Б Ъ Л Г А Р С К А А К А Д Е М И Я Н А Н А У К И Т Е
ИНСТИТУТ ПО ИНФОРМАЦИОННИ И
КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

Кристина Иванова Динева

ИНТЕГРИРАНЕ НА ХЕТЕРОГЕННИ ДАННИ ОТ РАЗПРЕДЕЛЕНИ IoT УСТРОЙСТВА

ДИСЕРТАЦИЯ

за присъждане на образователна и научна степен „доктор“
по докторска програма „Информатика“
професионално направление 4.6. “Информатика и компютърни науки“

Научен ръководител: доц. д-р Татяна Атанасова

София, 2020 г.

Съдържание

Речник на термини и съкращения, използвани в дисертационния труд

Artificial intelligence (AI) – технология занимаваща се с изграждане на интелигентни машини, способни да изпълняват задачи, които обикновено изискват човешки интелект.

ASSN (Application Specific Sensor Nodes) – специфични сензорни възли.

Back-end - слой за достъп до данни в софтуерна архитектура.

Basic Authentication - схема за удостоверяване, вградена в HTTP протокол.

Big Data - големи масиви от данни и технологии за работа с тях.

Cloud - отдалечени сървъри, за съхранение и достъп до данни и програми през интернет.

Cloud computing - съхранение, управление, обработка на данни и извършване на изчислителни дейности върху отдалечени сървъри, хоствани в интернет.

Exploratory data analysis (EDA) - подход за анализ на набори от данни с цел обобщаване на основните им характеристики.

Extensible Markup Language (XML) – стандарт, разработен от World Wide Web Consortium (W3C) за структуриране на данни, формат на данни за електронни документи. Широко приет като универсален език за обмен на информация между приложения, системи и устройства в интернет.

Event Bus – система за предаване на съобщения до нужния получател.

Fog Computing - децентрализирана изчислителна инфраструктура, където изчислението и съхранението да данни се извършва между източника на данни и Cloud.

Hyper Text Transfer Protocol Secure (HTTPS) – интернет комуникационен протокол, който защитава целостта и конфиденциалността на данните между компютъра на потребителя и уеб приложението.

IA (Information Architecture) – информационна архитектура.

Internet of Things (IoT) – свързани чрез интернет компютърни устройства, вградени в ежедневни обекти, които събират и изпращат данни за състоянието на обектите.

IP Whitelisting - функция за защита на IP адрес или IP домейн, често използвана за ограничаване и контрол на достъпа само до доверени потребители.

IPv6 - комуникационен протокол, който осигурява система за идентификация и местоположение за компютри в мрежи и трафик на маршрути през интернет.

Lightweight Machine to machine (LwM2M) - протокол за дистанционно управление на устройства с M2M или IoT.

Local Area Network (LAN) - локална компютърна мрежа, която свързва компютрите в ограничена зона.

Machine to Machine (M2M) – технология, която позволява на мрежовите устройства да обменят информация и да извършат действия без ръчна помощ от човек.

Message Queuing Telemetry Transport (MQTT) - протокол за сензори и мобилни устройства, оптимизиран за мрежи с висока латентност.

MSA (Microservice architecture) – архитектура основана на микросървиси.

Not a Number (NAN) - числов тип данни, който може да бъде интерпретиран като стойност, която е неопределена.

Not Available (NA) - липсваща стойност.

Null - празен обект.

Outliers - стойности позициониращи се извън границите на нормалното разпределение.

REST (Representational State Transfer) – протокол за работа с данни за реализация на уеб услуги.

Snowflake – логическа схема на групиране „Снежинка“.

SPA (Single Page Application) - уеб приложение, което взаимодейства с браузъра чрез динамично пренаписване на текущата страница с нови данни от сървъра.

SSH File Transfer Protocol (SFTP) - защитен протокол за прехвърляне на файлове. Той работи над SSH протокола.

UX (User Experience) - потребителско изживяване при работа с потребителски интерфейс.

Virtual Private Network (VPN) - анонимна виртуална частна мрежа.

Web Service - представлява софтуерна услуга, която предоставя комуникация между взаимно съвместими компютърни системи по компютърни мрежи.

WebSocket – компютърен комуникационен протокол, осигуряващ двупосочни комуникационни канали през една TCP връзка.

Z-Wave - безжичен комуникационен протокол. Използва радио вълни за комуникация с ниско потребление на енергия.

ZigBee - стандарт базиран на IEEE 802.15.4 спецификация за набор от комуникационни протоколи, използвани за създаване на частни мрежи с малки цифрови радиостанции с ниска мощност.

Увод

Информационната революция е резултат от развитието на съвременните технологии и се основава на постиженията на научно-техническия прогрес. Нарастващото присъствие на WiFi и 5G безжичен достъп до интернет води до бързи темпове на развитие и увеличаващ се брой на взаимосвързани устройства.

В парадигмата на "Интернет на нещата" (IoT), обработката на данни придобити от IoT устройства, може да бъде обособена като отделна академична дисциплина, насочена към разработване и надграждане на методи и инструменти, които са важни за повишаване на производителността на труда, конкурентоспособността на производството и качеството на живот. Мрежите от взаимосвързани обекти, не само извличат информация от околната среда и взаимодействат с физическия свят, но също така използват съществуващите интернет стандарти за предоставяне на услуги за трансфер на данни в отдалечена облачна среда. Това води до генериране на огромни количества данни, които трябва да се съхраняват, обработват и представят в лесна и ефективна за интерпретиране форма. Облачните услуги могат да осигурят виртуална инфраструктура за такива процеси, които позволяват данни от устройства за наблюдение да бъдат обработени, анализирани, моделирани и визуализирани в специално разработени за тях платформи.

В същото време, развитието на "Интернет на нещата" създава значителни предизвикателства, които затрудняват пълното реализиране на неговия потенциал.

Настоящият дисертационен труд анализира методи и средства за интегриране, обработка и моделиране на хетерогенни данни, получени от разпределени IoT устройства. На тази основа са разработени и внедрени софтуерни и хардуерни решения с теоретична и практическа значимост.

Структура на дисертацията

Дисертационният труд е структуриран в **пет** глави.

В **първа глава** е направен аналитичен обзор на теоретичната база, свързана с проблемната област на дисертацията. Тя включва кратко въведение, актуалност на темата, приложения, предизвикателства и съществуващи решения на научното изследване. Мотивирана е необходимостта от създаване и прилагане на нова методология за работа с хетерогенни данни, която надгражда и подобрява съществуващите към момента.

Във **втора глава** е представена систематизирана методология за обработване, моделиране и интеграция на хетерогенни данни, извлечени от IoT устройства. Изложена е обща концептуална схема на разработката. На следващ етап подробно е описана и изяснена теоретичната методологична рамка - поредица от стъпки, групирани в четири основни етапа. В процеса на определяне на етапите е извършен обзор и сравнителен анализ на съществуващите методи и подходи при определяне на конкретни случаи, за които тяхното прилагане е правилно.

