НАУЧНО-ИЗДАТЕЛЬСКИЙ ЦЕНТР «АКАДЕМИЯ ЕСТЕСТВОЗНАНИЯ»

SCIENTIFIC PUBLISHING CENTER «ACADEMY OF NATURAL HISTORY»

НАУЧНОЕ ОБОЗРЕНИЕ • ТЕХНИЧЕСКИЕ НАУКИ № 2

SCIENTIFIC REVIEW • TECHNICAL SCIENCES 2022

Журнал «Научное обозрение. Технические науки» зарегистрирован Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство ПИ № ФС77-57440
ISSN 2500-0799

Двухлетний импакт-фактор РИНЦ – 0,270
Пятилетний импакт-фактор РИНЦ – 0,242

Учредитель, издательство и редакция: ООО НИЦ «Академия Естествознания», Почтовый адрес: 105037, г. Москва, а/я 47 Адрес редакции и издателя: 410056, Саратовская область, г. Саратов, ул. им. Чапаева В.И., д. 56

Founder, publisher and edition: LLC SPC Academy of Natural History, Post address: 105037, Moscow, p.o. box 47 Editorial and publisher address: 410056, Saratov region, Saratov, V.I. Chapaev Street, 56

Подписано в печать 29.04.2022
Дата выхода номера 31.05.2022
Формат 60×90 1/8

Типография ООО НИЦ «Академия Естествознания», 410035, Саратовская область, г. Саратов, ул. Мамонтовой, д. 5

Signed in print 29.04.2022
Release date 31.05.2022
Format 60×90 8.1

Typography LLC SPC «Academy Of Natural History» 410035, Russia, Saratov region, Saratov, 5 Mamontovoi str.

Технический редактор Доронкина Е.Н. Корректор Галенкина Е.С., Дудкина Н.А.

Тираж 1000 экз.
Распространение по свободной цене
Заказ НО 2022/2

Журнал «НАУЧНОЕ ОБОЗРЕНИЕ» выходил с 1894 по 1903 год в издательстве П.П. Сойкина. Главным редактором журнала был Михаил Михайлович Филиппов. В журнале публиковались работы Ленина, Плеханова, Циолковского, Менделеева, Бехтерева, Лесгафта и др.

Journal «Scientific Review» published from 1894 to 1903. P.P. Soykin was the publisher. Mikhail Filippov was the Editor in Chief. The journal published works of Lenin, Plekhanov, Tsiolkovsky, Mendeleev, Bekhterev, Lesgaft etc.

М.М. Филиппов (M.M. Philippov)

С 2014 года издание журнала возобновлено Академией Естествознания

From 2014 edition of the journal resumed by Academy of Natural History

НАУЧНОЕ ОБОЗРЕНИЕ • ТЕХНИЧЕСКИЕ НАУКИ

SCIENTIFIC REVIEW • TECHNICAL SCIENCES

www.science-education.ru

2022 г.

В журнале представлены научные обзоры, статьи проблемного и научно-практического характера

The issue contains scientific reviews, problem and practical scientific articles

СОДЕРЖАНИЕ

CONTENTS

СТАТЬИ

УДК 681.5

АНАЛИЗ СХЕМЫ СИМУЛЯЦИИ ЁМКОСТНОГО ДАТЧИКА, ФИКСИРУЮЩЕГО СУЩЕСТВОВАНИЕ ОБЪЕКТА

Агаева Ф.Ш.

Сумгаитский государственный университет, Сумгаит, e-mail: agayeva.feride71@mail.ru

В статье анализируется ёмкостный датчик, фиксирующий существование объекта. Рассматривается функциональная схема и соответствующая схема графа датчика, а также схема симуляции гиратора, построенная с помощью программы Electronics Workbench. Емкостные датчики, фиксирующие существование объекта, могут использоваться для контроля динамики движения (заполнения и выгрузки) жидкостей и гранул в цистернах, в качестве счетчика при подсчете готовой продукции, в качестве контрольного устройства для предупреждения транспортных средств о препятствиях, в качестве чувствительного элемента в весовых системах. Как видно, в связи с развитием информационных технологий применение датчиков в измерительной технике актуально и спрос на них растет. В этом смысле актуально создание различных автоматизированных систем управления на основе емкостных преобразователей. Преимуществом емкостных датчиков является малая инерционность и высокая чувствительность. Недостатком является то, что он чувствителен к внешним электромагнитным полям. В результате можно сказать, что емкостные датчики больших перемещений обладают высокой чувствительностью и нелинейными характеристиками преобразования. Применение структурно-алгоритмического метода для обеспечения точности измерений таких датчиков считается более эффективным.

Ключевые слова: емкостный датчик, чувствительный элемент, функциональная схема, усилитель, преобразователь

ANALYSIS OF THE SCHEME OF SIMULATION OF A CAPACITIVE SENSOR FIXING THE EXISTENCE OF AN OBJECT

Agaeva F.Sh.

Sumgait State University, Sumgait, e-mail: agayeva.feride71@mail.ru

The article analyzes a capacitive sensor that detects the existence of an object. The functional diagram and the corresponding diagram of the sensor graph are considered. Also a gyrator simulation circuit built using the Electronics Workbench program. Capacitive sensors fixing the existence of an object can be used to control the dynamics of movement (filling and unloading) of liquids and granules in tanks, as a counter when counting finished products, as a control device for warning vehicles about obstacles, as a sensitive element in weighing systems. As you can see, in connection with the development of information technology, the use of sensors in measuring technology is relevant and the demand for them is growing. In this sense, the creation of various automated control systems based on capacitive transducers is a topical issue. As a result, we can say that large displacement capacitive sensors have high sensitivity and non-linear conversion characteristics. The use of a structural-algorithmic method to ensure the accuracy of measurements of such sensors is considered to be more effective.

Keywords: capacitive sensor, sensitive element, functional diagram, amplifier, converter

Как видно, с течением времени меняется мышление людей, создаются новые подходы, новые технологии. Наибольшее внимание уделяется обеспечению жизнедеятельности человека, комфортного образа жизни и безопасности. Безопасность жизни людей в авиарейсах, на кораблях, в общественных, жилых и служебных зданиях, во всех областях промышленности в целом следует считать высшим приоритетом. Для этого нужно создавать более надежные системы безопасности.

Освещение является неотъемлемой частью любой системы электроснабжения, будь то промышленные предприятия, офисы или жилые здания. Для снижения расхода электроэнергии, вызванного работой осветительных приборов во время отсутствия в помещении людей [1], а также для надежной обороны и защиты людям нужны более совершенные системы. Для этого существует большая потребность в создании современных сенсорных устройств. Таким образом, одним из важных устройств являются различные датчики присутствия и движения, ёмкостные датчики, фиксирующие существование объекта. Выходной сигнал датчика, указывающий на присутствие объекта, не зависит от того, движется ли объект или покоится [2]. Такие датчики широко используются во всех типах систем защиты, управления энергопотреблением и даже в интерактивных игрушках. В современных производственных, коммерческих и жилых зданиях установка датчиков присутствия как средства для управления искусственным освещением позволяет не только повысить комфортность при эксплуатации помещений, но и существенно снижает расходы электроэнергии, что в конечном итоге делает установку датчиков экономически выгодной [3]. Также существует бесконтактный емкостный датчик, который чувствует приближение объекта или измеряет расстояние приближения.

Цель исследования:

1. Анализировать схему емкостного датчика, измеряющего расстояние приближения.

2. Анализировать схему симуляции гиратора, построенного с помощью программы Electronics Workbench.

3. Составить соответствующую схему графа.

4. Рассмотреть принцип работы простого емкостного датчика, который фиксирует существование объекта.

5. Моделировать с помощью программы Electronics Workbench.

6. На основе конденсатора составить электрическую схему.

Материалы и методы исследования

Бесконтактный емкостный датчик, который выполнен из двух металлических электродов и описывается как «открытая» оболочка конденсатора с активной поверхностью. Электроды подключены к цепи обратной связи высокочастотного автоматического генератора, а автоматический генератор сконфигурирован таким образом, чтобы не генерировать сигнал при отсутствии объекта вблизи активной поверхности. Когда объект приближается к активной поверхности бесконтактного конденсатора, он попадает в электрическое поле, и емкость конденсатора изменяется по контуру обратной связи. Генератор начинает воспринимать эти эффекты, и по мере приближения объекта амплитуда сигнала воздействия увеличивается, создавая выходной сигнал.

Бесконтактные конденсаторы срабатывают при попадании в зону воздействия как токопроводящих предметов, так и диэлектриков. При воздействии предметов из электропроводящих материалов реальная дистанция L включения приобретает максимальное значение, а при воздействии предметов из диэлектрических материалов дистанция L_p уменьшается в зависимости от диэлектрической проницаемости ε_r материала.

Эти зависимости отражены на рис. 1.

При исследовании емкости график этой функциональной зависимости используют при выборе объектов из материалов с различной диэлектрической проницаемостью. Номинальные расстояния $L_н$ пуска и отключения L_o определяются из технических характеристик. Гарантированные диапазоны значений воздействия передатчика устанавливаются относительно заземленного металлического объекта (L_p = 100 %). Определен интервал соотношения для определения реального расстояния включения (L_p): $0{,}9L_н < L_p < 1{,}1\ L_н$.

Теперь рассмотрим принцип построения функциональной структуры датчика емкостного преобразователя. Функциональная схема датчика с емкостным преобразователем представлена на рис. 2.

В качестве чувствительного элемента в схеме используется резонансный контурный усилитель. Когда частота чувствительного элемента изменена и равна резонансной частоте, $f_k = f_{rez}$ сопротивление чувствительного элемента достигает своего максимального значения и, соответственно, выходное напряжение достигает высокого уровня.

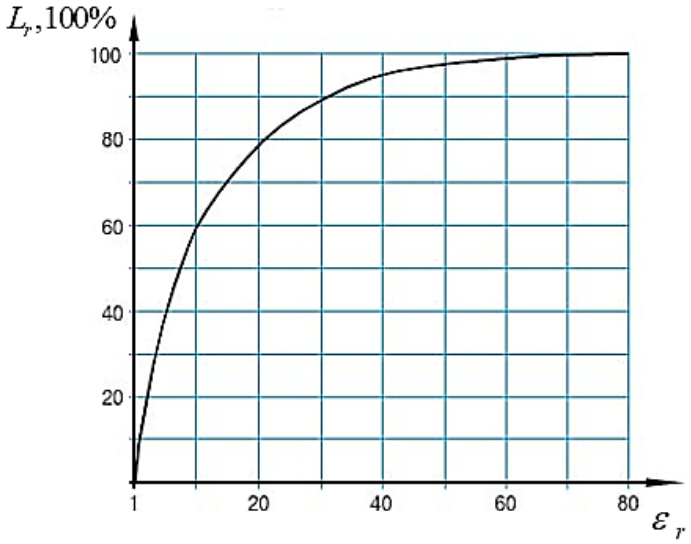

Рис. 1. График зависимости диэлектрической проницаемости материала объекта от фактического расстояния сближения

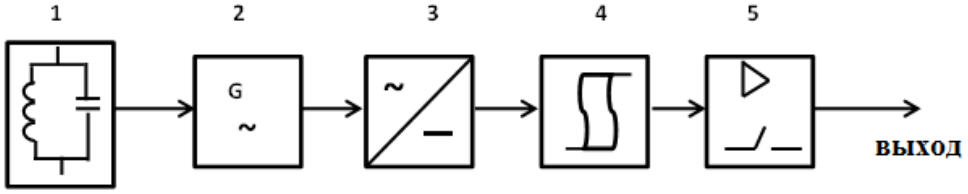

Рис. 2. Функциональная схема датчика с емкостным преобразователем больших перемещений: 1 – чувствительный элемент – датчик (резонансный контурный усилитель); 2 – генератор синусоидальных колебаний LC; 3 – демодулятор; 4 – компаратор; 5 – выходной переключатель усилителя

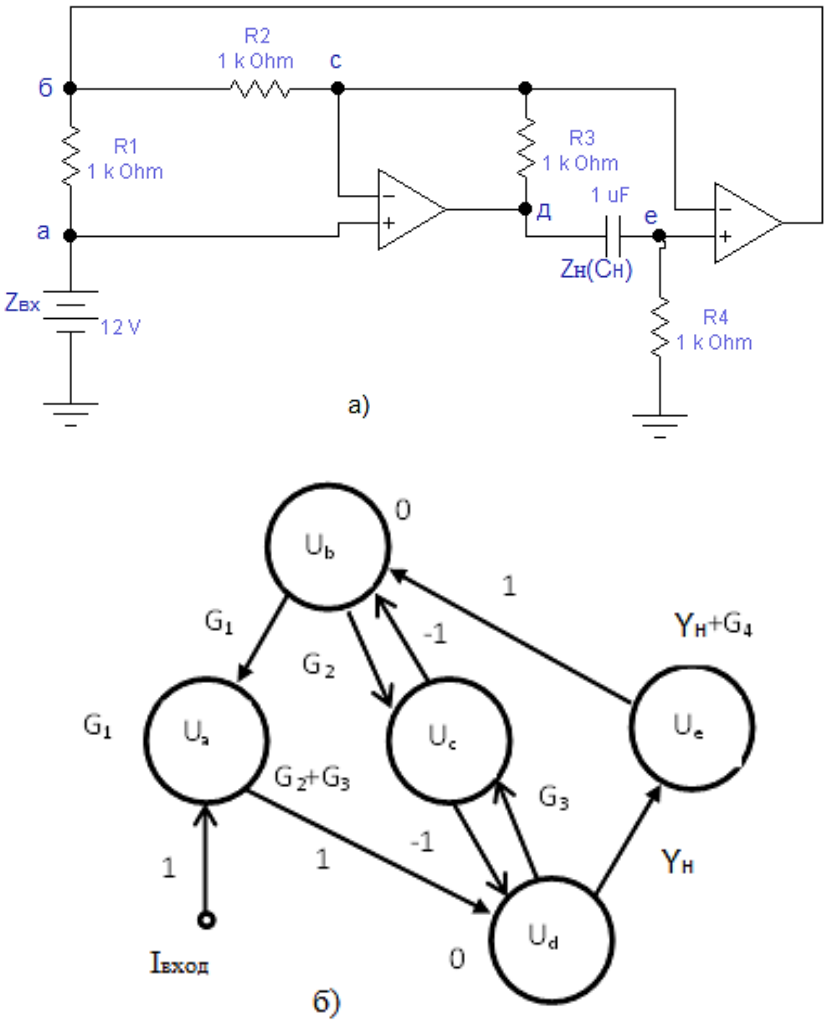

Рис. 3. а) схема гиратора, б) схема графа

Вместо индуктивности в резонансном контуре используется гиратор, а входное сопротивление гиратора обратно пропорционально сопротивлению нагрузки и выражается следующим образом:

$$Z_{вход} = a / Z_{наг}, \quad (1)$$

где a – коэффициент измерения, постоянный.

