.labels != user, "user"] = -1
            test_df.loc[test_df.labels == user, "user"] = 1

            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()
```

```
            VALIDATION_SVC_BIG_DICT[str(user)][ex_user] = {}
            VALIDATION_SVC_BIG_DICT[str(user)][ex_user]["y_test"] =
y_test.copy()
            VALIDATION_SVC_BIG_DICT[str(user)][ex_user]["y_predict"] =
model.predict(X_test).copy()
            VALIDATION_SVC_BIG_DICT[str(user)][ex_user]["y_proba"] =
model.decision_function(X_test).copy()

            print("Valid user = ", user, ", Extracted user = ", ex_user,
"accuracy = ",

accuracy_score(VALIDATION_SVC_BIG_DICT[str(user)][ex_user]["y_test"],

VALIDATION_SVC_BIG_DICT[str(user)][ex_user]["y_predict"])
                  )

            print("-------------------------------------------------------------
--------------------")


# ## LogReg CV

penalty = 'l2'
C = 0.01
solver = 'newton-cg'
max_iter = 1000
n_jobs = -1

df['labels'] = df['user']

CV_LR_BIG_DICT = {}

for user in df.labels.unique():
    print("Valid User: ", user)
    print("-------------------------------------------------------------------------
-----------")

    df_ = resample(df.copy())
    df_ = split_users_into_two_classes(df_.copy(), user)
    df_ = resample(df_)

    group_labels = df_.labels.to_numpy().copy()
    df_ = df_.drop('labels', axis=1)

    dataset = df_.to_numpy().copy()
    X = dataset[:, :-1]
    y = dataset[:, -1]

    cv_split = PredefinedSplit(test_fold=get_cv_split(X, y, group_labels, user))
    scoring = ('accuracy', 'balanced_accuracy')

    model = LogisticRegression(penalty=penalty, C=C, solver=solver,
max_iter=max_iter, n_jobs=n_jobs)

    cv_results = cross_validate(model, X, y, scoring=scoring, cv=cv_split,
n_jobs=-1)
    accuracy = cv_results['test_accuracy']

    CV_LR_BIG_DICT[str(user)] = {}
    CV_LR_BIG_DICT[str(user)]["accuracy"] = accuracy.copy()
    CV_LR_BIG_DICT[str(user)]["mean_accuracy"] = np.mean(accuracy).copy()
```

```
    CV_LR_BIG_DICT[str(user)]["max_accuracy"] = np.max(accuracy).copy()
    CV_LR_BIG_DICT[str(user)]["min_accuracy"] = np.min(accuracy).copy()

    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("------------------------------------------------------------------------
-----------")


# ## LogReg Final Validation

penalty = 'l2'
C = 0.01
solver = 'newton-cg'
max_iter = 1000
n_jobs = -1

df["labels"] = df["user"]

VALIDATION_LR_BIG_DICT = {}

for user in df.labels.unique():
    print("Valid User: ", user)
    print("------------------------------------------------------------------------
-----------")

    VALIDATION_LR_BIG_DICT[str(user)] = {}

    for ex_user in df.labels.unique():
        if ex_user != user:

            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_ = resample(df_.copy())
            df_ = split_users_into_two_classes(df_.copy(), user)
            df_ = resample(df_)

            df_ = df_.drop("labels", axis=1)

            dataset = df_.to_numpy().copy()
            np.random.shuffle(dataset)

            X = dataset[:, :-1]
            y = dataset[:, -1]
```

```
            model = LogisticRegression(penalty=penalty, C=C, solver=solver,
max_iter=max_iter, n_jobs=n_jobs)
            model.fit(X, y)

            # Testing

            test_df = pd.concat(df_for_test)

            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]

            min_others_test_count = min(others_in_test_count)

            is_important_min = True
            if min_others_test_count <= ex_user_in_test_count and
min_others_test_count <= valid_user_in_test_count:
                is_important_min = False

            new_df_parts = []
            if is_important_min is True:
                part_size = min(valid_user_in_test_count, ex_user_in_test_count)
                other_sample_size = part_size // len(others_in_test_count) + 1

            else:
                part_size_can_be = min_others_test_count *
len(others_in_test_count)

                if part_size_can_be > min(valid_user_in_test_count,
ex_user_in_test_count):
                    part_size = min(valid_user_in_test_count,
ex_user_in_test_count)
                    other_sample_size = part_size // len(others_in_test_count) +
1
                else:
                    part_size = part_size_can_be
                    other_sample_size = min_others_test_count

            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())

            for x in test_df.labels.unique():
                if x != user and x != ex_user:
                    new_df_parts.append(test_df[test_df.labels ==
x].sample(other_sample_size).copy())

            test_df = pd.concat(new_df_parts)

            test_df.loc[test_df.labels != user, "user"] = 0
            test_df.loc[test_df.labels == user, "user"] = 1