В **трета глава** е описан процесът на разработване на архитектура на IoT платформа. Тя се състои от две самостоятелни системи - хардуерна и софтуерна, които общуват помежду си. В първият раздел на главата се представя хардуерна архитектурата и нов метод за комуникация между IoT устройства. Във втори раздел се представят архитектурни решения за софтуерна реализация на сървър, базиран на микросървиси и имплементация на потребителски уеб интерфейс. За изграждането на софтуерната система е използван нов подход за организация на услуги за интелигентна обработка и обмен на данни в IoT системата.

В **четвърта глава** е направена експериментална реализация и валидиране на разработената методология. Разгледани са и са решени два типа задачи - за класификационен и регресионен анализ. За целите на тяхното решаване са преминати всички стъпки, описани в методологията. Като резултат от извършения експеримент са получени валидирани модели за машинно обучение, които са готови за интегриране в реална среда.

В **пета глава** е представено практическо приложение на създадената система. Разгледани са нуждите и ползите от нейната употреба. Направен е сравнителен анализ между съществуващите на пазара системи. На база на този анализ е съставена сравнителна характеристика, обобщаваща полезността на съществуващите системи спрямо разработената IoT система в тази дисертация.

В **Заключението** е представено резюме на получените резултати от разработката. Определени са насоки за бъдещи изследвания и развитие. Представен е списък с научни публикации по темата и забелязани цитирания.

Дисертационният труд съдържа 166 страници, 55 фигури, 16 таблици и 175 литературни източника.

Глава 1. Анализ на състоянието на изследването

1.1 Индустрия 4.0

Индустрия 4.0 представлява интензивна информационна трансформация на традиционните производствени процеси в свързана среда от данни. Първите три индустриални революции се характеризират с технически иновации и имат за цел да подобрят ефективността на производството. Четвъртата индустриална революция надгражда предишните, като осигурява повсеместен достъп до интернет, комуникация между машини и усъвършенствано моделиране и обработване на големи масиви от данни чрез алгоритми за машинно обучение.

Индустрия 4.0 се характеризира с холистичен подход, състоящ се от анализ и оценка на процеса на трансформация, управление на приложенията, управление на данните, управление на активи и привеждането им в организационно съответствие. Главната цел е да се създаде обща база за оценяване при създаването на технологиите и насочване на компаниите към постигане на по-висок етап на зрялост, за да се максимизират икономическите ползи от Индустрия 4.0 (Gökalp E., 2017).

Кибер-физичните системи и “Интернет на нещата” (IoT) са в основата на тази трансформация и предоставят нови възможности за проектиране и прототипиране в области като разработване на продукти и услуги, дистанционно управление и диагностика, мониторинг на състоянието, проактивна и контролирана поддръжка, проследяване и планиране, приложения в реално време, енергия и екология, зелени съоръжения, иновации и др.

Индустрия 4.0 включва много иновативни елементи (Jazdi N., 2014), които са част от производствената технология, като:

• Интернет на нещата (Internet of Things - IoT) – свързани чрез интернет компютърни устройства, вградени в ежедневни обекти, които събират и изпращат данни за състоянието на обектите.

• Изкуствен интелект (Artificial intelligence - AI) – технология занимаваща се с изграждането на интелигентни машини, способни да изпълняват задачи, които обикновено изискват човешки интелект.

• Машинно обучение (Machine learning - ML) – технология занимаваща се с изграждането на математически модел въз основа на примери от данни, известни като „данни за обучение“, за да се идентифицират модели и да се вземат решения с минимална човешка намеса (Molnar, 2020).

- Машини към машини (Machine to Machine - M2M) – технология, която позволява на мрежовите устройства да обменят информация и да извършат действия без ръчна помощ от човек. Изкуственият интелект и машинното обучение улесняват комуникацията между системите, позволявайки им да правят своя самостоятелен избор.
- Големи данни (Big Data) – големи масиви от данни, които могат да бъдат анализирани чрез изчисления, за да разкрият модели, тенденции или асоциации (Pilloni V., 2018).

Също така, в Индустрия 4.0 е заложен силен акцент върху сигурността. Това не означава само сигурност на мрежите за данни и комуникации, но и защитата на данните (особено след приемането на Общия регламент за защита на данните и Регламента за неприкосновеност на личния живот и електронни комуникации - GDPR).

1.2 Интернет на нещата

Концепцията за “Интернет на нещата” (IoT) се базира върху идеята за осъществяване на постоянна връзка между физическия и дигиталения свят. Идея, която днес е вече възможна технологично. Използването на интернет в устройства прогресивно навлиза във всекидневието. Също така паралелното развитие на технологиите "Машина към машина" (M2M) и "Изкуствения интелект" (AI) разширяват областите си на приложение, както в бита, така и в бизнеса (Bertelle C., 2016).

“Интернет на нещата” може да се разглежда като единна мрежа, свързваща физически и виртуални обекти. Може да бъде описан сценарий, при който в голям брой обекти са вградени уникално идентифицируеми изчислителни устройства. Свързаността към интернет им позволява да събират, съхраняват, споделят и анализират данни и да бъдат управлявани дистанционно посредством други устройства с интернет връзка.

Употребата на интернет непрекъснато се разширява. Обектите с вградени изчислителни устройства стават разпознаваеми и интелигентни. Те могат да предприемат действия или да предлагат решения, съобразени с контекста им на имплементация. Също така имат способността да предоставят информация за себе си и да достъпват обобщена информация от други споделени “умни устройства”. Тези устройства могат да взаимодействат помежду си чрез безжични или кабелни комуникационни канали и уникални адресиращи схеми като IPv6. Целта на “Интернет

на нещата” е да създаде възможност обектите да бъдат свързани по всяко време и на всяко място.

"Нещата" в “Интернет на нещата” се състоят от съвкупност от разнообразие на хардуерни спецификации, комуникационни възможности и качества на услугите, което прави IoT разнороден по неговия характер. Тъй като "Нещата" могат да бъдат нещо от всеки нежив обект до всеки жив предмет като човек или животно, устройствата в тази технология могат да се различават по своите изчислителни възможности, разпределението и управлението на необходимата енергия, спецификациите на паметта, консумацията на енергия и управлението на сигурността и надеждността. Това определя необходимостта от разработване на оптимизационна структура с кръстосани слоеве, която може да осигури ефикасна оперативна съвместимост в тази хетерогенна мрежа. Такава структура с кръстосан слой въвежда концепцията за оперативна съвместимост между мрежовите слоеве без да променят оригиналните свойства на мрежовия слой (Deol Sh., 2016), което осигурява съвместимост с хетерогенните мрежи, въведени от технологията “Интернет на нещата”.

1.2.1 Технологии за IoT

Познаването на технологичните особености на една система ни позволява да разберем по-добре как тя функционира, нейните силни и слаби страни.

Технологичния стек на „Интернет на нещата“ може да бъде разделен на четири основни слоя – хардуер, софтуер, комуникации и платформа (фиг. 1.1):

Фигура 1.1: Технологичен стек на “Интернет на нещата”

Хардуер

Устройствата са обекти, които всъщност представляват „нещата“ в “Интернет на нещата”. Действайки като интерфейс между реалния и дигиталния свят, те могат да приемат различни размери, форми и нива на технологична сложност в зависимост от задачата, която се изисква да изпълнят в рамките на конкретното внедряване на IoT. Всяко устройство, което измерва и събира нужните данни може да се превърне в свързано такова чрез добавяне на необходимите хардуерни компоненти. Сензорите,

задвижващите механизми и други телеметрични съоръжения също могат да представляват самостоятелни умни устройства.