Если мы используем конденсатор в качестве сопротивления нагрузки генератора, мы можем написать следующее выражение:

$$Z_{наг} = 1 / (j\omega C_{наг}). \quad (2)$$

Таким образом, получаем индуктивное входное сопротивление:

$$Z_{вход} = j\omega a C_{наг}. \quad (3)$$

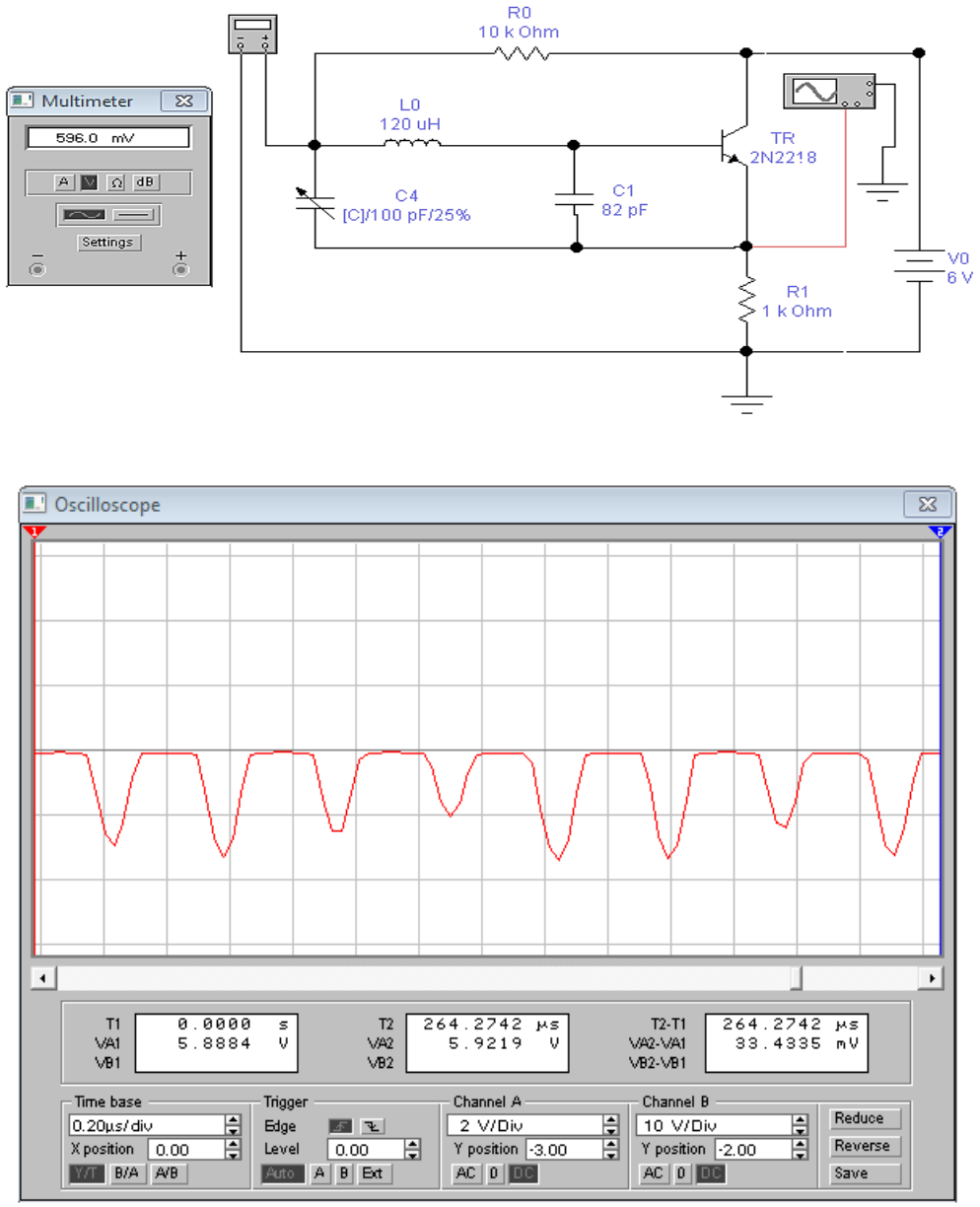

Рис. 4. Схема симуляции ёмкостного датчика, фиксирующего существование объекта

На рис. 3, а, представлена схема симуляции гиратора, построенная с помощью программы Electronics Workbench, а соответствующая схема графа – на рис. 3, б.

Эта схема подходит для диапазона низких частот и позволяет создать резонансный контур с высокой добротностью. Используемый диапазон частот зависит от параметров выбранного операционного усилителя (ОУ).

Рассмотрим принцип работы простого емкостного датчика, который фиксирует существование объекта. Простой пример на основе конденсатора, визуально указывающего на наличие объекта, моделируется с помощью программы Electronics Workbench, а электрическая схема показана на рис. 4.

Схема основана на высокочастотном генераторе с внешним конденсатором и высокоомным выходным транзистором. В результате внешней нагрузки изменяется ток, протекающий по каналу транзистора, и этот чувствительный элемент использует конденсатор *C1*. В качестве объекта может быть использован любой предмет с металлическим содержанием, например металлическая пластина, рамка, кусок проволоки и т.п. на полу или на стене. В рассматрива-

емой нами схеме конденсатор С4 заменен на предмет, а выходные сигналы, полученные при изменении этого конденсатора в диапазоне от 10 пФ до 50 пФ – характеристика, контролируются осциллографом.

Емкостные датчики, фиксирующие существование объекта, могут использоваться для контроля динамики движения (заполнения и выгрузки) жидкостей и гранул в цистернах, в качестве счетчика при подсчете готовой продукции, в качестве контрольного устройства для предупреждения транспортных средств о препятствиях, при парковке автомобилей, в качестве чувствительного элемента в весовых системах.

Многие автолюбители сталкиваются с определенными трудностями во время парковки. Габариты автомобиля мешают водителю точно оценить расстояние до объектов, расположенных в неудобных местах. Парковочный радар, который состоит из датчиков, оповещает водителя с помощью звукового сигнала. Так как каждый из датчиков сначала излучает ультразвуковой сигнал, затем переключается на приемник для захвата сигнала, отраженного от препятствия. Электронный блок измеряет, как долго длится сигнал, вычисляет расстояние до ближайшего объекта и предупреждает водителя звуковым сигналом, индикатором расстояния [4].

Использование емкостных датчиков в качестве элемента антигравитационных систем автомобиля позволяет установить предельную нагрузку автомобиля. Известно, что для перевозки неметаллических материалов используются различные типы грузовых автомобилей [5]. Вес груза важен при транспортировке грузов на дальние расстояния. Датчики, установленные в грузоподъемной части грузовиков, позволяют контролировать нагрузку на протяжении всего пути следования. Эти датчики также используются для погрузки и разгрузки грузов, подлежащих транспортировке в транспортное средство [6].

Как видно, в связи с развитием информационных технологий применение датчиков в измерительной технике актуально и спрос на них растет. В этом смысле актуально создание различных автоматизированных систем управления на основе емкостных преобразователей.

Преимуществом емкостных датчиков является малая инерционность и высокая чувствительность. Недостатком является то, что они чувствительны к внешним электромагнитным полям.

Существует два основных пути повышения точности измерительных приборов:

1. Некоторые инструменты используются для повышения точности и стабильности измерительных приборов. Однако у этого подхода есть и другая особенность, заключающаяся в том, что компоненты измерительного прибора не добавляются в структуру инструмента. К средствам измерений относятся только те блоки и узлы, которые необходимы для выполнения измерительных операций. По этой причине возможности такого подхода сильно сокращаются.

2. Их совместное применение приводит к повышению точности средств измерений, улучшению метрологических характеристик отдельных узлов средств измерений в результате выполнения дополнительных операций измерений и обработки их результатов по определенным алгоритмам.

Второй подход более эффективен, поскольку исключает в первую очередь необходимость доработки исходного высокоточного средства измерения, а результат измерений многократно повышается за счет обеспечения процесса измерений эталонами, их оптимальными значениями и сочетаниями.

Емкостные датчики обладают высокой чувствительностью и нелинейными характеристиками преобразования. Применение структурно-алгоритмического метода для обеспечения точности измерений таких датчиков считается более эффективным.

Список литературы

1. Павлов Д.Д., Зимин А.Д. Разработка датчика присутствия // Проектирование и технология электронных средств. 2012. № 1. С. 36–40.

2. Чичинов С.А. Выбор датчика для определения координат движущегося тела в цилиндровом корпусе // Вестник молодежной науки России. 2019. № 2. С. 21.

3. Игнатьев А.С. Применение датчиков движения и присутствия и экономический эффект от их применения // Вестник научных конференций. 2019. № 5–1 (45). С. 64–65.

4. Ермаков В.В. Система обнаружения препятствий при парковке автомобиля // Ceteris Paribus. 2015. № 5. С. 20–24.

5. Advicelawyer.ru [Электронный ресурс]. URL: https://advicelawyer.ru/avtomobil/peregruz-avtomobilya.html (дата обращения: 08.04.2022).

6. Control-Auto: мониторинг транспорта [Электронный ресурс]. URL: http://www.control-auto.ru/datchiki_osevoy_ves.html (дата обращения: 08.04.2022).

УДК 004.384

АРХИТЕКТУРА СИСТЕМЫ «УПРАВЛЕНИЯ УМНЫМ ДОМОМ»

Засыпкин Д.С., Белов Ю.С.

Московский государственный технический университет имени Н.Э. Баумана, филиал г. Калуга, Калуга, e-mail: fn1-kf@mail.ru

Технологии «умного дома» развиваются быстрыми темпами, и «умный дом» сегодня – это лишь малая часть того, чем может стать «умный дом». Цель «умного дома» – облегчить повседневную жизнь его жителей за счет повышения комфорта, безопасности и эффективности. По мере повышения точности распознавания речи голос становится все более популярным средством контроля в «умном доме». И распознавание речи, и технология «умного дома» были перечислены как важные развивающиеся технологии в течение нескольких последних лет с ожидаемым высоким ростом рынка. Интеграция мобильных устройств в сегмент домашней автоматизации является важной задачей на данный момент. При этом пользователю необходимо предоставить возможность интуитивно взаимодействовать с системой. Данная интеграция стремится выйти за рамки обычных мобильных приложений домашней автоматизации при взаимодействии с пользователем, а также необходима концепция распознавания объектов с обнаружением в реальном времени. В данной статье описана архитектура системы управления «умным домом». Она состоит из модулей: мобильного модуля, домашнего модуля, конечного пользователя, интеллектуальных устройств. В разработанной системе пользователь может управлять устройствами, существующими в его доме, с помощью пользовательского интерфейса камеры, имеющего визуальное представление действий и обеспечивающего интуитивное взаимодействие. Мобильное приложение реализовано на платформе Android.

Ключевые слова: «умный дом», мобильное устройство, домашний модуль

THE ARCHITECTURE OF THE SMART HOUSE CONTROL SYSTEM

Zasypkin D.S., Belov Yu.S.

Bauman Moscow state technical University, Kaluga branch, Kaluga, e-mail: fn1-kf@mail.ru

Smart home technologies are developing rapidly, and a smart home today is only a small part of what a smart home can become. The purpose of a smart home is to facilitate the daily life of its residents by increasing comfort, safety and efficiency. As the accuracy of speech recognition increases, voice is becoming an increasingly popular means of control in the smart home. Both speech recognition and smart home technology have been listed as important emerging technologies over the past few years with high market growth expected. Integration of mobile devices into the home automation segment is an important task at the moment. At the same time, the user must be given the opportunity to interact intuitively with the system. This integration aims to go beyond the usual mobile home automation applications when interacting with the user, and also requires the concept of object recognition with real-time detection. This article describes the architecture of the smart home management system. It consists of modules: mobile module, home module, end user, smart devices. In the developed system, the user can control the devices existing in his house using the camera user interface, which has a visual representation of actions and provides intuitive interaction. The mobile application is implemented on the Android platform.

Keywords: smart home, mobile device, home module

В области домашней автоматизации уже ведется большая работа – от энергоэффективных до полностью автоматизированных решений. Однако внедрение машинного обучения во взаимодействие пользователя с системой находится на начальном этапе, поэтому необходимо провести огромную работу по внедрению машинного обучения в автоматизированную систему.

Поэтому предлагаемая система состоит из последовательности модулей:

− мобильный модуль;
− домашний модуль;
− конечный пользователь;
− интеллектуальные устройства.