            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()

            VALIDATION_LR_BIG_DICT[str(user)][ex_user] = {}
            VALIDATION_LR_BIG_DICT[str(user)][ex_user]["y_test"] = y_test.copy()
```

```
            VALIDATION_LR_BIG_DICT[str(user)][ex_user]["y_predict"] =
model.predict(X_test).copy()
            VALIDATION_LR_BIG_DICT[str(user)][ex_user]["y_proba"] =
model.predict_proba(X_test).copy()

            print("Valid user = ", user, ", Extracted user = ", ex_user,
"accuracy = ",

accuracy_score(VALIDATION_LR_BIG_DICT[str(user)][ex_user]["y_test"],

VALIDATION_LR_BIG_DICT[str(user)][ex_user]["y_predict"])
                  )

            print("----------------------------------------------------------------
--------------------")

for d in [CV_CATBOOST_BIG_DICT, CV_RFC_BIG_DICT,           CV_SVC_BIG_DICT,
CV_LR_BIG_DICT]:
    acc = []
    for user, res in d.items():
        print("Valid User: ", user)
        print("----------------------------------------------------------------
---------------")
        acc.append(res['mean_accuracy'])
        print('Mean accuracy: ', res['mean_accuracy'])
        print("----------------------------------------------------------------
---------------")
    print("MEAN: ", np.mean(acc))

for d in [VALIDATION_CATBOOST_BIG_DICT, VALIDATION_RFC_BIG_DICT,
VALIDATION_SVC_BIG_DICT, VALIDATION_LR_BIG_DICT]:

    mean_auc = []
    mean_f = []
    for user, res in d.items():
        print("Valid User: ", user)
        print("----------------------------------------------------------------
---------------")
        means_acc = []
        means_prec = []
        means_rec = []
        means_roc = []
        means_f1 = []

        for ex_user, ex_res in res.items():
            print('Ex user: ', ex_user)

print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++")

            y_true = ex_res['y_test']
            y_pred = ex_res['y_predict']
            if len(ex_res['y_proba'].shape) > 1 and ex_res['y_proba'].shape[1] >
1:
                y_proba = ex_res['y_proba'][:, 1]
            else:
                y_proba = ex_res['y_proba']

            acc = accuracy_score(y_true, y_pred)
            prec = precision_score(y_true, y_pred)
            rec = recall_score(y_true, y_pred)
            f1 = f1_score(y_true, y_pred)
```

```
            means_acc.append(acc)
            means_prec.append(prec)
            means_rec.append(rec)
            means_f1.append(f1)

            print('Accuracy: ', acc)
            print('Precision: ', prec)
            print('Recall: ', rec)
            try:
                roc = roc_auc_score(y_true, y_proba)
                means_roc.append(roc)
                print('ROC-AUC: ', roc)
            except Exception as e:
                print('ROC-AUC: skip')
            print('F1: ', f1)

print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

        print('Mean accuracy: ', sum(means_acc) / len(means_acc))
        print('mean Precision: ', sum(means_prec) / len(means_prec))
        print('mean Recall: ', sum(means_rec) / len(means_rec))
        if len(means_roc) > 0:
            print('mean ROC-AUC: ', sum(means_roc) / len(means_roc))
            mean_auc.append(sum(means_roc) / len(means_roc))
        print('mean F1: ', sum(means_f1) / len(means_f1))
        mean_f.append(sum(means_f1) / len(means_f1))