Софтуер

Софтуерът е ключовият елемент, който прави свързаните устройства „умни“. Той отговаря за осъществяването на комуникацията с „Облака“, събирането на данни, интегрирането на устройства, както и извършването на анализ на данните в реално време в мрежата на IoT. Чрез софтуер данните могат да бъдат визуализирани в таблични, графични или други удобни за потребителите формати.

Комуникации

Комуникационните механизми са силно свързани с хардуера и софтуера на устройствата. Комуникационният слой включва както решения за физическа свързаност (клетъчни, сателитни, LAN), така и специфични протоколи, използвани в различни IoT среди (ZigBee, Thread, Z-Wave, MQTT, LwM2M и др.). Изборът на съответното комуникационно решение е една от жизненоважните части при конструирането на всеки стек от IoT технологии. Избраната технология определя не само начините, по които данните се изпращат и получават от „Облака“, но и как устройствата се управляват и комуникират с други периферни такива.

Платформа

Благодарение на комбинацията между хардуер и инсталиран софтуер, устройството е в състояние да „усети“ какво се случва около него и да го съобщи на потребителя чрез определен комуникационен канал. Платформата е мястото, където всички тези данни се събират, управляват, обработват, анализират и представят по удобен за потребителите начин. По този начин, платформата е важна част от технологичния стек не просто поради възможностите й за събиране и управление на данни, но и поради способността й да анализира и да прави полезни изводи от събраните данни, предоставени от устройствата и комуникационния слой. Платформите могат да бъдат дистрибутирани с инсталации на място или на базата на „Облак“ (Glushkva T., 2019) (Gilchrist A., 2016).

1.3 Приложения на IoT

“Интернет на нещата” намира приложение в различни области като (Табл. 1.1):

Таблица 1.1 Области и приложения на IoT (Atanasova T., 2019)

Област	Приложение
Интелигентна градска среда	Умни градове, Умни сгради, Умно осветление, Мониторинг на културно поведение, Мониторинг на начина на живот, Обществена безопасност, Системи за наблюдение чрез дронове, Преносим личен цифров асистент;
Интелигентна енергийна мрежа	Интелигентна енергийна система, Умни консуматори, Умни измервателни уреди, Зелена енергия;
Умно здравеопазване	Мониторинг на здравословно състояние, Прогноза за здравословно състояние, Амбиентно подпомагане на живот, Неонатологични грижи;
Умен транспорт	Контрол на трафика, Умно маршрутизиране, Умно паркиране, Разпознаване на пешеходци, Умни превозни средства, Автономни превозни средства;
Умен дом	Умни домашни уреди, Умни мебели, Умни термостати, Умно контакти, Умни ключалки, Умни домашни охранителни системи;
Умно земеделие	Прецизно земеделие, Умен контрол на вредителите, Умно управление на добитък, Умни оранжерии и обори, Умен мониторинг на складови съоръжения;

1.3.1 Интелигентна градска среда

Наблюдава се тенденция през последните години на значително мигриране на населението от села към градове, което от своя страна води до пренаселване на едни територии и обезлюдяване на други. Градовете биват изправени пред различни предизвикателства, включително създаване на работни места, икономически растеж, устойчивост на околната среда и на социалното положение (Atanasova T., 2016). Поради тези причини е от съществено значение изграждането на план за правилно разпределение и управление на ресурсите, което налага тенденция за изграждане на технологична архитектура в градовете с цел ефективна взаимосвързаност между хора и машини (Popchev I., 2019) (Talari S., 2017).

За изпълнението на тези цели технологията трябва да може да борави с милиони устройства и сензори, хиляди сървъри, обработка и предаване на големи обеми от данни. Също така е необходимо инфраструктурата да може да обработва данните, да разбира контекста и да взема решения за възможно най-кратък срок от време. Това налага употребата на „*fog computing*“ или „*IoT Edge*“ (Mostafavi S., 2019), чрез който се улеснява работата и ефективността на изчисление, съхранение и мрежови услуги между крайните устройства и изчислителните облаци (центрове за данни). Тази услуга позволява по-бърза обработка на данните в сравнение с времето за изпращане и обработване на данните в „Облака“.

1.3.2 Интелигентна енергийна мрежа

Интелигентната енергийна мрежа е електрическата мрежа (Ramamuthy A., 2017), която включва разнообразие от оперативни и енергийни мерки в рамките на мрежата, от производството на електроенергията през доставката и транспорта до консумацията. Тя съдържа свързани чрез Интернет наблюдаващи комуникационни и компютърни устройства като интелигентни измервателни уреди, интелигентни разпределителни уреди, възобновяеми енергийни източници, както и енергийно ефективни ресурси. Чрез тях се извършва наблюдение на консумацията на енергия в сгради в реално време и предоставяне на информация за оптимизиране на производството и снабдяването с електроенергия. Също така може да се извърши настройване на предлагането спрямо нуждата на системата за автоматизация на мощността, която позволява диагностика и поддръжка на електрическата мрежа при смущения или прекъсване на електрозахранването.

1.3.3 Умно здравеопазване

Електронното здравеопазване обхваща широка област от здравното обслужване, което се управлява чрез информационни и комуникационни технологии (Atanasov J., 2011) (Mezghani E., 2017). IoT приложенията спомагат развитието на платформи за прилагане на системи за интелигентно електронно здравеопазване, които предлагат услуги в областите на подпомагане извършването на ежедневни дейности, мониторинг на активността, повишаване на безопасността и сигурността, получаване на достъп до медицинска помощ и системи за спешни случаи, улесняване на бързото диагностициране на здравното състояние на пациентите и ранно предупреждение за появата на здравен проблем.

1.3.4 Умен транспорт

Умният транспорт представлява система, при която инфраструктурата (Janssen M., 2019) и превозните средства са оборудвани с цифрови сензори и комуникационни компютърни устройства, които позволяват да бъдат събирани и обменяни данни между превозните средства и инфраструктурата и обратно, също така и от превозно средство към превозно средство.

Целта на създаването на умен транспорт и умна инфраструктура е да се оптимизира пътния трафик, ефективно да бъдат използвани ресурсите, повишаване на безопасността и надзор на правоприлагането.

1.3.5 Умен дом

Умният дом е този, който предлага на собствениците си комфорт, сигурност, енергийна ефективност и удобство по всяко време, независимо от местоположението на притежателите си.

Умен дом е термин, който често се използва за определяне на дом, в който има уреди, осветление, отопление, телевизори, компютри, аудио и видео системи, системи за сигурност и други домашни уреди, които са способни да комуникират един с друг и могат да бъдат контролирани дистанционно от всяко място в дома и извън него чрез интернет (Harper R., 2006).

1.3.6 Умно земеделие

Умното земеделие представлява използване на M2M и IoT технологии с цел увеличаване производителността чрез модерни средства по непрекъснато устойчив начин, за да се постигне най-доброто от гледна точка на качеството, количеството и финансова възвръщаемост, както и да се гарантира, че процеса е екологично съобразен (Sowmya J., 2019). Сензори, вградени в селскостопански машини, в почвата или прикрепени към селскостопански животни, следят определени параметри и изпращат данните за съхранение и обработка. Това се извършва с цел оптимизиране на процесите при вземане на решения с помощта на софтуер за машинно обучение, който може да анализира големите количества данни и да оптимизира процесите в реално време.