На рисунке 1 представлена архитектура высокоуровневой системы, в которой можно идентифицировать представление пользователя, мобильный модуль, модуль платформы автоматизации и интеллектуальные устройства, подключенные внутри дома.

Цель исследования: рассмотреть архитектуру и особенности системы управления «умным домом».

Мобильный модуль. Этот модуль служит точкой подключения пользователя ко всему рабочему процессу. Все взаимодействия будут происходить в приложении, установленном на мобильном устройстве, и действия, предпринятые внутри него, вызовут серию событий во всей системе.

Как показано на рисунке 2, мобильный компонент состоит из мобильного приложения и переобученной модели, интегрированной в приложение. Эта интеграция происходит в режиме реального времени, обеспечивая мгновенную обратную связь с пользователем, и поэтому, согласно сценарию, конкретные команды и действия будут отправляться по каналу связи, существующему между этим модулем и модулем домашней автоматизации. Оба компонен-

та играют важную роль в быстродействии и полезности системы и описаны в следующих подразделах.

Мобильное приложение. Мобильное приложение будет установлено на пользовательском устройстве, и его основная роль заключается в сборе пользовательских взаимодействий, интерпретации собранных данных и установлении связи с платформой автоматизации.

С учетом простоты использования и доступности наиболее разумным выбором в данном контексте является приложение для Android. Разработка собственного приложения для iOS потребует специального оборудования и охватит меньшее сообщество пользователей, чем Android. Таким образом, при выборе операционной системы Android для разработок можно гарантировать, что большое количество устройств будет совместимо, можно будет охватить огромное количество пользователей, а процесс общения не будет ограничен возможностями программного обеспечения.

Кроме того, это приложение должно обеспечивать возможность установления связи между мобильным устройством и платформой домашней автоматизации. Для этого ему необходимо будет играть роль клиента в протоколе связи между ними, отправляя информацию на платформу. Поскольку Android поддерживает широкий спектр интеграций с коммуникационными плагинами и программным обеспечением, этот выбор гарантировал, что можно найти решение, близкое к оптимальному.

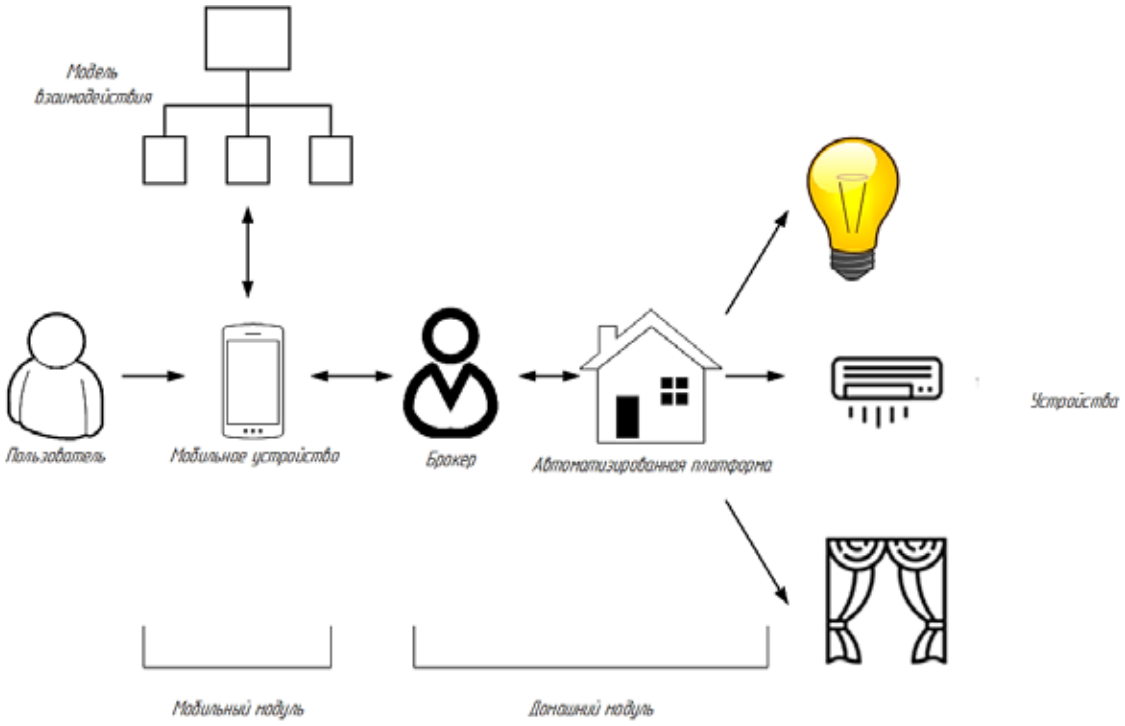

Рис. 1. Архитектура высокоуровневой системы

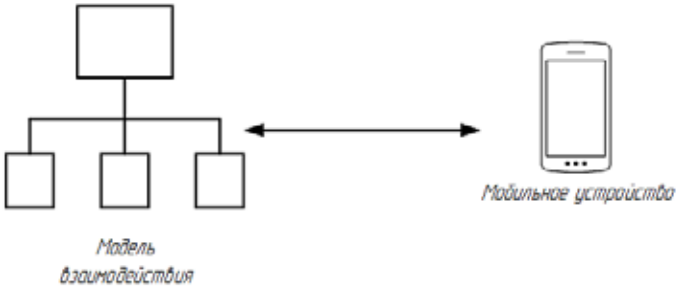

Рис. 2. Архитектура мобильного модуля

Датчики мобильного устройства. Основной целью предлагаемой интеграции является возможность использования датчиков мобильных устройств для интуитивного взаимодействия с интеллектуальными устройствами. Одним из основных и крупнейших сборщиков информации, который можно найти в каждом мобильном устройстве, является камера. Таким образом, чтобы воспользоваться всеми возможностями, необходимо найти способ интерпретации данных, собранных датчиком камеры.

Решением стало использование библиотеки машинного обучения, совместимой с предложенной архитектурой, способной загружать и обрабатывать данные, создавать, обучать и повторно использовать модели с простым развертыванием. Таким образом, решение, используемое в компоненте интерпретации, опирается на структуру тензорного потока [1] .

В дополнение к предоставлению возможности переобучения нейронной сети без использования сложного и мощного оборудования самым большим общим преимуществом этого инструмента является возможность запускать модели машинного обучения на мобильных устройствах в рамках сжатого и мобильного решения Tensor Flow Lite [2].

Использование машинного обучения на устройстве позволяет упростить архитектуру системы без необходимости выполнения последовательных вызовов сервера для оценки информации.

На рисунке 3 представлена архитектура решения – стек компонентов, участвующих в процессе. Кроме того, программные интерфейсы приложений Java и C++ отвечают за загрузку и вызов интерпретатора, который, в свою очередь, выполняет модель с использованием набора ядер. В версиях Android, превосходящих 8.1, поддерживается API нейронных сетей Android (NN API) [3], который позволяет эффективно распределять вычисления между процессорами устройств и использовать преимущества аппаратного ускорения с помощью аппаратного уровня абстракции нейронных сетей Android (NN HAL). Если ни один из них не доступен, будет задействован центральный процессор.

Домашний модуль. В нем находятся два основных компонента: брокер, отвечающий за связь модулей, и экземпляр домашней автоматизации, отвечающий за интеграцию интеллектуальных устройств и датчиков. Оба они размещены внутри одной и той же установочной платформы.

Одной из основных проблем, которые возникали, является установка брокера и платформы автоматизации на одно и то же оборудование, чтобы можно было избежать необходимости в дополнительном оборудовании. Следовательно, выбор всех трех компонентов был сделан с учетом необходимости того, чтобы каждый из них был совместим между собой.

Подходящим решением для интеграции обоих элементов является установка их внутри Raspberry Pi. Таким способом обеспечиваются снижение стоимости, простота установки и обслуживания, так как весь этот модуль основан на Raspberry Pi 3B + [4], где работают брокер MQTT и платформа автоматизации.

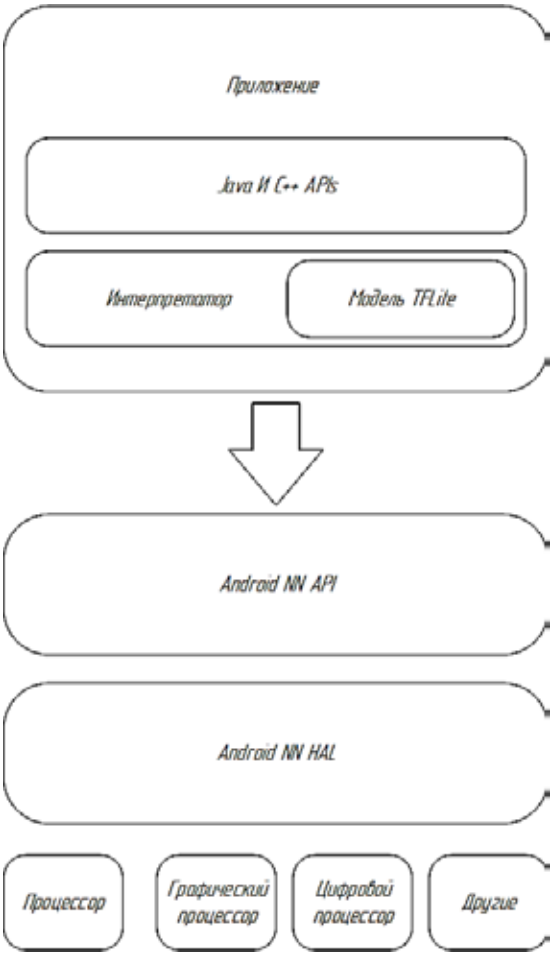

Рис. 3. Архитектура стека модели вывода

Брокер. Выбор MQTT [5] в качестве протокола связи привел к необходимости наличия брокера для управления всеми вызовами публикации и подписки и, следовательно, поддержания связи между задействованными элементами. Решением MQTT Broker можно управлять тремя различными способами: первый предполагает наличие публичного брокера, предоставляющего ус-

луги, второй – частного брокера, а третий подразумевает использование встроенного брокера. Поскольку в публичном брокере любое устройство или организация могут публиковать и подписываться на любую тему на нем и у большинства платформ домашней автоматизации по умолчанию отсутствует брокер, самый безопасный и простой способ интегрировать этот узел в архитектуру – полагаться на частного брокера, где только устройства с предоставленным разрешением могут публиковать и подписываться на темы, управляемые этим брокером.

Для выполнения этой интеграции выбранным брокером MQTT был Mosquitto [6]. Как легкое решение с открытым исходным кодом, широко используемое для обмена сообщениями Интернета вещей и, кроме того, легко устанавливаемое на Raspberry Pi, Mosquitto предоставляет инструменты, необходимые для работы в качестве посредника связи.

На рисунке 4 представлено размещение брокера в архитектуре системы, принимающего запросы на подписку от платформы автоматизации и, соответственно, доставляющего туда сообщения при получении публикации данных с мобильного устройства, следовательно, устанавливающего соединение для передачи данных между ними.

Платформа автоматизации. Принимая во внимание предполагаемую интеграцию и учитывая выводы, сделанные ранее, можно заключить, что Home Assistant [7] является хорошим выбором для реализации, поскольку он проверяет все флажки: имеет открытый исходный код, разработан на хорошо известном легком в освоении языке, имеет сильное базовое сообщество пользователей и может быть установлен на устройстве с низкой вычислительной мощностью, таком как Raspberry Pi.

Установка состоит в экземпляре Home Assistant, размещенном в Raspberry Pi, изначально работающем в виртуальной среде Python поверх ОС Raspian. Как показано на рисунке 5, платформа также будет иметь подключенные к себе устройства «умного дома», способные отправлять команды, выполнять операции и изменять состояние в соответствии с информацией, отправленной пользователем с мобильного устройства и переданной через брокера. Таким образом, при определении желаемого решения и учитывая, что уже существовало, а также ограничения и возможности предлагаемой интеграции, процесс проектирования и разработки можно разделить на ряд этапов:

− разработка компонента машинного обучения и преобразование его в нужный формат;

− создание мобильного приложения, отвечающего за интеграцию модели в режиме реального времени;

− установка и настройка платформы автоматизации;

− внедрение коммуникационных процессов.

Эти четыре основных этапа разработки и внедрения системы являются ключом к успешной работе предлагаемого решения.

Связь модулей. MQTT представляется наиболее подходящим выбором в этом сценарии, предоставляя решение для обеспечения связи между двумя модулями, которое широко используется в средах Интернета вещей с низкой пропускной способностью, низкой задержкой и хорошей производительностью (рис. 6).

В контексте архитектуры системы и с учетом принципов MQTT один компонент будет выступать в качестве издателя, один – в качестве брокера, а третий – в качестве подписчика [8].