        print("----------------------------------------------------------------------------------------")
    print("MEAN_F ", np.mean(mean_f))
    print("AUC_F ", np.mean(mean_auc))
```

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	0.752	0.842	0.674	0.618
	10	0.769	0.809	0.723	0.612
	30	0.809	0.901	0.808	0.704
	60	0.850	0.907	0.816	0.686
	90	0.849	0.912	0.846	0.677
	120	0.875	0.922	0.852	0.658
	240	0.805	0.924	0.972	0.749
	600	0.802	0.884	0.919	0.729
	Лучший результат	120 с — 0.875	240 с — 0.924	240 с — 0.972	240 с — 0.749

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	0.752	0.849	0.674	0.618
	10	0.756	0.817	0.743	0.605
	30	0.837	0.907	0.769	0.725
	60	0.902	0.906	0.761	0.751
	90	0.839	0.915	0.837	0.732
	120	0.781	0.857	0.793	0.761
	240	0.820	0.848	0.811	0.776
	600	0.844	0.909	0.843	0.776
	Лучший результат	60 с — 0.902	90 с — 0.915	600 с — 0.842	240 с, 600 с — 0.776

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	0.869	0.922	0.948	0.812
	10	0.799	0.772	0.804	0.717
	30	0.861	0.985	0.936	0.818
	60	0.884	0.931	0.901	0.744
	90	0.809	0.912	0.939	0.774
	120	0.901	0.979	0.972	0.799
	240	0.898	0.960	0.951	0.727
	600	0.799	0.949	0.962	0.683
	Лучший результат	120 с — 0.901	30 с — 0.985	120 с — 0.972	30 с — 0.818

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	0.807	0.898	0.891	0.712
	10	0.824	0.902	0.901	0.737
	30	0.924	0.974	0.911	0.782
	60	0.813	0.912	0.879	0.824
	90	0.881	0.929	0.932	0.849
	120	0.861	0.959	0.942	0.821
	240	0.903	0.958	0.931	0.811
	600	0.808	0.904	0.879	0.719
	Лучший результат	30 с — 0.924	30 с — 0.974	120 с — 0.942	90 с — 0.849

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	0.869	0.931	0.839	0.726
	10	0.872	0.911	0.865	0.752
	30	0.891	0.928	0.889	0.732
	60	0.912	0.943	0.873	0.749
	90	0.923	0.956	0.901	0.803
	120	0.944	0.952	0.932	0.814
	240	0.925	0.964	0.954	0.808
	600	0.957	0.971	0.953	0.812
	Лучший результат	600 с — 0.957	600 с — 0.971	240 с — 0.954	120 с — 0.814

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	0.826	0.891	0.798	0.804
	10	0.843	0.878	0.804	0.742
	30	0.861	0.869	0.814	0.784
	60	0.867	0.927	0.778	0.736
	90	0.904	0.934	0.799	0.802
	120	0.911	0.929	0.802	0.812
	240	0.902	0.964	0.859	0.803
	600	0.907	0.963	0.906	0.808
	Лучший результат	120 с — 0.911	240 с — 0.964	600 с — 0.906	120 с — 0.812

Таблица В.7 — Усреднённые значения метрик AUC-ROC и F-меры для модуля ВТ на этапе финального тестирования для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	Logistic Regression
AUC-ROC	5	0.972	0.989	0.859	0.851
	10	0.961	0.984	0.902	0.881
	30	0.998	0.999	0.973	0.969
	60	0.989	0.997	0.979	0.963
	90	0.997	0.999	0.998	0.989
	120	0.996	0.997	0.996	0.979
	240	0.998	0.999	0.999	0.993
	600	0.994	0.997	0.996	0.971
	Лучший результат	30 с, 240 с — 0.998	30 с, 90 с, 240 с — 0.999	90 с — 0.998	240 с — 0.993
F-мера	5	0.896	0.938	0.786	0.764
	10	0.859	0.916	0.814	0.758
	30	0.923	0.953	0.924	0.892
	60	0.942	0.959	0.933	0.892
	90	0.946	0.968	0.936	0.904
	120	0.949	0.961	0.942	0.893
	240	0.925	0.965	0.987	0.927
	600	0.913	0.931	0.963	0.897
	Лучший результат	120 с — 0.949	90 с — 0.968	240 с — 0.987	240 с — 0.927

Таблица В.8 — Усреднённые значения метрик AUC-ROC и F-меры для модуля ВТ на этапе финального тестирования для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	Logistic Regression
AUC-ROC	5	0.974	0.989	0.866	0.849
	10	0.972	0.984	0.915	0.914
	30	0.999	0.999	0.969	0.935
	60	0.998	0.999	0.984	0.969
	90	0.999	0.999	0.992	0.985
	120	0.999	0.999	0.991	0.995
	240	0.999	0.999	0.968	0.967
	600	0.999	0.999	0.967	0.961
	Лучший результат	240 с — 1.0	30 с, 60 с, 90 с, 120 с, 240 с, 600 с — 1.0	120 с — 0.991	120 с — 0.995
F-мера	5	0.899	0.932	0.793	0.759
	10	0.895	0.919	0.849	0.809