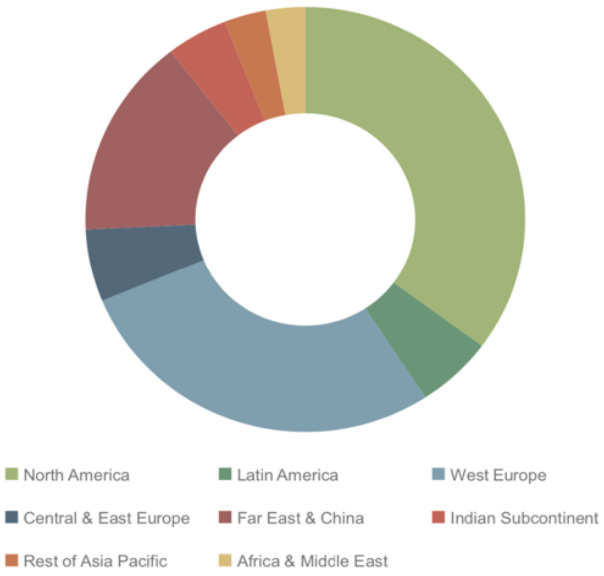
Някои от основните области на приложение на интелигентното земеделие (Gallai N., 2009) могат да бъдат структурирани по следния начин:

- Прецизно земеделие;
- Животновъдство;
- Контрол на вредителите;
- Управление на селскостопанската техника;
- Мониторинг на добитък;
- Закрито земеделие – оранжерии и обори;
- Рибовъдство;
- Горско стопанство;
- Мониторинг на складови съоръжения – резервоари за вода, резервоари за гориво и др.

"Интернет на нещата" (IoT) намира все по-голямо приложение в екологичното бъдеще на планетата (Borcard D., 2011), например в областта на защитата на застрашените биологични видове.

1.4 Предизвикателства пред IoT

Благодарение на разширяването на мрежите и по-доброто разбиране на стратегиите за внедряване на IoT се наблюдава ускорен растеж на свързаните устройства (Sharma A., 2020).



Фигура 1.2: Свързани IoT устройства през 2022, разделени по региони.
Прогнозен брой свързани устройства – 51 милиарда.
(Източник: Juniper Research)

Според Juniper Research (Sorrell S., 2018) броят на свързаните с IoT устройства ще наброява 38,5 милиарда до края на 2020 г. и 50 милиарда до 2023 г. (фиг. 1.2), спрямо 21 милиарда през 2018 г. и 13,4 милиарда през 2015 г., като Северна Америка, Западна Европа, Далечния Изток и Китай остават основни пазари за потребителски IoT устройства за целия прогнозен период (Barker S., 2020).

Повишеният интерес към внедряването на успешни IoT решения налага създаване на задълбочени анализи относно областите, където тези решения могат да бъдат компрометирани. Резултатите от тези анализи определят и предизвикателствата, пред които IoT индустрията е изправена.

На база извършени анализи от Juniper Research (Sorrell S., 2018), към този момент, като най-големи предизвикателства пред успешното внедряване на IoT

устройства се очертават свързаност, скорост, съвместимост, големи хетерогенни данни, интелигентен анализ и сигурност.

Свързаност

Поддържането на устойчива, висикоскоростна свързаност с отлична пропускателна способност за големи обеми от данни между IoT устройства се счита като едно от най-големите предизвикателства пред IoT (Zikria B., 2019).

В днешни дни бизнесът разчита основно на централизирана сървърна система за удостоверяване, упълномощаване и свързване на различни възли в мрежа. Този модел е достатъчен за сегашните IoT екосистеми, но когато мрежите се разраснат, централизираните системи ще изпаднат в затруднение. Такива системи се нуждаят от големи инвестиции и разходи за поддържането им, за да се осигури устойчивост (непрекъсваемост) на услугата. В противен случай сървъри, които не могат да се справят с толкова големи количества обмен на данни, биха направили цели мрежи или част от тях недостъпни. Бъдещето на IoT зависи от създаване на децентрализирани IoT мрежи.

Това може да стане възможно чрез преместване на някои от задачите към IoT Edge, където се дава възможност на интелигентни сървърни услуги като IoT хъбовете да поемат отговорността за критичните операции, а облачните сървъри отговарят за събирането на данни и аналитичните операции (Varga E., 2018).

Скорост

За много области на приложение на IoT, скоростта е от съществена важност за целевата им дейност. Зависимостта на финансовия сектор от алгоритми за търговия в реално време, например, означава, че забавяне само от няколко милисекунди може да има скъпи последици, измерващи се в огромни финансови загуби. В здравната индустрия загубата на части от секундата дори е въпрос на живот или смърт. А за бизнеса, който предоставя услуги, базирани на данни на клиентите, изоставащите скорости могат да доведат до загуби и неконсистентност на данните и да причинят щети в дългосрочен аспект. Скоростта вече не е само конкурентно предимство - тя вече е най-добрата практика. Именно поради това е необходимо разработването, имплементирането и внедряването на технологии от ново поколение като 5G (Pozza M., 2020), която значително надгражда и подобрява съществуващата 4G технология. За нейни основни предимства могат да бъдат посочени значително по-високата скорост в предаването на данни, по-ниска латентност, по-голям капацитет на отдалечено изпълнение, възможност за по-голям брой свързани устройства, внедряване на

виртуални мрежи, осигуряващи по-добра адаптивна свързаност спрямо дефинирани конкретни нужди.

Скоростта на предаване на данните чрез 5G е в диапазон 15-20 Gbps с латентност <10 милисекунди и средна скорост 200-400 Mbps, за разлика от 4G мрежа, поддържаща 200 Mbps с латентност 20-30 милисекунди и средна скорост 25 Mbps.

Чрез повишената употреба на облачни услуги, всички устройства ще зависят по-малко от вътрешната си памет и от натрупването на данни, това ще доведе и до намаляване на необходимостта от инсталиране на голям брой процесори на някои обекти, тъй като изчисленията могат да се извършват в облака.

С 5G броят на свързаните устройства към мрежата може да се увеличи до мащаб от милион устройства на квадратен километър. Всички свързани устройства ще могат да обменят информация в реално време през интернет (благодарение на IoT). Също така чрез употребата на 5G се позволява реализацията на виртуални мрежи и създаването на подмрежи, което дава специфични характеристики на част от мрежата чрез програмиране и позволява да се даде приоритет на връзките, каквито биха могли да бъдат аварийните ситуации пред други потребители, по този начин се избягват възможни претоварвания на мобилната мрежа (Kranenburg R., 2012).

Съвместимост

Наличието на множество от производители на устройства прави трудно спазването на общи стандарти. Някои производители са загрижени повече за себестойността на устройствата отколкото за тяхната съвместимост с устройства на други производители. Разрастването на свързаните IoT устройства и липсата на единен технологичен стандарт, както за софтуера, така и за хардуера създава трудности за бъдеща поддръжка на цялостната система поради несъвместимост между компонентите, M2M протоколите, различията във фърмуера и операционните системи, които те използват. Много често това прави комуникацията с облачните услуги невъзможна или силно затруднена и ограничена.