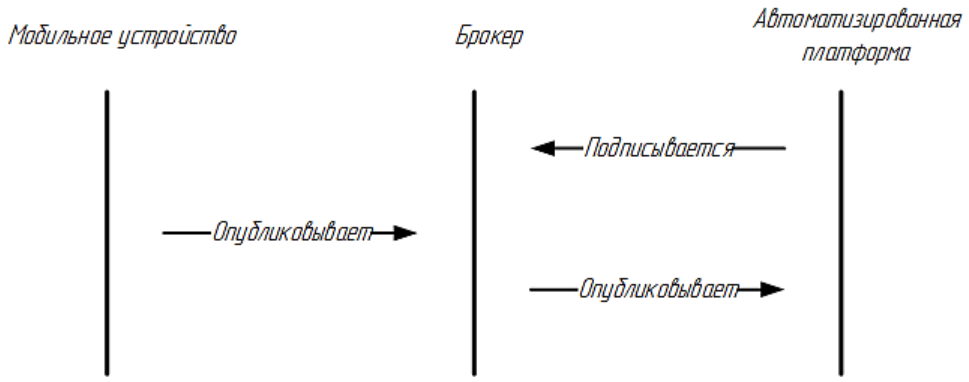

Рис. 4. Роль посредника в коммуникации в архитектуре системы

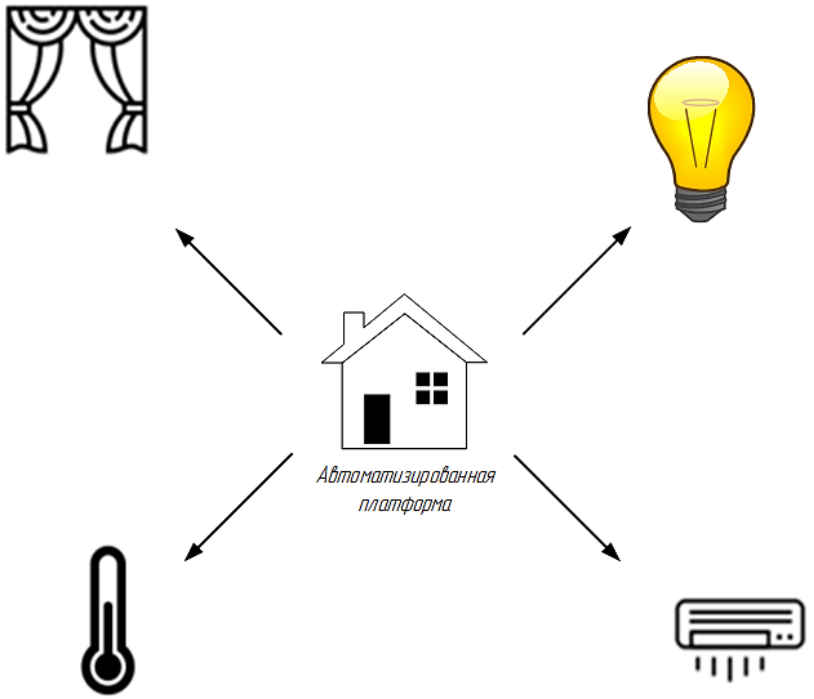

Рис. 5. Интеллектуальные устройства, подключенные к платформе домашней автоматизации

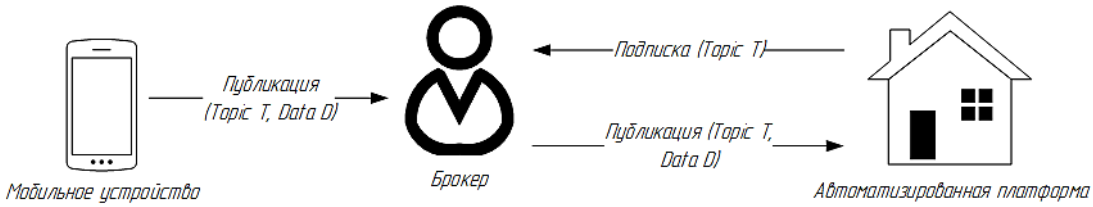

Рис. 6. Архитектура процесса коммуникации

Наиболее логичным способом реализации архитектуры протокола в этом сценарии является использование мобильного устройства абонентом, ответственным за публикацию данных в определенной теме, на которую подписана платформа автоматизации. Таким образом, мобильное устройство может постоянно публиковать обновления об изменениях состояния и взаимодействиях с пользователями, зная, что они будут получены на платформе автоматизации, прослушивающей данные в определенных подписанных темах.

В данной статье описана разработанная система. Она состоит из следующих модулей:

– мобильный модуль;
– домашний модуль;
– конечный пользователь;
– интеллектуальные устройства.

В разработанной системе пользователь может управлять устройствами, существующими в его доме, с помощью пользовательского интерфейса камеры, имеющего визуальное представление действий и обеспечивающего интуитивное взаимодействие.

Список литературы

1. Засыпкин Д.С., Белов Ю.С. Обзор алгоритмов распознавания лица человека в библиотеке OpenCV [Электронный ресурс]. URL: http://e-scio.ru/wp-content/uploads/2021/07/Засыпкин-Д.-С.-Белов-Ю.-С.pdf. (дата обращения: 15.04.2022).

2. Ислам К., Шен У., Ван Х. Соображения безопасности и конфиденциальности для беспроводных сенсорных сетей в среде «умного дома» // Материалы 16-й международной конференции IEEE 2012 года по совместной работе при поддержке компьютеров в области проектирования (CSCWD). 2012. С. 626-633.

3. Кучер М.Ю., Белов Ю.С. Подходы к распознаванию лиц и их методы // Технические и естественные науки: сбор-

ник избранных статей по материалам Международной научной конференции. СПб., 2020. С. 38-40.

4. Бодров С.А., Журавлёв А.В., Ерпелев А.В. Умный дом: история, принцип работы, устройства умного дома, протоколы // Технические науки: проблемы и решения: сборник статей по материалам XLIV международной научно-практической конференции. М., 2021. С. 29-32.

5. Васильчиков М.Г., Клименко А.В. Разработка вебсервера для удаленного управления системой «Умный дом» // Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях: материалы XXI Республиканской научной конференции студентов и аспирантов. Гомель, 2018. С. 83-84.

6. Zainal Al.N. The architecture of smart home internet of things. T-Comm. 2021. Т. 15. № 8. С. 58-61.

7. Шиков С.А., Алексеев Е.Г., Шестопалова А.Н. Gрименение нейронных сетей в системах умного дома // Материалы XXI научно-практической конференции молодых ученых, аспирантов и студентов Национального исследовательского Мордовского государственного университета им. Н.П. Огарёва. Саранск, 2017. С. 131-137.

8. Быков В.Э. Метод управления светом в умном доме на основе искусственных нейронных сетей // Международный журнал информационных технологий и энергоэффективности. 2021. Т. 6. № 2 (20). С. 3-12.

УДК 004.94

СОЗДАНИЕ МЕТОДИК ПРОГРАММНОЙ ВИЗУАЛИЗАЦИИ МОДЕЛЕЙ ТЕОРИИ ГРАФОВ

Ильичев В.Ю., Илюхин И.Ю.

Калужский филиал ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)», Калуга, e-mail: patrol8@yandex.ru

Теория графов является всё более развивающейся областью науки в связи с её применением в таких современных отраслях, как проектирование информационных (в частности, компьютерных) и прочих (например, коммуникационных) сетей. Графами описывается также соединение атомов в молекулы в химии, с их помощью изображаются социальные и экономические связи. При этом актуальной является проблема автоматизации процесса построения положения отдельных узлов и связей (рёбер) графов по заданным базам данных, представленным в популярных форматах, например Microsoft Excel. К настоящему времени благодаря своей универсальности и простоте освоения всё большую популярность приобретают языки программирования высокого уровня, развиваемые в основном энтузиастами. Так как данные языки используются для решения задач в совершенно разных отраслях науки, для них уже разработано большое количество модулей, каждый из которых содержит только функции, необходимые для определённой области. Наиболее известным и популярным свободно распространяемым языком программирования является язык Python. По данному языку существует не только огромное количество обучающих материалов, но также и множество сообществ в сети Интернет. В представленной работе подробно рассматривается применение модуля DeepGraph для визуализации сетевых графов по численным данным, приведённым в базах. Данный модуль применяется совместно с другими библиотеками функций Python. Созданы и рассматриваются три методики визуализации графов: двухмерные без построения рёбер, с рёбрами, а также изображения графа в трёхмерном виде (где показано изменение положения узлов графа с течением времени). Ко всем трём методикам приведены примеры изображения графов, построенных по одной и той же базе данных. В заключение даны рекомендации по дальнейшему использованию созданных методик и программных кодов, а также по популяризации применения языка Python.

Ключевые слова: теория графов, узел графа, ребро графа, язык Python, библиотека DeepGraph, модуль Numpy, модуль Matplotlib

CREATION OF SOFTWARE VISUALIZATION TECHNIQUES FOR GRAPH THEORY MODELS

Ilichev V.Yu., Ilukhin I.Yu.

Kaluga Branch of Bauman Moscow State Technical University, Kaluga, e-mail: patrol8@yandex.ru

Graph theory is increasingly emerging field of science due to application in such modern industries as design of information (in particular, computer) and other types (for example, communication) networks. Graphs also describe connection of atoms into molecules in chemistry, with their help they depict social and economic bonds. At same time, problem of automating process of constructing the position of links (edges) of graphs on given databases presented in popular formats, for example, Microsoft Excel, is relevant. To date, due to versatility and ease of development, high-level programming languages, developed mainly by enthusiasts, are becoming increasingly popular. Since these languages are used to solve problems in completely different branches of science, large number of modules have already been developed for them, each of which contains only functions necessary for particular area. The most famous and popular free programming language is Python. In this language, there are not only huge amount of educational materials, but also many communities on Internet. Present work discusses in detail use of DeepGraph module for visualizing network graphs from numerical data given in databases. This module is used in conjunction with other Python function libraries. Three graph visualization techniques have been created and are being considered: two-dimensional without edges, with edges, three-dimensional (which shows change in position of the nodes of graph over time). For all three methods, examples of graphs built on same database are given. In conclusion, recommendations are given on further use of created methods, on popularization of use of Python language.

Keywords: graph theory, graph node, graph edge, Python language, DeepGraph library, Numpy module, Matplotlib module

Теория графов занимается изучением специальных структур – абстрактного представления систем любой природы, объекты которых имеют попарные связи [1]. Граф является совокупностью двух математических множеств – множества исследуемых объектов, называемого множеством узлов (вершин), и множества связей вершин, называемого множеством рёбер [2]. Таким образом, получается, что каждое ребро является зависимым от положения вершин, а также часто может характеризоваться направлением от одной вершины к другой. В теории графов (называемой ещё теорией сетей) организация таких связей позволяет достичь параллелизации процесса прохож-

дения информации, ускоряя таким образом данный процесс [3]. Применение теории графов позволяет во многих случаях уменьшить загрузку узлов (а конкретно – снизить использование памяти и нагрузки процессоров в компьютерных сетях) [4]. В случае сильно разветвлённых сетей теория графов позволяет произвольно изменять прохождение информационных потоков через разные группы узлов, добиваясь наибольшей пропускной способности, предотвращения зашумления информации и достигая при этом наименьшей загрузки кромок.

Теория графов применяется во многих науках, но наибольшее употребление нашла в информатике и в разработке сетевых технологий. В настоящее время существует множество специальных программ и функций, позволяющих производить автоматизированную визуализацию схем, строящихся в теории графов [5]. Наиболее удобным для применения представляется современный модуль для среды программирования Python, позволяющий автоматически строить графы, который называется DeepGraph [6]. Он был разработан с использованием модуля научных вычислений Pandas [7] и даёт возможность разработчику реализовать вывод на схему графа как узлов сети (DataFrames), так и изображения связывающих их кромок. Также имеется возможность организации так называемых многослойных сетей (в которых, например, рассматривается изменение положения узлов с течением времени или трёхмерная архитектура сетей) [8].

Модуль DeepGraph является достаточно «продвинутым», но при этом и специфичным средством изображения и исследования особенностей организации графов. Хотя он не может создать конкуренцию таким «тяжёлым» средствам разработки локальных сетей для Python, наиболее известным из которых является NetworkX [9], его задачей является расширение функциональности данной среды программирования путём применения наиболее наглядных (но сочетающих при этом классические приёмы) визуальных средств. При этом достоинством DeepGraph является простота его освоения и интуитивного понимания параметров используемых в нём функций.

Целью данной работы является исследование возможностей визуализации различного типа графических сетей с помощью библиотеки DeepGraph и разработка на их основе методик работы с функциями модуля (создания узлов сетей разного типа, а также соединения их кромками).

Материалы и методы исследования

Первая методика визуализации узлов сети не требует даже использования модуля DeepGraph, достаточными являются функции библиотеки Pandas языка Python. Методика включает в себя последовательное применение следующих действий:

1) импортирование подбиблиотеки Pyplot графической библиотеки Matplotlib;

2) подключение модуля расширения Numpy [10] для добавления поддержки работы с массивами данных и библиотеки научных функций Pandas;

3) настройка параметров построчного вывода данных из подгружаемых баз данных модуля Pandas;

4) выполнение команды Pandas для чтения файла;

5) определение размеров изображаемой сетки графа с помощью модуля Matplotlib;

6) применение команды Scatter [11] библиотеки Matplotlib для изображения узлов графа, создаваемого по загруженной базе данных;

7) запись сетки графа на жёсткий диск в виде графического файла (например, имеющего формат jpg или png).

Используем данную методику для построения простой сетки графов без рёбер, соединяющих узлы. Как пример базы данных используем файл «flying_balls.csv» (с разделителями данных – запятыми), взятый из архива [6], структурированный следующим образом (приведён заголовок с названиями данных и первые несколько содержащих их строк; всего же файл содержит 1169 строк):

```
,time,x,y,ball_id
0,0,1692.0,0.0,0
1,0,8681.0,0.0,1
2,0,490.0,0.0,2
3,0,7439.0,0.0,3
```

Каждая строка приведённого файла данных содержит пять значений: первое означает номер строки, второе – номер графического слоя (означающий, например, время), третье – координату *x*, четвёртое – координату *y*, пятое – номер графа в текущем слое.

Данный граф является многослойным (с помощью слоёв можно имитировать поведение точек графа во времени), но при его изображении в файл выводятся только узлы (без рёбер). На рис. 1 изображена результирующая графическая сеть, построенная по базе данных «flying_balls.csv» со следующими настройками относительного размера выводимых фигур: ширина 6, высота 4. В результате размер всего изображения, определяемый конфигурацией элементов базы данных, получился равным 10000x500.