	30	0.934	0.964	0.906	0.864
	60	0.951	0.973	0.913	0.887
	90	0.926	0.956	0.932	0.899
	120	0.921	0.929	0.933	0.929
	240	0.935	0.943	0.919	0.935
	600	0.935	0.964	0.931	0.908
	Лучший результат	60 с — 0.951	60 с — 0.973	120 с — 0.933	240 с — 0.935

Таблица B.9 — Усреднённые значения метрик AUC-ROC и F-меры для модуля WIFI на этапе финального тестирования для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForestClassifier	SVM-SVC	Logistic Regression
AUC-ROC	5	0.999	0.997	0.996	0.946
	10	0.998	0.999	0.993	0.989
	30	0.999	0.999	0.996	0.984
	60	0.998	0.999	0.997	0.953
	90	0.999	0.999	0.996	0.973
	120	0.999	0.999	0.999	0.988
	240	0.996	0.998	0.998	0.979
	600	0.998	0.999	0.998	0.992
	Лучший результат	30 с, 90 с, 120 с — 0.999	10 с, 30 с, 60 с, 90 с, 120 с — 0.999	120 с — 0.999	600 с — 0.992
F-мера	5	0.952	0.968	0.979	0.925
	10	0.933	0.936	0.929	0.935
	30	0.949	0.992	0.924	0.922
	60	0.952	0.978	0.946	0.899
	90	0.932	0.962	0.963	0.906
	120	0.956	0.978	0.965	0.926
	240	0.964	0.962	0.964	0.919
	600	0.899	0.967	0.973	0.929
	Лучший результат	240 с — 0.964	30 с — 0.992	5 с — 0.979	10 с — 0.935

Таблица B.10 — Усреднённые значения метрик AUC-ROC и F-меры для модуля WIFI на этапе финального тестирования для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	Logistic Regression
AUC-ROC	5	0.982	0.996	0.995	0.953
	10	0.984	0.998	0.993	0.951
	30	0.991	0.997	0.998	0.972
	60	0.996	0.998	0.999	0.968
	90	0.999	0.999	0.999	0.988
	120	0.999	0.999	0.999	0.973
	240	0.999	0.999	0.998	0.969
	600	0.999	0.999	0.999	0.979
	Лучший результат	90 с, 120 с, 240 с, 600 с — 0.999	90 с, 120 с, 240 с, 600 с — 0.999	60 с, 90 с, 120 с, 600 с — 0.999	90 с — 0.988
F-мера	5	0.921	0.943	0.944	0.854
	10	0.934	0.921	0.946	0.852
	30	0.932	0.956	0.954	0.921
	60	0.954	0.969	0.981	0.932
	90	0.958	0.971	0.972	0.926
	120	0.957	0.985	0.976	0.927
	240	0.962	0.986	0.974	0.931
	600	0.927	0.959	0.953	0.895
	Лучший результат	240 с — 0.962	240 с — 0.986	60 с — 0.981	60 с — 0.932

Таблица В.11 — Усреднённые значения метрик AUC-ROC и F-меры для модуля LOCATION на этапе финального тестирования для стационарного окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	Logistic Regression
AUC-ROC	5	0.999	0.999	0.989	0.937
	10	0.998	0.997	0.987	0.933
	30	0.998	0.998	0.992	0.942
	60	0.999	0.998	0.994	0.939
	90	0.999	0.999	0.998	0.956
	120	0.999	0.999	0.993	0.967
	240	0.999	0.999	0.997	0.971
	600	0.999	0.999	0.995	0.969
	Лучший результат	60 с, 90 с, 120 с, 240 с, 600 с — 0.999	90 с, 120 с, 240 с, 600 с — 0.999	90 с — 0.998	600 с — 0.969
F-мера	5	0.945	0.976	0.941	0.856
	10	0.954	0.981	0.954	0.867
	30	0.937	0.984	0.967	0.845
	60	0.961	0.991	0.959	0.864
	90	0.965	0.986	0.976	0.872
	120	0.972	0.979	0.981	0.874
	240	0.971	0.993	0.985	0.882
	600	0.969	0.989	0.984	0.875
	Лучший результат	120 с — 0.972	240 с — 0.993	240 с — 0.985	240 с — 0.882

Таблица B.12 — Усреднённые значения метрик AUC-ROC и F-меры для модуля LOCATION на этапе финального тестирования для скользящего окна

Метрика	Размер окна, с	CatBoostClassifier	RandomForest	SVM-SVC	Logistic Regression
AUC-ROC	5	0.999	0.999	0.964	0.965
	10	0.997	0.998	0.971	0.962
	30	0.998	0.999	0.946	0.967
	60	0.999	0.999	0.953	0.937
	90	0.999	0.999	0.974	0.963
	120	0.999	0.999	0.973	0.971
	240	0.999	0.999	0.962	0.968
	600	0.999	0.999	0.971	0.967
	Лучший результат	60 с, 90 с, 120 с, 240 с, 600 с — 0.999	5 с, 30 с, 60 с, 90 с, 120 с, 240 с, 600 с — 0.999	90 с — 0.974	240 с — 0.968
F-мера	5	0.941	0.968	0.929	0.927
	10	0.939	0.953	0.939	0.887
	30	0.956	0.954	0.937	0.911
	60	0.942	0.975	0.929	0.879
	90	0.941	0.965	0.936	0.895
	120	0.962	0.978	0.941	0.907
	240	0.971	0.984	0.954	0.911
	600	0.969	0.973	0.962	0.909
	Лучший результат	240 с — 0.971	240 с — 0.984	600 с — 0.962	30 с — 0.911