Технологичният прогрес бързо въвежда нови технологии и намалява или спира поддръжката на други стари такива, което води до етап, в който ефективността и съвместимостта на системите, използващи по-стари технологии ще намалее значително за разлика от системите, въвели технологични новости. Този проблем произтича от липсата на стандарти, при които да бъде регламентиран и задължително въведен период от време, през който производителите да бъдат задължени да поддържат разработките си (Banafa A., 2017).

Сигурност

Данните, получени от IoT сензори и контролери, са достатъчно ценни, за да имат силата да променят цялостните бизнес модели и поведение на цели индустрии в дадена икономика. За да бъде изградена защита на IoT система е необходимо да се знае какво представлява тя - каква архитектура се използва (Firouzi F., 2020), как работи, какви видове компоненти и части има, използвани протоколи, основни области на приложение и др. Познаването на тези зависимости, както и силните и слабите им страни може да изгради пълна и точна представа за видовете атакуващи вектори, към които IoT системите са уязвими.

Понастоящем няма стандартна методология за моделиране и справяне с реални сложни сценарии с използване на "Интернет на нещата". Неразрешените проблеми, които са налице в системите, в голяма степен предопределят основните заплахи (Hameed S., 2019), на които са поставени системите за "Интернет на нещата". Предизвикателствата по отношение на аспектите на сигурността при проектиране и използване на IoT системите са:

- *Поддръжка на фърмуер и актуализации на ОС* - IoT устройствата имат слабости в своя софтуер и фърмуер, проявяващи се най-вече в опциите им за обновяване. IoT машините все още са прекалено доверчиви и лесно биха допуснали злонамерена атака от разстояние през опцията си за системно актуализиране. Също така техният хардуер се нуждае от периодична профилактика, поддръжка, мониторинг и диагностика на сензорите, което задължава IoT устройствата да бъдат разположени на физически достъпни места, което ги прави лесна мишена.
- *Проблеми с паметта на устройствата* - Повечето устройства за "Интернет на нещата" страдат от недостиг на памет. Те имат достатъчно памет, само за да работят. Често в такива устройства се съхраняват ключове за криптиране и идентификационни данни в паметта.
- *Имплицитно доверие* - По подразбиране всички компоненти в една IoT система имат взаимно доверие. Ако някой от компонентите е компрометиран, то цялата система също може лесно да се контролира. Този тип имплицитно доверие е огромна уязвимост.
- *Несигурен уеб интерфейс* - Уеб интерфейсите на голяма част от устройствата служат за входна точка за злонамерени атаки. Веднъж свързани устройствата с локалната мрежа, те се превръщат в удобна точка за неоторизиран достъп. Сред най-честите уязвимости на уеб интерфейсите са: Cross Site Scripting (XSS), Cross Site

Request Forgery (CSRF), SQL инжектиране, качване (upload) на външни файлове, недобър контрол на потребителските сесии, пропуски в конфигурирането и др.

- *Липса на криптиране* - IoT устройствата често не разполагат с функция за криптиране на данните при транспортиране в локална мрежа или по Интернет (Morville P., 2006). Все още не е изготвен единен стандарт за защита на "Интернет на нещата" и поверителната информация протича от точка А до точка Б с недостатъчно ниво на сигурност и може да стане достъпна за четене или да не пристигне на точното място.
- *Физическа сигурност* - Повечето устройства са разположени на отдалечени места и могат да бъдат сравнително лесно достъпни. Всеки, който има някакъв физически достъп до мрежата (Dominguez S., 2016) може да използва лесни точки като USB портове, четци за карти памет и др. Така се достига до системни файлове на операционната система или други съхранявани данни.
- *Несигурни мобилни приложения* - Много приложения не изискват пароли за стартиране на приложението, отсъства 2FA (Two Factor Authentication - Аутентикация с два фактора).

Сигурността на IoT устройствата не трябва да бъде пренебрегвана дори и ако за тях се знае, че събират нечувствителни данни. Всяко едно от тези устройства може да бъде използвано, посредством което да бъде достигната реалната цел.

Данни

"Интернет на нещата" представлява един от най-големите източници на нови данни. Разнообразието им е също толкова впечатляващо - IoT устройствата събират хетерогенни данни, вариращи от аудио и видео до сензорни данни. Тяхната поява въвежда термина „Големи данни" (Suchetha K., 2015), поради големия си обем и разнообразие от формати, които са по-тежки от познатите досега текстови такива. Непосредствената близост на IoT устройствата до потребителя дава възможност за събиране на данни през определени интервали от време, което ги прави подходящи за обработка и моделиране като времеви серии.

Съществуват няколко аспекта на данните от IoT устройства, които затрудняват процеса по обработването им.

Събраните данните са зашумени поради грешки, възникващи по време на придобиване и предаване. По своята същност тези данните са силно променливи, както поради огромното несъответствие в потока от данни, генериран от разнородни компоненти, така и поради съществуването на различни времеви модели. Ползата от данните силно зависи от честотата, при която те се улавят и начина, по който се

обработват. В случаите, когато данните идват от еднакви устройства, преди да се считат за надеждни, е необходимо да се отчете факта, че тези устройства могат да се държат различно, дори при сходни условия. Превръщането на тези неструктурирани потоци от данни в структурирани такива и отделянето на полезния сигнал от шума са едни от най-важните стъпки при процеса на обработка.

Друг аспект, които затруднява обработката на данни получени от IoT устройства е тяхната „грубост“ - събраните данни чрез различни сложни сензори са в суров и често небалансиран вид (Mojgan G., 2017), което налага да се извърши задължителна предварителна обработка преди да започне процеса по извличане на бизнес стойност от тях.

При критично-важни системи работещи в реално време, от не по-малко значение е бързото разчитане на данните, за да се установи, какво стои зад тях и да се оцени ситуацията точно, правилно и своевременно.

В резултат на това действията по структуриране, почистване и валидиране на данните представляват критичен етап в процеса по обучение и прилагане на алгоритми за машинно обучение, чрез които да се разкрият неявни корелации в наборите от данни, които да се отразяват в прогнозни модели (Valdés J., 2018). Частично или грешно обработените данни водят до фалшиво позитивни или фалшиво негативни резултати след прилагане на алгоритми за обучение, което от своя страна води до понижена или нулева бизнес стойност и предприемане на подвеждащи неправилни решения.

1.5 Хетерогенни данни

Хетерогенните данни се характеризират с голямо разнообразие по типове стойности и формати. Те често са нееднозначни и са с ниско качество поради наличието на шум и/или липсващи стойности. Интегрирането на такива разнородни данни е труден и сложен процес. Затова в този си вид те не носят голяма полза за нуждите на бизнес приложенията. IoT системите са най-големият генератор на хетерогенни данни (Wang L., 2017).

Специфики

Хетерогенните данни имат следните четири специфики:

- Те са разнородни - придобитите данни са различни по вид и тип поради голямото разнообразие от устройства, които ги събират и липсата на ясно изградени конвенции за нужния вид и формат, който данните трябва да притежават.