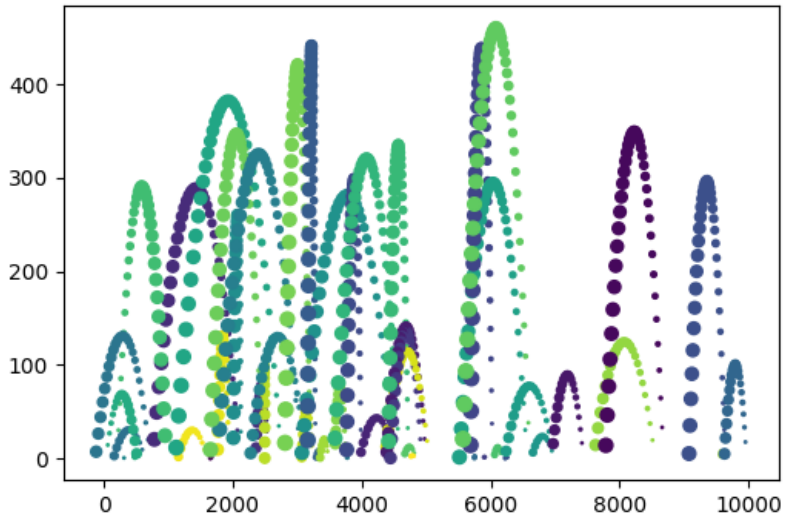

Рис. 1. Изображение многослойного графа, состоящего только из узлов

В приведённом примере команда построения точечного графика Scatter модуля Matplotlib для отображения времени протекания процесса преобразования узлов использует разный размер точек для разных моментов времени (с течением времени размер точки увеличивается). При этом для разных узлов графа используются разные цвета точек. Получающаяся таким образом команда построения узлов многослойного (мультивременного) графа выглядит следующим образом:

Scatter(v.x, v.y, s=v.time, c=v.ball_id),

где x означает координату x;

y – координату y;

s (size) – размер узловой точки (в зависимости от времени time);

c (color) – цвет точки (в зависимости от номера узловой точки).

Все представленные средства графического отображения различных характеристик графа делают получающееся изображение наглядным даже при его двухмерном представлении, как в данном случае.

Наряду с представленной методикой отображения баз данных в виде графов только с узлами, была разработана также и другая, более сложная, методика изображения сетевых графов ещё и с рёбрами. В данном случае для создания рёбер в программе становится необходимым использование специального модуля DeepGraph, включающего в себя ряд дополнительных функций. Также в программе необходимо сформировать четыре функции, определяющие координаты *x* и *y* начала и окончания каждого ребра. После этого заполняется массив координат рёбер (с использованием возможностей ещё одного дополнительного модуля Numpy), и полученные рёбра добавляются на изображение графа с применением команд графического модуля Matplotlib.Pyplot [12].

Также используются и описанные выше технологии, применяемые для создания графа, имеющего только узловые точки.

На рис. 2 изображён граф, полученный из файла данных «flying_balls.csv», с применением второй методики рисования сетевого графа, в котором узловые точки соединяются рёбрами в порядке увеличения номеров слоёв (по мере прохождения времени). Относительный размер и «скорость» уменьшения размера точки при этом взяты такими же, как и в первом примере (6 и 4 соответственно). Поэтому размер изображения получился, как и в первом случае (10000x500).

Результаты исследования и их обсуждение

Исходя из рис. 2 можно сделать вывод, что изображение сетевого графа, имеющее в своём составе соединяющие узлы рёбра, является более наглядным (при сравнении с рис. 1 без рёбер). В данном случае явственно видны пути смещения узлов при их временном развитии, полученные путём соединения найденных с помощью второй разработанной методики с использованием модуля DeepGraph координат начал и окончаний узлов. Так же как и рис. 1, данный граф является многослойным, и в некоторых областях рис. 2 это заметно.

Ещё более наглядным является трёхмерное изображение графа, которое можно вывести в графический файл при использовании функции Plot_3d библиотеки DeepGraph вместо использованной во втором методе функции Plot_2d.

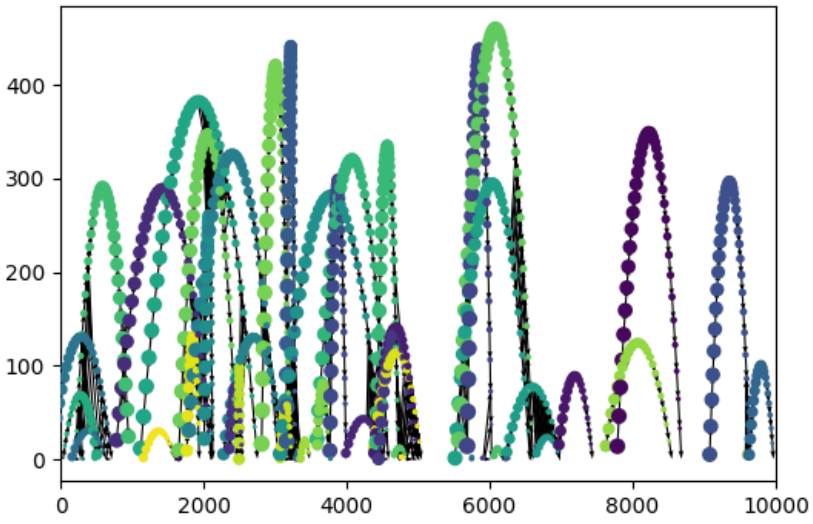

Рис. 2. Созданное из базы данных изображение сетевого графа, имеющего как узлы, так и рёбра

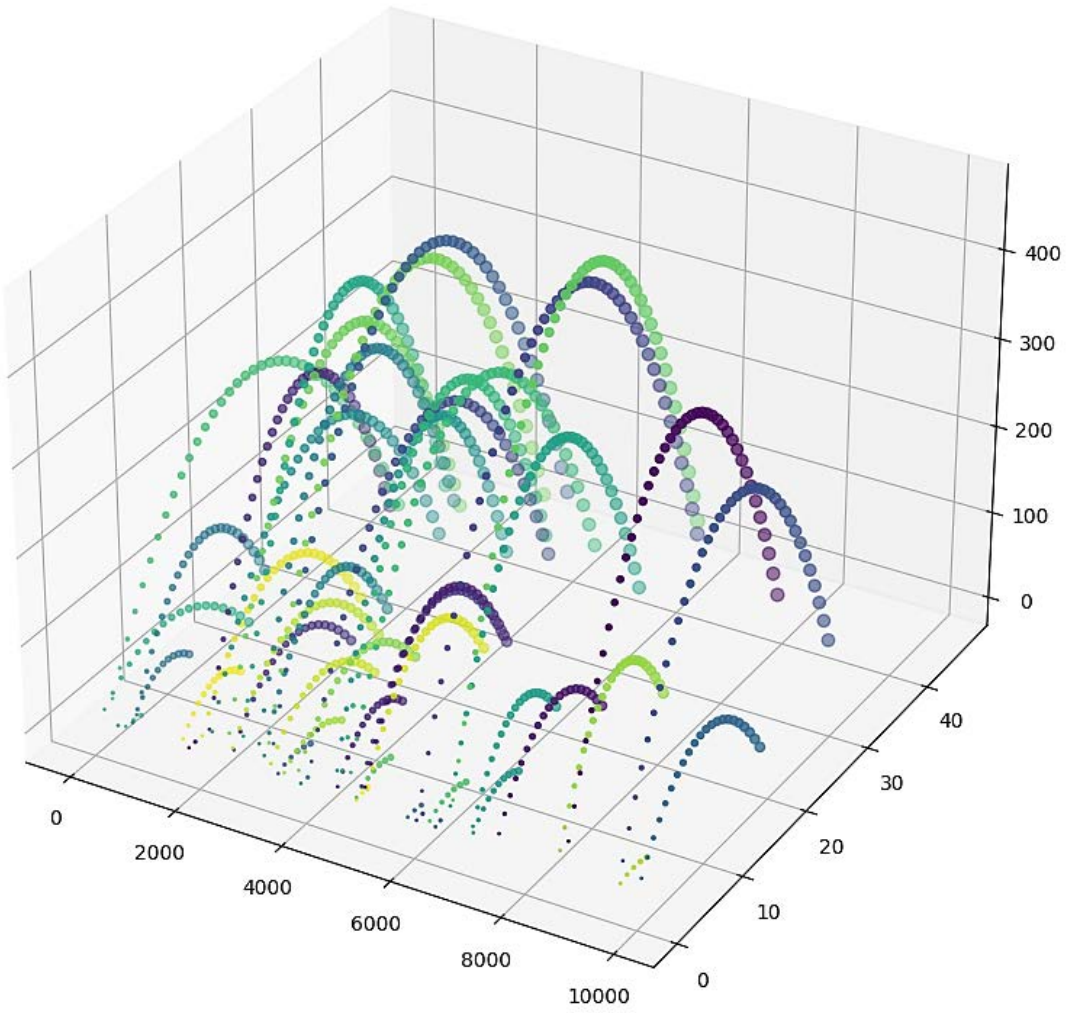

Рис. 3. Объёмное изображение сетевого графа

В качестве третьей координаты объёмного графика целесообразнее всего использовать графу базы данных «time» (время). Получаемая в этом случае картина для той же базы данных представлена на рис. 3.

В данном случае изменение положения узлов графа видно наиболее отчётливо, поэтому именно такой вид изображения рекомендуется использовать при демонстрации динамически изменяющихся сетевых графов.

Заключение

Таким образом, цели и задачи представленной работы полностью выполнены – разработаны действующие для любых баз данных три методики их визуализации в виде многослойных графов, как с использованием рёбер (и, соответственно, модуля DeepGraph), так и без использования рёбер, на которых присутствуют лишь узлы графа (в этом случае применение модуля DeepGraph не требуется), двухмерные и трёхмерные. Показаны примеры построения графов с помощью всех разработанных методик, которые можно без существенной модификации (исключая список исходных данных) применять также для визуализации любых подобных баз данных. Результаты выполнения программных кодов являются достаточно наглядными и рекомендуются для применения при любых видах исследований, использующих теорию графов (т.е. описанная работа является прикладной, направленной на практическое использование). Среди многочисленных работ авторов, посвящённых применению языка Python для математических вычислений, данное исследование раскрывает ещё одну область использования данного универсального языка программирования.

По результатам визуализации графов можно сделать определённые выводы: например, в данном случае становится понятным, что с течением времени положение каждого узла смещается по параболе своей, определённой, формы и что перемещение отдельных узлов является независимым.

Также можно сделать вывод и об удобстве и простоте применения именно библиотеки функций DeepGraph для обработки несложных баз данных. Также лёгкость понимания лежащих в основе разработанных методик алгоритмов позволяет рекомендовать использование языка программирования Python совместно с описанными библиотеками функций (главной из которых, конечно же, является модуль DeepGraph) для решения задач, связанных с визуализацией поведения сетевых графов, благодаря следующим доказанным достоинствам данного подхода:

1) высокая скорость вычислений и построения графических моделей;

2) понятность алгоритмов и кодов программ;

3) наглядность и многообразие форм визуализации результатов обработки баз данных;

4) использование только свободно распространяемых и доступных всем программных продуктов;

5) достижение понимания исследователем всех рассматриваемых процессов: от формулирования до визуализации решения задач, что является особенно важным для впервые изучающих рассматриваемые методы.

Методики и блоки программ на языке Python, разработанные в данной работе, можно использовать также внутри любых других программных продуктов.

Список литературы

1. Kozniewski E. Roof skeletons and graph theory trees // Геометрия и графика. 2016. Т. 4. № 1. С. 12–20.

2. Lee E., Park J., Cho N.I., Koo H.I. Deep-learning and graph-based approach to table structure recognition. Multimedia Tools and Applications. 2022. DOI: 10.1007/s11042-021-11819-7.

3. Ильичев В.Ю. Использование рекурсивных функций для создания фрактальной графики средствами языка Python // Системный администратор. 2021. № 3 (220). С. 92–95.

4. Шарафутдинов А.Г., Бударина Я.С. Компьютерные сети. Виды компьютерных сетей // Экономика и социум. 2014. № 2–4 (11). С. 1180–1181.

5. Zhansultan A., Sanzhar A., Trigo P. Parallel implementation of force algorithms for graph visualization. Journal of Theoretical and Applied Information Technology. 2021. Т. 99. No. 2. P. 503–515.

6. Deepgraph. [Электронный ресурс]. URL: https://github.com/deepgraph/deepgraph/blob/master/doc/source/tutorials/flying_balls.csv (дата обращения: 15.04.2022).

7. Ильичев В.Ю. Разработка программных продуктов с использованием модуля Python Coolprop для исследования эффективности утилизации тепла продуктов сгорания газообразных топлив // Системный администратор. 2020. № 11 (216). С. 80–83.

8. Gutman I. Relating graph energy with vertex-degree-based energies. Military Technical Courier. 2020. Т. 68. No. 4. P. 715–725.

9. Костенников Д.В. Обзор технологий визуализации графов на Python // Региональная информатика и информационная безопасность. СПб., 2020. С. 221–223.

10. Ilichev V.Yu. Development of program for determination of fractal dimensions of images. International Research Journal. 2021. № 4–1 (106). С. 6–10.

11. Doruyter A.G.G., Holness J.L. Dual energy window imaging for optimisation of P/V ratios in VP spect. EJNMMI Physics. 2021. Т. 8. No. 1. DOI: 10.1186/s40658-021-00417-z (дата обращения: 15.04.2022).