- Те са в голям обем - необходимо е да се съхраняват за определен период от време не само получените към момента данни, но и историческите данни. Това води до непрекъснато нарастване на обема данни, с който се работи в момента.
- Има силна зависимост между времето и локацията - всяко устройство за събиране на данни се поставя на определено географско местоположение и всяка част от данните има времеви идентификатор. Комбинацията на времеви и геолокационни идентификатори е важно свойство на данните от IoT системи.
- Полезните и смислени данни представляват само малка част от общия обем генерирани данни. Поради спецификата на работа на IoT системите съществува голяма вероятност данните да бъдат зашумени по време на процеса събиране и предаване. Само малко количество от общия обем събрани данни носят реална полза.

Видове хетерогенност

Съществуват няколко вида хетерогенност на данните:

- Синтактична хетерогенност - възниква, когато два източника на данни не са изразени на един и същ език.
- Концептуалната хетерогенност - известна още като семантична хетерогенност или логическо несъответствие и обозначава разликите при моделирането на една и съща област, която представлява интерес.
- Терминологичната хетерогенност - вариации в имената, когато се отнасят до едни и същи единици от различни източници на данни.
- Семиотичната хетерогенност - известна още като прагматична хетерогенност и означава различно тълкуване.

Нива на представяне на хетерогенни данни (data representation):

Представянето на данни (Filip I., 2019) може да бъде описано на четири нива:

Ниво 1: Получаване на разнообразни сурови данни от различни източници.

Ниво 2: Унифициране на данните - хетерогенните данни трябва да бъдат унифицирани, защото твърде големият обем на данни може да доведе до високи когнитивни възможности и разходи за обработката им. Този слой преобразува отделните атрибути в информация по отношение на „какво-кога-къде“.

Ниво 3: Агрегиране. Пространствените данни могат да бъдат естествено представени под формата на пространствени решетки с тематични атрибути. Операторите за обработка са сегментиране и агрегиране. Агрегацията подпомага лесната визуализация и осигурява интуитивна заявка.

Ниво 4: Детекция и категоризация на събития. Събития възникващи на място се откриват чрез анализ на относителни времеви характеристики, които са резултат от определени операции. Категоризацията е последната стъпка при детекция на събитията, като за класификация се използват вече известните свойства, които са характерни за конкретната област на приложение.

Математически хетерогенен домейн от данни се дефинира като декартово произведение на колекция от източници:

$$H^n=\Psi_1\times...\times\Psi_n,$$

където n> 0 е броят на източниците на информация, които трябва да се вземат предвид.

Налице е тенденция за въвеждане на методи от Изкуствения интелект и Машинно обучение в различните разработки на IoT. Важно е да се намерят нови методи и интелигентни изчислителни техники, за да се повиши ефективността на данните, предоставени от устройства с интернет връзка.

Предизвикателства при интегриране на хетерогенни данни

Хетерогенността е една от основните характеристики на данните придобити от IoT системи и са основна причина за проблемите при тяхното интегриране и анализиране.

Стандартизирането на тези данни е предпоставка за извършване на ефективен и точен анализ и получаване на достоверни резултати. Липсата на подходяща стратегия за интеграция и обработка води до специфични проблеми, които възпрепятстват производителността и забавят резултатите.

Агрегирането и консолидирането на данни от различни структурирани, полуструктурирани и неструктурирани източници е сложна задача. Според проучване на Gartner (Thoo E., 2019), интегрирането на хетерогенни данни от множество източници е едно от най-големи предизвикателства.

Познаването на възможните предизвикателствата по време на този процес, подпомага тяхното успешно противодействие.

1. Извличане на данни – извличането на изходни данни е първата стъпка в процеса на интеграция. Това е сложен и времеемък процес, когато източниците на данни са в различни формати, структури и видове. Необходимостта от тяхното трансформиране цели изграждане на съвместимост с целевата система преди тяхното интегриране.

2. Цялост на данните – качеството на данните е основна грижа във всяка стратегия за интегриране на данни. Лошото качество на данните създава проблеми, които влияят на целия интеграционен цикъл. Обработката на невалидни, неправилни или небалансирани данни води до дефектни анализи, които биват предавани по веригата, което води до грешни резултати. Затова е налице необходимостта от създаване на план за управление качеството на данните.

3. Скалируемост – хетерогенността на данните води до приток на данни от различни източници влизащи в единна система, което резултира до експоненциално нарастване на обема на събраните данни. Това обуславя нужда от стабилно интеграционно решение, което не бива да компрометира производителността на системите.

1.6 Съществуващи решения за интегриране на хетерогенни данни

Ежедневно IoT системите генерират огромни количества данни. Всеки поток от данни, генериран от различните IoT системи, може да бъде в различен формат. Всеки формат е проектиран по някаква причина. Всеки от тях представлява информация с уникални атрибути, метаданни, структура и схема. Интегрирането на данни от различни формати добавя нуждата от наличието на различни нива на компетентност и това прави процесът на интеграция на информационни потоци от съществено значение за научните, техническите и бизнес практики свързани с работа с данни.

IBM определя интеграцията на данни като „комбинация от технически и бизнес процеси, използвани за комбиниране на данни от различни източници в смислена и ценна информация“ (Locht A., 2013). По същество нуждата от правилно организиран процес на интеграция на информационни потоци е създаването на единна, унифицирана структура от данни, която да притежава характеристика да се създават полезни изводи въз основа на целостта и хомогенността на данните, независимо от техния оригиналния източник или формат.

Интеграцията на данни поетапно развива своите подходи от исторически използваните ръчни и автоматизирани методи за интеграция до ясни, подредени научни, технически и бизнес процеси като:

ETL (Extract Transform Load) – процес за интеграция на данни, който се отнася до три основни стъпки – извличане, преобразуване и зареждане, използвани за агрегиране на данни от множество източници. Често се използва за изграждане на база от данни. По време на този процес данните се извличат от изходна система,

преобразуват се във формат, който може да бъде анализиран и се съхраняват в база от данни или в друга система. ETL е предназначен да насочи обработката на данни надолу към базата данни за подобрена производителност (Atwal H., 2020).
Използва се за традиционни решения насочени към трансформация на данни преди тяхното записване в база данни.

ELT (Extract Load Transform) – модерна алтернатива на ETL. Процес за интеграция на данни в три стъпки - извличане, зареждане и преобразуване. В този процес след извличане на данните, веднага се стартира фазата на зареждане – премества всички данни в едно централизирано хранилище на данни (Hadoop).
ELT е много по-гъвкав от ETL, като позволява на потребителите да изпълняват нови трансформации, да тестват и подобряват заявки директно върху суровите данни, като това спестява времето за зареждане, трансформация и поддръжка, които ETL причинява (Anoshin D., 2020). Използва се за облачни изчисления на големи данни.

CDC (Change Data Capture) е процес за интегриране на данни, който се основава на идентифициране, улавяне и предоставяне на промените, направени в източниците на данни. Чрез този подход е необходима обработка само на променените данни преди тяхното съхранение, а не на целия поток, което води до минимизиране на ресурсите, необходими на ETL процеса. Целта на CDC е да осигури синхронност и преносът на данни да се сведе до минимум от източника до базата (Singh J., 2019).
Използва се за разпределено съхранение и обработка на големи данни.

OSEMN (Obtain Scrub Explore Model iNterpret) – стандартизиран и широко приет процес на организация на научноизследователската дейност в областта на Data Science за интеграция на данни. Процесът се състои от пет основни стъпки – извличане, почистване, изследване, моделиране и интерпретиране. Базиран на принципите на ETL процеса. Той допълва съществуващите методи, използвайки инструменти за машинно обучение и подходи за интерпретация на получените резултати (Guhr S., 2019). Използва се за подготовка и моделиране на данните преди тяхното записване в база данни.

Съществуващите процеси за събиране, обработка и използване на данни са проектирани да изпълняват определен вид стандартни задачи. Независимо от постигнатите успехи през последните години, тези процеси не са гъвкави и не могат да бъдат приспособени към изпълнение на комбинирани задачи, които по същество се различават от стандартните. За да бъде изпълнен целият цикъл на едно приложение от добиване на данни до превръщането им в информация е необходимо да бъдат

комбинирани няколко процеса на работа. Това води до използване на сложни архитектури и трудно проследими етапи на работа, което затруднява и забавя работния процес и последващата поддръжка.

Налице е нуждата от надграждане на познатите процеси за интегриране на данни и насочване на усилията към въвеждане на гъвкави стандартизирани процеси и методологии на работа, включващи все повече типове данни, формати и обеми и в същото време да бъдат лесни за имплементиране и възпроизвеждане. Те трябва да обхващат целия път от придобиване на данни, през обработване, анализиране и моделиране до визуализиране и извличане на бизнес стойност.

1.7 Изводи

В резултат на направения аналитичен обзор могат да бъдат изведени следните заключения:

1. Количеството събирани данни от IoT устройства непрекъснато нараства и достига до безпрецедентни нива. Това води до промени в бизнеса, социалните взаимодействия и бъдещето на обществото.
2. Липсата на стандартна референтна архитектура за моделиране на IoT системи затруднява изграждането на общ подход за обработка на генерираните хетерогенни данни.
3. Липсата на ясно структурирана методология за обработка и интегриране на хетерогенни данни прави процесите по трансформацията им хаотични и трудни за повторно възпроизвеждане. При възникване на грешки по време на работа е необходимо процесът да започне отначало, което затруднява и забавя цялостния процес по получаване на полезни знания от извлечените данни от IoT устройства.
4. Обществото е изправено пред предизвикателството да успее да извлече максимална полза от всички технологични новости, като същевременно е необходимо да се сведат до минимум рисковете, пред които е поставено. Сигурността е важен фактор, който не бива да бъде оставен на втори план.

На база направените изводи може да се обобщи, че при съществуващите проблеми за работа с данни от IoT системи е важно да бъдат взети навременни решения чрез създаване и използване на единен и систематизиран подход, което ще доведе до устойчиви, надеждни и повторно възпроизводими резултати.

1.8 Цел и задачи на дисертационния труд

От направения анализ на състоянието на изследването е формулирана целта на дисертационния труд:

Да се предложи система и инструменти за интегриране на хетерогенни данни от разпределени IoT устройства, които да позволяват тяхната обработка, моделиране и интеграция.

За тази цел се дефинират следните задачи:

1. Да се предложи методология за обработка, моделиране и интеграция на хетерогенни данни от разпределени IoT устройства.
2. Да се предложи архитектура и метод за комуникация на модулна IoT хардуерна система.
3. Да се предложи архитектура на софтуерна платформа и подход за организация на услугите за интелигентна обработка на хетерогенни данни от IoT система.
4. Да се създадат валидни модели за машинно обучение за експериментално потвърждение на разработената методология.
5. Да се покаже възможно приложение на IoT системата и инструментите за интегриране на хетерогенни данни от разпределени устройства в интелигентното земеделие.

Глава 2. Методология за обработка, моделиране и интеграция на хетерогенни данни

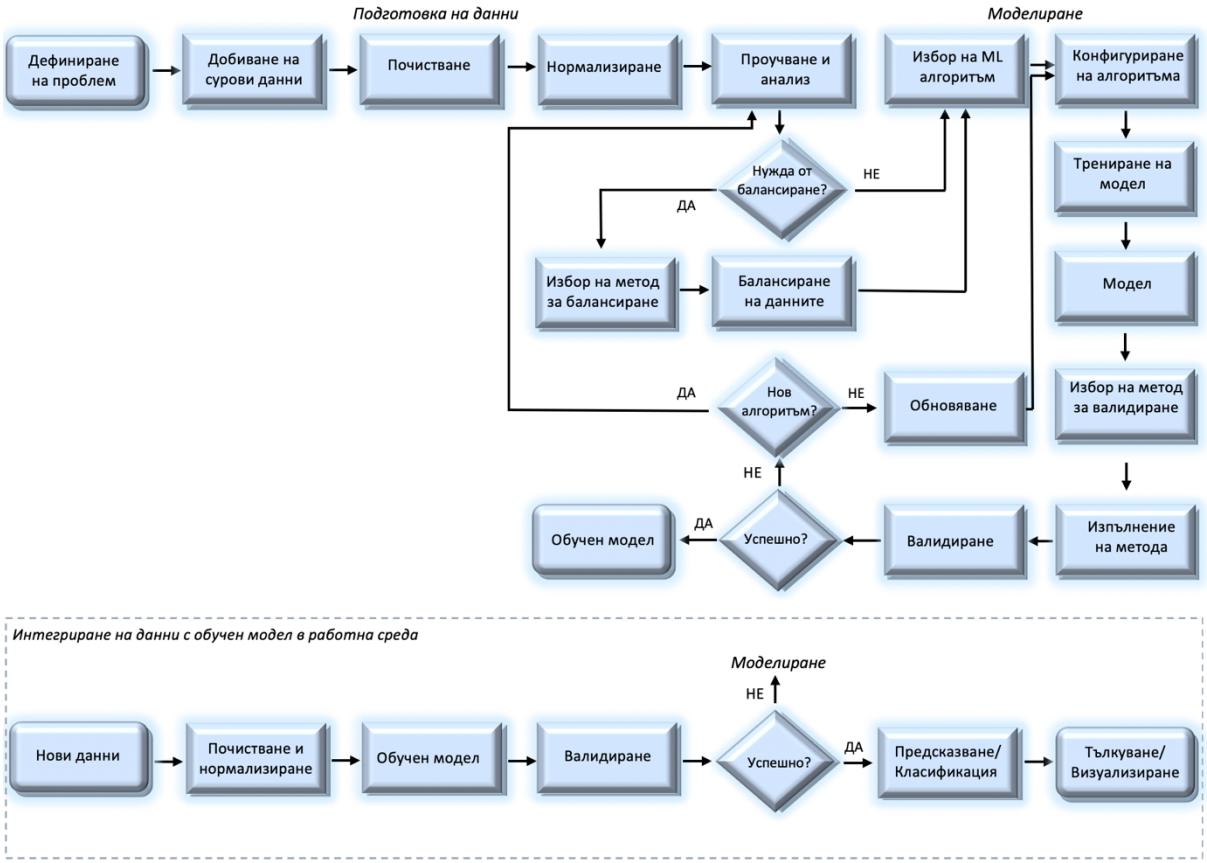
Методологията описва "общата изследователска стратегия, която очертава начина, по който трябва да се извършват научни изследвания" (Howell K., 2013). По своята същност тя представлява система или група от методи, правила и твърдения използвани в конкретна дисциплина.