12. Ильичев В.Ю. Программа для вычисления площади фигуры сложной конфигурации разными способами // Системный администратор. 2021. № 1–2 (218–219). С. 134–137.

УДК 004.89

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ ANDROID ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Черевко Н.А., Белов Ю.С.

ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана», филиал, Калуга, e-mail: nikikwiki@mail.ru

Мобильные приложения ежедневно используются более чем половиной населения мира для выполнения самых разных задач. По мере все более широкого использования этих приложений возникает потребность в эффективных методах их тестирования. Традиционно компании использовали методы автоматического тестирования. Однако в последнее время все большее число организаций используют автоматизированные инструменты для тестирования. Многие фреймворки позволяют автоматизировать процесс тестирования приложений, однако существующие фреймворки в основном полагаются на разработчика приложений для предоставления тестовых сценариев для каждого разработанного приложения, что предотвращает повторное использование этих тестов для аналогичных приложений. В этой статье представляется новый подход к автоматизации тестирования приложений Android, используя методы машинного обучения и повторное использование популярных тестовых сценариев. Он основан на предположении, что различные компоненты в Android приложениях имеют схожую структуру интерфейса. Чтобы использовать это сходство, используются методы машинного обучения, которые классифицируют каждый компонент как один из семи типов заранее заготовленных компонентов, определённых в данной работе. Демонстрируются потенциальные преимущества нового подхода в эмпирическом исследовании, где показывается, что разработанный инструмент тестирования, основанный на предложенном подходе, превосходит стандартные методы в реальных условиях.

Ключевые слова: тестирование Android-приложений, автоматизация мобильного тестирования, классификация компонентов

AUTOMATION OF ANDROID APPLICATIONS TESTING USING MACHINE LEARNING ACTIVITIES CLASSIFICATION

Cherevko N.A., Belov Yu.S.

Bauman Moscow State Technical University, branch, Kaluga, e-mail: nikikwiki@mail.ru

Mobile applications are being used every day by more than half of the world's population to perform a great variety of tasks. With the increasingly widespread usage of these applications, the need arises for efficient techniques to test them. Traditionally, companies used manual testing methods for testing their applications. However, recently, a growing number of companies and organizations use automated tools for their testings. Many frameworks allow automating the process of application testing, however existing frameworks mainly rely on the application developer for providing testing scripts for each developed application, thus preventing reuse of these tests for similar applications. The problem with this approach is that these tests are hand-coded for specific applications and specific scenarios, and each new application requires spending many resources to reuse these tests. In this paper, we present a novel approach for the automation of testing Android applications by leveraging machine learning techniques and reusing popular test scenarios. We discuss and demonstrate the potential benefits of our approach in an empirical study where we show that our developed testing tool, based on the proposed approach, outperforms standard methods in realistic settings.

Keywords: Android application testing, mobile testing automation, activities classification

Мобильные устройства становятся ключевым компонентом в нашей жизни, и в настоящее время ими владеет более половины населения мира. На сегодня разработано более пяти миллионов приложений. По мере того, как мобильные устройства становятся все более популярными, возникает потребность в эффективных методах тестирования такого программного обеспечения. Исследование [1] обнаружило, что мобильные приложения склонны к значительному количеству дефектов и ошибок. Традиционно компании использовали методы автоматического тестирования. Однако в последнее время все большее число организаций используют автоматизированные инструменты для тестирования [2]. Автоматизация тестирования стала стандартом с множеством решений и фреймворков, позволяющих автоматизировать процесс тестирования приложений. Проблема данного подхода заключается в том, что тесты написаны вручную, для конкретных приложений и сценариев, и каждое новое приложение потребует затрат большого количества ресурсов для повторного использования этих тестов. Кроме того, каждое изменение в приложении приводит к доработке тестовых сценариев. В данной статье представлен новый подход к автоматическому тестированию Android-приложений, который позволяет найти наибольшее число функциональных ошибок. Он основан на предположении, что различные компоненты в Android приложени-

ях имеют схожую структуру интерфейса. Чтобы использовать это сходство, используются методы машинного обучения, которые классифицируют каждый компонент как один из семи типов заранее заготовленных компонентов, определённых в данной работе. Затем для каждого компонента запускаются специализированные тесты пользовательского интерфейса, которые были созданы с учетом структуры определённого типа компонента.

Цель исследования – изучить способы автоматизации тестирования программного обеспечения с помощью методов машинного обучения.

Материалы и методы исследования

Автоматическое тестирование. Создание инструмента, который позволит автоматически тестировать произвольное мобильное приложение, является чрезвычайно сложной задачей [3]. Многие из ошибок в мобильных приложениях отличаются от ошибок, представленных в традиционных настольных приложениях, в основном из-за неотъемлемого различия в архитектуре и методологиях разработки. Таким образом, привычные подходы к тестированию программного обеспечения не могут быть естественным образом перенесены на мобильные приложения. Чтобы уменьшить потребность в написании избыточных сценариев тестирования для мобильных приложений, которые имеют много общих характеристик, большое количество исследовательских усилий в последнее время было сосредоточено на разработке методов и алгоритмов автоматизированного тестирования, позволяющих автоматически тестировать приложения. Тестирование приложения Android с использованием методов автоматического тестирования обычно выполняется путем запуска назначенного приложения с генерацией событий пользовательского интерфейса, которые имитируют поведение пользователя, включая такие действия, как щелчки, прокрутка и пролистывание [4]. Это нужно для генерации как можно большего количества релевантных входных данных, чтобы изучить приложение с максимальным охватом.

На данный момент существующие методы тестирования Android-приложений можно разделить на три категории:

– Метод случайного тестирования: программы, использующие этот подход, генерируют события пользовательского интерфейса случайным образом, выполняя их одно за другим. В основном этот подход используется для проверки надежности приложения, так как большинство этих событий маловероятны для обычного пользователя.

– Метод исследований на основе модели: программы, использующие данный подход, создают модель графического пользовательского интерфейса приложения, чтобы использовать его для создания событий пользовательского интерфейса. Модель представляет собой конечный автомат, состояния которого являются различными компонентами приложения, а его переходы – различные события пользовательского интерфейса.

– Систематический метод тестирования: такие программы генерируют уникальное поведение пользователей путем динамического анализа исходного кода приложения.

Все вышеперечисленные подходы имеют одно заметное ограничение: эти методы направлены на поиск технических ошибок и дефектов в приложении, то есть сбоев, вызванных необработанными исключениями [5]. Однако многие из ошибок, присутствующих в современных приложениях, связаны с логикой программы.

Классификация типов компонентов. После изучения более сотни мобильных приложений были выведены семь видов компонентов.

1. Загрузочный экран. Это экран, отображаемый при открытии компонентов, на котором обычно изображены картинка или текст. Пока мы видим данный компонент, приложение загружается в фоновом режиме. Чаще всего заставка является первым экраном в приложении.

2. Компонент-реклама. Большинство Android-приложений можно загрузить бесплатно. Поэтому для получения прибыли разработчики включают в продукт рекламу. Рекламные объявления могут появиться в любое время в любом месте приложения, что усложняет их классификацию. Идентификация таких действий важна, так как нажатие объявления во время тестирования приведет к выходу из приложения.

3. Компонент входа. Множество современных приложений требуют имени пользователя и пароля для использования служб приложения.

4. Компонент соединения: У многих современных средств коммуникации, таких как веб-сайты, новостные порталы, онлайн-телевидение, есть специальное мобильное приложение, позволяющее пользователю получить доступ к медиаконтенту на своем телефоне. Компонент соединения – это экран, содержащий такие приложения.

5. Почтовый компонент. Это компонент, агрегирующий почтовые приложения с такими функциями, как управление входящей

почтой и отправкой новых писем. Такие приложения имеют четко определенную цель с ограниченным количеством возможных действий.

6. Компонент-браузер. Браузеры – это одни из наиболее важных приложений, поскольку они позволяют пользователю получать доступ к веб-сайтам на своем мобильном устройстве.

7. Компонент-список. Приложения со списками обычно имеют общую цель – отслеживать список задач пользователя.

Построение вектора признаков. Каждый компонент может быть наполнен большим количеством функций, связанных с содержащимися в них элементами пользовательского интерфейса, такими как классы элементов, атрибуты, различное их положение в компоненте, а также навигационная панель [6]. При построении признакового описания объектов необходимо решить, какие элементы являются наиболее информативными и могут различаться для разных компонентов. Такие элементы являются интерактивными. Кроме того, изучив базовые шаблоны компонентов из руководства по дизайну компонентов Android Studio, можно сделать вывод, что каждый компонент можно разделить на три части: верхнюю, среднюю и нижнюю. Используется следующее деление экрана: 20 % – 60 % – 20 %, как показано на рис. 1.

Это позволяет выделить следующие группы интерактивных элементов:

– Элементы, которые можно нажимать.

– Горизонтально пролистываемые элементы.

– Вертикальные элементы, которые можно смахивать.

– Элементы текстового поля, в которые можно вводить текст.

Классификатор. Для создания классификатора используется специальный набор программного обеспечения для машинного обучения [7]. Требуется провести десятикратный процесс классификации набора данных с различными алгоритмами классификации, измеряя точность каждого из них. На рис. 2 показана точность, усредненная по предсказанию типа компонента с использованием различных моделей классификации.

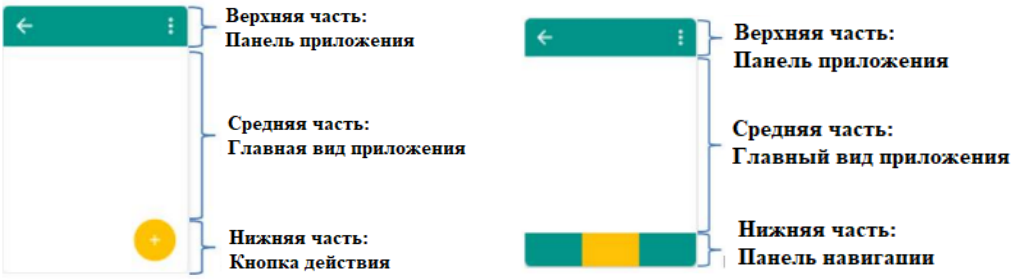

Рис. 1. Стандартные шаблоны Android-приложений

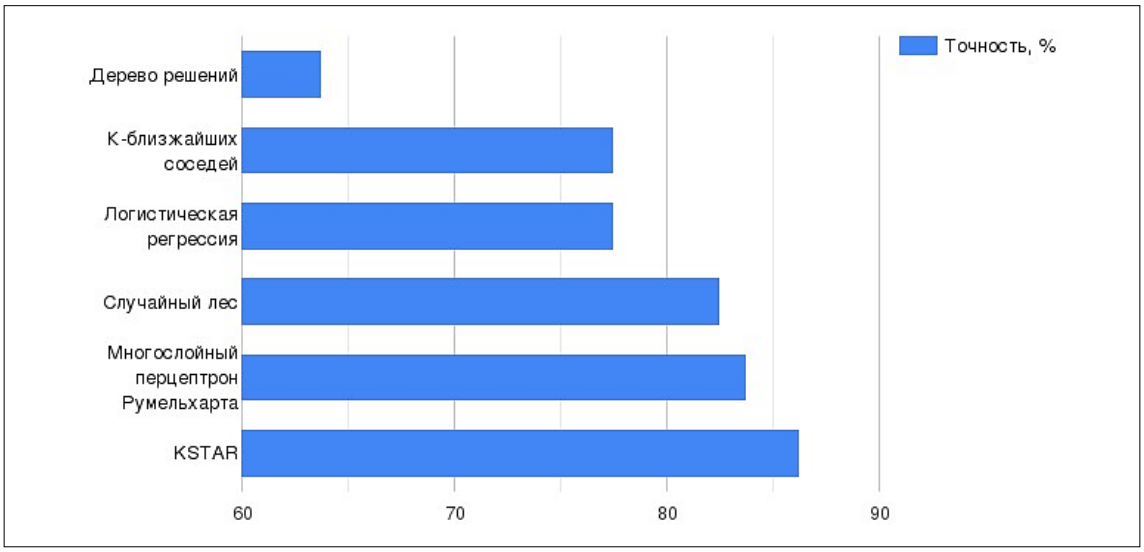

Рис. 2. Точность предсказания типа компонента с использованием различных моделей классификации

Как видно из рис. 2, с помощью классификатора KSTAR удалось достичь наивысшей точности классификации. Таким образом, моделью выбора для данной работы становится KSTAR. Данный классификатор использует меру энтропийного расстояния в качестве функции сходства, чтобы определить, какие из экземпляров больше всего похожи на исходный [8]. Кроме того, требуется оценить, какие функции вносят больше информации в модель исследования. Для этого нужно провести отбор признаков на основе коэффициента прироста информации, который оценивает «ценность» признака путем измерения прироста информации по отношению к классу.

Функции, связанные с количеством нажимаемых элементов в разных частях экрана, так же как и количество общих элементов, показывают высокий коэффициент прироста информации при классификации компонента по типу. Однако функции, связанные со смахиваемыми элементами, и текстовые поля в верхней и нижней частях экрана не вносят информации в модель.

Практическая оценка метода в реальном приложении. В качестве эксперимента используются два мобильных приложения с открытым исходным кодом. Чтобы протестировать эффективность выбранного метода, требуется искусственно добавить ошибки.

– *«K-9Mail»* – приложение электронной почты. Рассмотрены следующие компоненты:

о *«Список сообщений»* – компонент почты, содержащий следующие ошибки:

- Пользователь не может открыть содержимое письма из списка входящих.

- Пользователь может отправить электронное письмо без адреса получателя.

- Пользователь не может отправить письмо на действительный адрес электронной почты.

- Пользователь может отправить электронное письмо с недопустимым адресом получателя.

о *«Вход»* – компонент входа содержит следующие ошибки:

- Пользователь может войти в систему без ввода имени пользователя и пароля.

- Пользователь может войти с неверным именем пользователя и неверным паролем.

- Пользователь не может войти в систему с действительным именем пользователя и действительным паролем.

– *«CrimeTalk Reader»* – приложение для просмотра статей. Рассмотрены следующие компоненты:

о *«Главный компонент»* – компонент содержит следующие ошибки:

- Пользователь не может провести пальцем по экрану влево и вправо, чтобы просмотреть различные разделы портала.

- Пользователь не может нажимать на разные вкладки меню для просмотра различных разделов.

- Пользователь не может открыть статью.

Затем специальное программное обеспечение в течение двух минут совершало 50,000 псевдослучайных действий в каждом компоненте. Параллельно выполнялись обычные автоматические тесты. В результате были получены отчеты о количестве обнаруженных логических ошибок, а также классификация каждого компонента.

Результаты исследования и их обсуждение

В таблице представлены результаты эксперимента. Из нее можно выделить три основные тенденции: 1) тестируемый метод смог правильно классифицировать три «невидимых» действия; 2) удалось обнаружить все «заложенные» ошибки, в то время как стандартный тест не обнаружил ни одной. Кроме того, удалось найти логическую ошибку, которая уже была частью исходной версии приложения, без вмешательства в код; 3) стандартным методом удалось обнаружить один сбой в реальном времени, который не был обнаружен тестируемым методом.

Ошибки, обнаруженные стандартным методом автоматического тестирования

Приложение/компонент	Встроенный/добавленный (тип ошибки)	Критическая ошибка в реальном времени	Логические ошибки
K-9Mail: Список сообщений	Встроенный	0 критических ошибок	0 логических ошибок
	Добавленный	1 критическая ошибка	
K-9Mail: Вход	Встроенный	0 критических ошибок	
	Добавленный	0 критических ошибок	
CrimeTalk Reader: Главный компонент	Встроенный	0 критических ошибок	
	Добавленный	0 критических ошибок	

У данного подхода есть ограничения. Поскольку система может иметь бесконечное количество запусков, проверка поведения приложения ограничивается запусками, которые фактически выполняются. Таким образом, тестируемый метод ограничен типами компонентов и функциями, которые были определены заранее. Определенные в данной работе семь типов компонентов были разработаны для доказательства эффективности выбранного подхода, в то же время настоящий продукт будет содержать больше видов компонентов и функций. Кроме того, стоит учитывать объем эксперимента (два приложения), а также тот факт, что некоторые ошибки были заложены в приложения заранее.

Заключение

В данной статье представлен новый подход к тестированию Android-приложений с использованием методов машинного обучения. Использование таких методов позволило классифицировать каждое действие пользователя по определенному типу, что в свою очередь позволяет протестировать различные ожидаемые варианты поведения различных компонентов. Кроме того, данный метод протестирован на разных приложениях и продемонстрировал преимущество перед популярными методами автоматического тестирования. Наглядно продемонстрировано, что тестируемый метод обнаруживает логические ошибки приложения, а не только критические ошибки в реальном времени, что открывает возможность разработки более сложных инструментов тестирования.

Список литературы

1. Takeuchi Mi., Gustiar Mu. Development Money Diary Application Models on Android. International Conference on Information Management and Technology. 2020. P. 1–4.

2. Naja Fa., Mansur Sy., Wibawanto Ad. Automated Software Testing on Mobile Applications: A Review with Special Focus on Android Platform. 20th International Conference on Advances in ICT for Emerging Regions. 2020. P. 4–6.

3. Pan M., Xu To., Pei Yu. GUI-Guided Test Script Repair for Mobile Apps. IEEE Transactions on Software Engineering. 2022. Vol. 48. No. 3. P. 3–5.

4. Motan Ma., Zein Sa. Android App Testing: A Model for Generating Automated Lifecycle Tests. 4th International Symposium on Multidisciplinary Studies and Innovative Technologies. 2020. P. 2–3.

5. Md Faiz., Anwar Md. Hybrid Classification Model to Detect Android Application-Collusion. 43rd International Conference on Telecommunications and Signal Processing. 2020. P. 1–2.

6. Mahmud Ta. API Compatibility Issue Detection, Testing and Analysis for Android App. 36th IEEE/ACM International Conference on Automated Software Engineering. 2021. P. 6–8.

7. Jha K., Nadi S. Annotation practices in Android apps. IEEE 20th International Working Conference on Source Code Analysis and Manipulation. 2020. P. 2–3.

8. Черевко Н.А., Белов Ю.С. Применение технологий машинного обучения в тестировании программного обеспечения // Высокие технологии и инновации в науке. Международной научной конференции. СПб., 2021. С. 127–129.

УДК 004.67

ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ ДЛЯ МАШИННОГО ОБУЧЕНИЯ

Акимов А.А., Валитов Д.Р., Кубряк А.И.

Стерлитамакский филиал Башкирского государственного университета, Стерлитамак, e-mail: denisvalitof@yandex.ru

Данные – это не что иное, как актив в современном мире. Данные в настоящее время сильно искажены несоответствиями, шумом, неполной информацией и пропущенными значениями. Они агрегируются из разнообразных источников с использованием методов интеллектуального анализа данных и хранилищ. Предварительная обработка данных служит основой для достоверного анализа. Это необходимый шаг в построении анализа оперативных данных, учитывая недостатки в их качестве. Также предобработка относится к основному набору методов для повышения качества исходных данных, таких как очистка, нормализация, отбор необходимых признаков и экземпляров. Так как полученные данные представлены в необработанном виде, обучение модели с их использованием может оказаться недостижимым. Эта статья представляет собой обзор методов предварительной обработки для анализа используемых данных. Также в ней рассмотрены этапы предварительной обработки данных, изучены виды признаков в машинном обучении, произведен обзор категориальных переменных. Продемонстрированы пять этапов предобработки данных, рассмотрены виды признаков при обработке данных для машинного обучения, описаны категориальные признаки и приведены два вида примеров категориальных переменных.

Ключевые слова: данные, обработка, категориальные переменные, генерация признаков, очистка данных

DATA PREPROCESSING FOR MACHINE LEARNING

Akimov A.A., Valitov D.R., Kubryak A.I.

Sterlitamak branch of Bashkir State University, Sterlitamak, e-mail: denisvalitof@yandex.ru

Data is nothing but an asset in the modern world. The data is currently heavily distorted by inconsistencies, noise, incomplete information and missing values. They are aggregated from a variety of sources using data mining and storage methods. Preliminary data processing serves as the basis for reliable analysis. This is a necessary step in building an analysis of operational data, given the shortcomings in their quality. Also, preprocessing refers to the main set of methods to improve the quality of the source data, such as cleaning, normalization, selection of necessary features and instances. Since the data obtained is presented in raw form, training the model using them may not be achievable. This article is an overview of preprocessing methods for analyzing the data used. The stages of data preprocessing are also considered, the types of features in machine learning are studied, and a review of categorical variables is made. Five stages of data preprocessing are demonstrated, the types of features in data processing for machine learning are considered, categorical features are described and two types of examples of categorical variables are given.

Keywords: data, processing, categorical variables, feature generation, data cleaning

Предварительная обработка данных в машинном обучении – это важный шаг, который помогает повысить качество данных. Предобработка данных в машинном обучении относится к технике подготовки необработанных данных с целью сделать их пригодными для построения и обучения моделей машинного обучения. Иными словами, это метод интеллектуального анализа данных, который преобразует необработанные данные в понятный и читаемый формат.

Предварительная обработка данных является одним из основных этапов, от качества выполнения которого зависит получение качественных результатов процесса анализа данных. Без подготовки данных не обходится ни один нейросетевой метод. Как правило, при описании различных нейроархитектур предполагается, что данные для обучения уже представлены в том виде, в котором требует нейросеть, однако на практике дела обстоят совсем иначе, именно этап предобработки данных может занимать большую часть времени, отведенного на проект в целом. Результат обучения нейросети также может зависеть от того, в каком виде представлена информация для ее обучения. Таким образом, предварительная обработка данных позволяет повысить качество как интеллектуального анализа данных, так и самих данных.

Цель исследования – рассмотрение этапов предварительной обработки данных и основ разработки признаков, а также обзор конструирования категориальных признаков.

Материалы и методы исследования

При создании модели машинного обучения предварительная обработка данных служит первым шагом. Как правило, реальные данные являются неполными, непоследовательными, могут содержать ошибки или выбросы, а также в них могут отсутствовать

конкретные значения или атрибуты. Именно здесь используется предобработка данных: она помогает очищать, форматировать и упорядочивать необработанные данные, тем самым делая их готовыми к работе с моделями машинного обучения.

Предобработка данных состоит из пяти этапов:

− очистка данных, которая направлена на повышение качества данных за счет присваивания пропущенных значений и удаления выбросов;

− сокращение объема данных, которое уменьшает объем данных и, следовательно, снижает связанные с ними вычислительные мощности;

− масштабирование данных – направлено на преобразование исходных данных в аналогичные диапазоны для прогнозного моделирования;

− преобразование, целью которого является организация исходных данных в подходящие форматы для различных алгоритмов интеллектуального анализа данных;

− разделение, которое делит весь набор данных на различные подмножества для более глубокого анализа.

Существует два общих способа обработки отсутствующих значений при построении оперативных данных. Первый – просто отбросить выборки данных с пропущенными значениями, так как большинство алгоритмов получения данных не могут обрабатывать данные с пропущенными значениями. Такой метод применим только тогда, когда доля отсутствующих значений незначительна. Второй – применение методов интерполяции пропущенных значений для замены пропущенных данных значениями, полученными в результате расчетов.

Как показано на рисунке 1, распространенные методы интерполяции отсутствующих значений можно разделить на две группы, т.е. одномерные и многомерные методы.

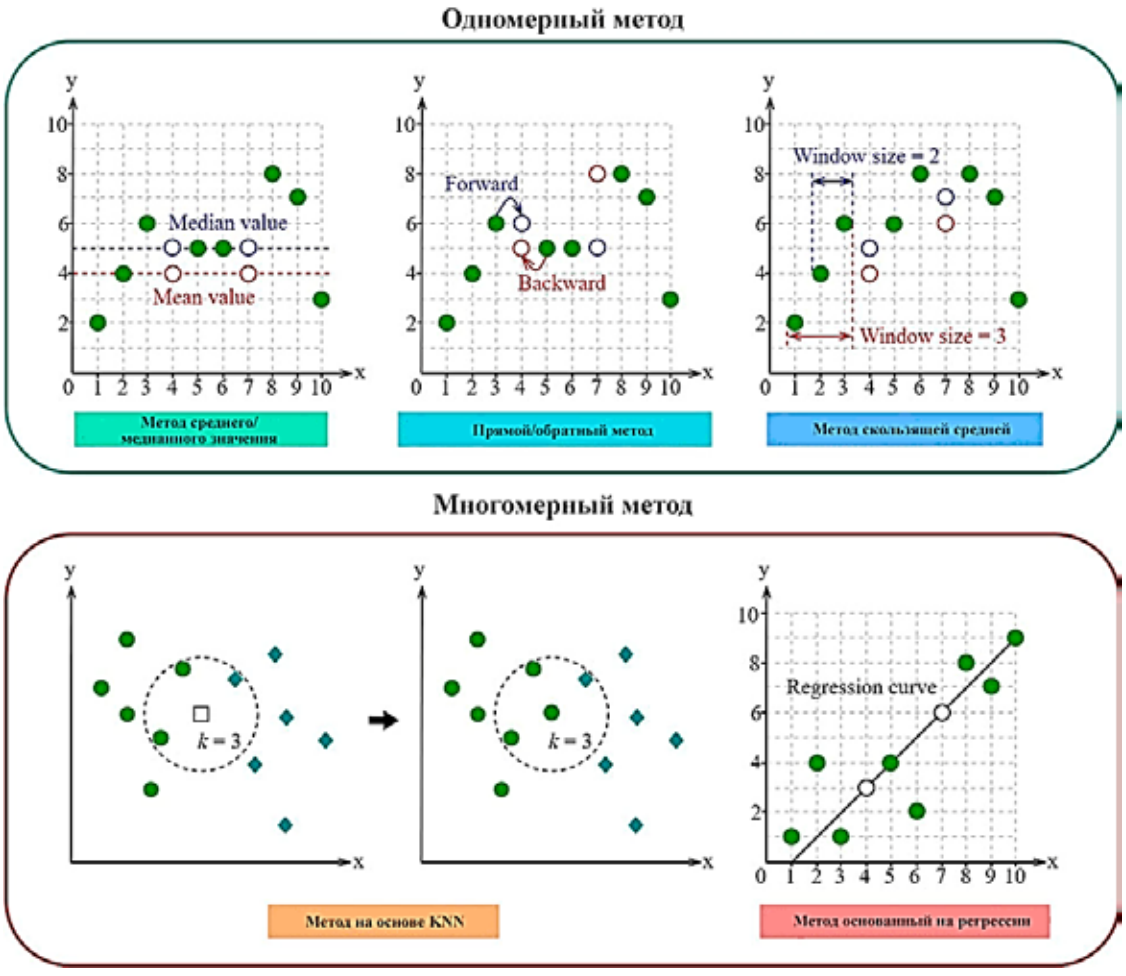

Рис. 1. Методы интерполяции пропущенных значений для построения данных

К первым относятся средняя интерполяция, прямая или обратная интерполяция и методы скользящего среднего. В этом случае недостающие значения выводятся на основе характеристик данных только одной переменной и поэтому называются одномерными методами. Метод интерполяции среднего или медианы заменяет недостающие значения средним или медианой этой переменной. Прямой или обратный метод просто заменяет отсутствующее значение предыдущим или следующим измерением данных. Эти два метода просты в реализации, но не учитывают временные корреляции на временных этапах и могут не сделать правильную замену данных [1].

В основном используются два метода обнаружения выбросов – статистические методы и методы, основанные на кластеризации.

Сокращение данных обычно проводится в двух направлениях, т.е. по строкам для сокращения выборки данных и по столбцам для сокращения переменных данных. Для сокращения данных по строкам могут применяться различные методы выборки данных, такие как случайная и стратифицированная выборка.

Существуют три основных метода сокращения переменных данных по столбцам. Первый заключается в использовании знаний о домене для прямого отбора интересующих переменных. Второй – использование статистических методов отбора признаков для выбора важных переменных для дальнейшего анализа. Третий – использование методов извлечения признаков для построения полезных признаков для анализа данных.

Масштабирование данных часто необходимо для обеспечения достоверности прогностического моделирования, особенно когда входные переменные имеют различные масштабы. Нормализация max-min (т.е., $x'=x-xmin/xmax-xmin$) и стандартизация z-score (т.е., $x'=x-\mu/\sigma$) – два наиболее широко используемых метода, где min(x) и max(x) означают минимум и максимум переменной x, значения переменной μ – среднее значение, а σ – стандартное отклонение.

Метод нормализации max-min чувствителен к выбросам данных, поскольку их присутствие может резко изменить диапазон данных. В отличие от него метод стандартизации z-score менее подвержен влиянию выбросов. Он обычно используется для реформирования переменной, чтобы она была нормально распределена со средним значением, равным нулю, и стандартным отклонением, равным единице. Теоретически, нормализация по z-score работает лучше всего, когда данные нормально распределены. Нормализация по методу max-min рекомендуется, когда эксплуатационные данные не соответствуют нормальному распределению и не содержат явных выбросов. Другой тип метода масштабирования данных может изменять структуры данных. Например, исходные данные могут быть отображены в новое пространство с помощью определенных математических функций, таких как логарифмическая, сигмоидальная функции или арктангенс. Такие методы часто используются для минимизации дифференциалов в переменных данных [2].

Преобразование данных в основном применяется для преобразования числовых данных в категориальные для обеспечения совместимости с алгоритмами интеллектуального анализа данных. Методы равной ширины и равной частоты широко используются благодаря своей простоте. Количество интервалов обычно предопределяется пользователем на основе знаний о предметной области. По сравнению с методом равной ширины метод равной частоты менее чувствителен к выбросам [3].

Преобразование данных также может применяться для преобразования категориальных переменных в числовые для облегчения разработки моделей прогнозирования. Для этой цели широко используется метод однократного кодирования, при котором для категориальной переменной с L уровнями создается матрица из L – 1 столбцов [4]. Все признаки могут быть следующих видов:

− бинарные, которые принимают два значения (да/нет, 0/1, true/false);

− номинальные, которые имеют конечное количество уровней. Также они могут быть упорядоченными и неупорядоченными;

− количественные значения в диапазоне от $-\infty$ до $+\infty$.

Признак – это переменная, которая описывает отдельную характеристику объекта. В табличном представлении выборки признаки – это столбцы таблицы, а объекты – строки. Входные, независимые переменные для модели машинного обучения называются предикторами, а выходные, зависимые – целевыми признаками.

Признаки могут извлекаться из данных любого типа, в том числе из текста, изображений и геоданных. При обработке текстовой информации сначала выполняется ее токенизация, а затем лемматизация и цифровизация – перевод слов в числовые вектора. В случае изображений часто анализируется не только содержание картинки как набора пикселей различного цвета, но и метаданные графического файла: дата

съемки, разрешение, модель камеры и т.д. Географические данные чаще всего представлены в виде адресов или пар [5].

Разработка признаков – очень важный аспект машинного обучения и науки о данных, и его нельзя игнорировать. Основная цель разработки признаков – получить наилучшие результаты от алгоритмов. В науке о данных производительность модели зависит от предварительной обработки и обработки данных. Если модель построена без обработки данных, то точность будет составлять около 70%. Применив генерацию признаков к той же модели, производительность можно повысить на несколько десятков процентов. Проще говоря, благодаря генерации признаков улучшается производительность модели.

Выбор признаков – это не что иное, как выбор необходимых независимых признаков. Выбор важных независимых признаков, которые имеют большую связь с зависимым признаком, поможет построить хорошую модель. Существует несколько методов отбора признаков [6].

В случае одномерного массива статистические тесты могут использоваться для отбора независимых признаков, которые имеют наиболее сильную связь с зависимым признаком. Метод SelectKBest может быть использован с набором различных статистических тестов для выбора определенного количества признаков (рис. 2). Признак, имеющий наивысший балл, будет более связан с зависимым признаком, и эти признаки будут выбраны для модели.

Метод ExtraTreesClassifier помогает определить важность каждого независимого признака с зависимым признаком. Важность признака дает оценку для каждого признака данных: чем выше оценка, тем важнее или релевантнее признак текущей выходной переменной (рис. 3).

Тепловая карта – это графическое представление двумерных данных (рис. 4). Здесь каждое значение данных представлено в матрице. Необходимо построить парный график между всеми независимыми и зависимыми функциями, в итоге получится отношение между двумя признаками. Если отношение между независимой и зависимой функциями меньше 0,2, тогда эта независимая функция выбирается для построения модели.

```
In [43]: from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import chi2
         bestfeatures = SelectKBest(score_func=chi2, k=10)
         fit = bestfeatures.fit(x,y)
         dfscores = pd.DataFrame(fit.scores_)
         dfcolumns = pd.DataFrame(x.columns)
         featureScores = pd.concat([dfcolumns,dfscores],axis=1)
         featureScores.columns = ['Specs','Score']
         featureScores
```

```
In [47]: featureScores
```

Out[47]:

	Specs	Score
0	City_Code	0.122643
1	Region_Code	74.805013
2	Accomodation_Type	0.756115
3	Reco_Insurance_Type	3.965972
4	Upper_Age	2.612166
5	Lower_Age	1.572930
6	Is_Spouse	0.632296
7	Health Indicator	0.453390
8	Holding_Policy_Duration	7.662646
9	Holding_Policy_Type	0.836189
10	Reco_Policy_Cat	1894.032997
11	Reco_Policy_Premium	9575.065324
12	diff_age	0.039347

Рис. 2. Пример одномерного отбора

```
In [192]: feat_importances = pd.Series(model.feature_importances_, index=x.columns)
          feat_importances.nlargest(10).plot(kind='barh')
          plt.show()
```

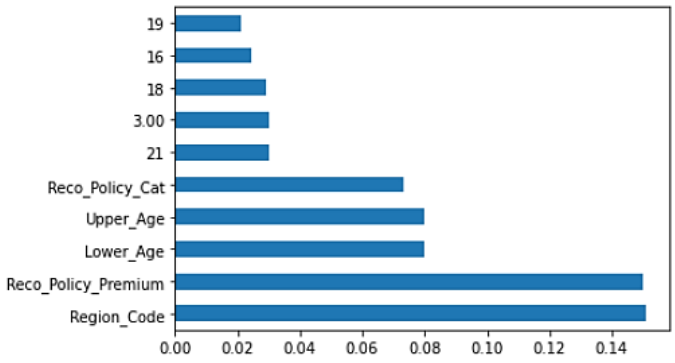

Рис. 3. Пример метода ExtraTreeClassifier

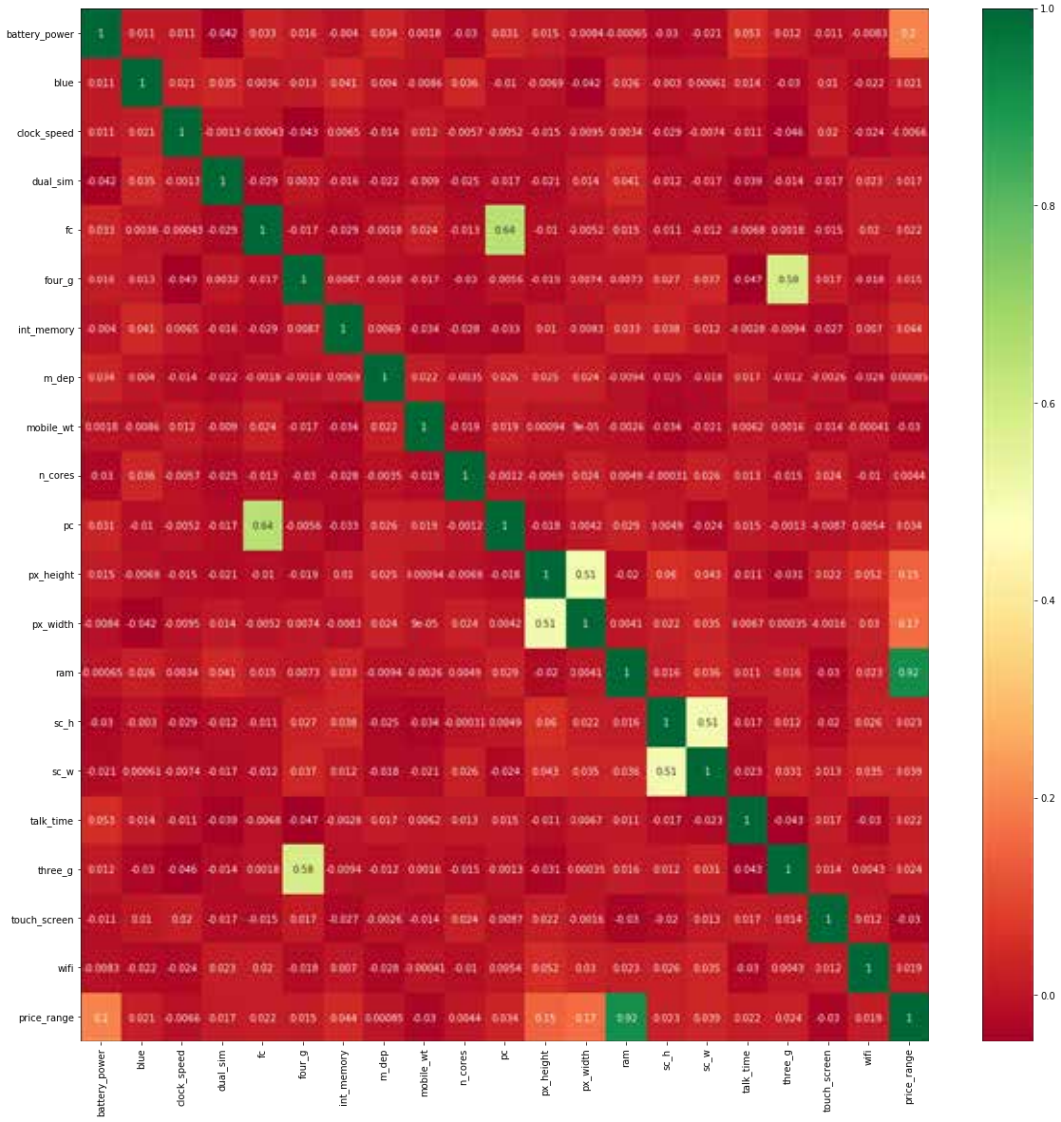

Рис. 4. Пример тепловой карты

Категориальные данные – это тип данных, который используется для группировки информации с похожими характеристиками, в то время как числовые данные – это тип данных, выражающих информацию в виде чисел. Примером категориальных данных может служить пол человека [7].

Большинство алгоритмов машинного обучения не могут работать с категориальными переменными, если их не преобразовать в числовые значения. Производительность многих алгоритмов даже зависит от того, как закодированы категориальные переменные.

Категориальные переменные можно разделить на две категории:

− номинальные, которые не имеют определенного порядка;

− порядковые, между значениями которых существует определенный порядок.

Заключение

Предобработка данных является важнейшим этапом построения моделей машинного обучения, она занимает большую часть времени, так как от подготовки данных зависит корректность будущей модели. В случае ошибки при первичном анализе возможны переобучение модели или утечка данных, которые могут нарушить корректность работы модели. Таким образом, предварительная обработка данных позволяет значительно повысить качество как самих данных, так и результата анализа.

Список литературы

1. Быков К.В. Особенности предобработки данных для применения машинного обучения // Молодой ученый. 2021. № 53 (395). С. 1-4. [Электронный ресурс]. URL: https://moluch.ru/archive/395/87491/ (дата обращения: 22.04.2022).

2. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. М.: ДМК Пресс, 2015. 400 с.

3. Близнюк Б., Васильева Л., Стрельников П., Ткачук Д. Современные методы обработки естественного языка // Вестник Харьковского национального университета им. Каразина. Серия «Математическое моделирование. Информационные технологии. Автоматизированные системы управления». 2017. № 36. С. 14-26.

4. Kumar D. Introduction to Data Preprocessing in Machine Learning. [Электронный ресурс]. URL: https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d (дата обращения: 22.04.2022).

5. Акимов А.А., Мустафина С.А. Обзор современных методов искусственного интеллекта по распознаванию девиантного поведения индивида // Вестник Технологического университета. 2020. Т. 23. № 8. С. 69-79.

6. Mustafina S., Akimov A., Plotnikova A. Comparison of semantic convolution neural networks on the example of crack segmentation in asphalt images. International Journal of Computing. 2021. Т. 19. № 3. С. 415-423.

7. Hamza A. Effective Data Preprocessing and Feature Engineering. [Электронный ресурс]. URL: https://becominghuman.ai/effective-data-preprocessing-and-feature-engineering-452d3a948262 (дата обращения: 22.04.2022).