В тази глава на дисертацията е предложена методология (Задача 1), която комбинира подходи, методи, процеси, инструменти и добри практики, за да се отговори на въпроса „Как да се обработят, моделират и интегрират хетерогенни данни от разпределени IoT устройства”. Тя е приложима за всички IoT системи събиращи хетерогенни данни, които е нужно да бъдат съхранени, обработени и визуализирани.

На Фиг. 2.1 е представена обща концептуална схема на методология за работа с хетерогенни данни, придобити от интегрирани IoT устройства. Тя се състои от четири основни последователни взаимосвързани етапа, които описват целия цикъл на работа с хетерогенни данни - от възникване на нужда от тяхното придобиване, през обработка и моделиране, до тяхната интеграция в работна среда.

За да се добие задълбочена представа за разработената методология, на Фиг. 2.2 е представена подробна схема, в която са представени основните етапи и стъпките включени в тях.

Фигура 2.1 Обща концептуална схема на методология за обработка, моделиране и интеграция на хетерогенни данни



Фигура 2.2 Подробна схема на методология за обработка, моделиране и интеграция на хетерогенни данни

Представената методология се състои от следните етапи и придружаващи ги стъпки:

Етап 1: Дефиниране на проблем

Етап 2: Подготовка на данни:

- Добиване на хетерогенни данни от IoT устройства;
- Почистване на придобитите данни;
- Трансформация на хетерогенните данни;
- Проучване и анализ на състоянието на данните;

Етап 3: Моделиране:

- Избор на алгоритъм за машинно обучение;
- Конфигуриране на избрания алгоритъм;
- Трениране и валидиране на модела;

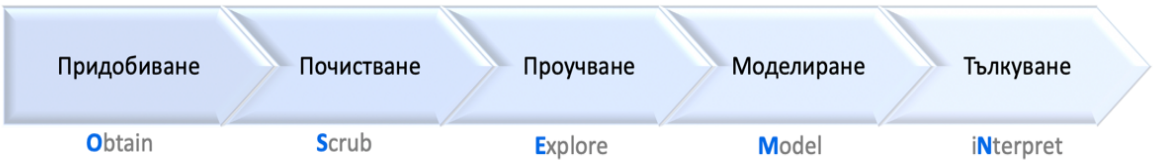
Етап 4: Интегриране на данни с обучен модел в работна среда:

- Добиване на нови данни;
- Почистване и трансформация на придобитите данни;

- Използване на обучения модел;
- Валидиране на резултатите;
- Тълкуване и визуализация;

За постигане на оптимални резултати от прилагането на тази методология е важно да бъдат следвани и изпълнени етапите и отделните стъпки в тях по описания начин. При пропускане или размяна на стъпките се крие риск от получаване на отклонение в резултатите при използване на едни и същи данни. Също така това ще доведе до невъзможност за повторно възпроизвеждане на резултатите от други изследователи.

При разработката на етапите в методологията е използван и надграден стандартизиран процес за обработка на данни - OSEMN (фиг. 2.3), който е подробно разгледан в (Dineva K., 2018). Този процес е съчетан с други методи, подходи и инструменти, които са характерни за обработката на хетерогенни данни и са неизменна част от един цялостен процес.



Фигура 2.3 OSEMN процес

Разработената методология е съчетание от очаквана функционалност и лекота за имплементиране. Ориентацията към използване на стандартизиран процес като основа е философия на работа, при която се преминава от фокусиране върху дейностите към фокусиране върху резултатите, защото дейностите се разглеждат в своята координирана съвкупност при създаването на стойност за крайния резултат (намиране на корелация между резултатите от анализираните данни и случващите се събития за предвиждане на бъдещи такива). Разбираемите и логически последователни стъпки на методологията, обогатени с придружаващи инструкции, осигуряват изпълнението на дейностите и постигане на възпроизводимост на резултатите от различните участници в него. Постига се по-висока зрялост на стандартизиране и хармонизиране на практиките в различните етапи (Dineva K., 2019). Избягват се грешки поради неконсистентност и недостатъчна информираност.

Една от най-важните характеристики на методологията е, че тя описва недвусмислено не само последователността, но и отговорностите. На всеки етап от

процеса е ясно какво се очаква от него, от кого той ще получи заявка за определена дейност и на кого следва да предостави резултата. Значително се улеснява идентифицирането на съществуващите добри практики. Постигането на по-голяма откритост на дейностите на всеки етап от процеса предоставя възможност за лесно откриване на допуснати грешки и бързо връщане към определен етап от процеса за отстраняването им.

2.1 Подготовка на данни

2.1.1 Придобиване на данни

Събирането на данни е първата стъпка от целия работен процес. Тя е определяща за крайния успех на зададената цел. Това е процес на събиране и измерване на данни за желани променливи по установен систематичен начин (Richard W., 2020), който позволява да се отговори на запитвания и заявени изследователски въпроси, за да се тестват хипотези и да се оценят резултатите.

2.1.2 Почистване на данни

Преди да бъдат обработени и анализирани данните (Yaghini M., 2010) получени от IoT устройства е необходимо да бъдат извършени няколко действия: сливане на отделните колони с данни в единна таблица, почистване на данните от невалидни стойности, нормализация на данните и обработка на екстремните стойности.

Сливане на отделните колони с данни в единна таблица

Събраните хетерогенни данни за различни параметри е необходимо да бъдат обединени в една обща таблица, а параметрите да останат, като променливи по колони.

Почистване на данните от невалидни стойности

При събиране на данни от реалния свят, често се наблюдават различни причини, поради които има данни със стойност Null, NaN или NA. При този проблем стойността, от която се нуждае анализът, по някаква причина не е известна, в резултат на което анализът може да бъде неправилен, а съставените модели за данни да бъдат некоректни, защото повече алгоритми за прогнозиране не могат да работят с липсващи данни. Често използвани подходи за решаване на този проблем са липсващите данни да бъдат заместени със средни стойности, с най-често срещани стойности или тези стойности да бъдат директно премахнати. Ако масивът от данни е подреден, тогава липсващите данни могат да бъдат заместени със следващата най-близка стойност във възходящ или низходящ ред.

Най-често използваните методи за работа с липсващите данни (Li T., 2019), (Meghanadh B., 2019) са представени в таблица 2.1.

Таблица 2.1: Анализ на методи за работа на липсващи данни

Метод	Описание
Многовариантно дописване с помощта на верижно уравнение MICE (Multivariate Imputation by Chained Equations)	За всяка липсваща стойност се задава нова стойност, която се изчислява по метода MICE, при който за всяка променлива с липсващи данни се моделира условие, в което се използват данни от други променливи, преди да се попълнят липсващите стойности.
Персонализирана стойност на заместване (Custom Substitution Value)	Посочва се стойност, която служи за заместител (като 0), която се прилага за всички липсващи стойности. Посочената стойност трябва да бъде съвместима с типа данни на колоната.
Замяна със средна стойност (Replace with Mean)	Изчислява средната стойност на колоната, която се използва като заместваща стойност за всяка липсваща стойност в колоната.
Замяна с най-често срещаната стойност (Replace with Mode)	Взима най-често срещаната стойност в колоната и се използва като заместваща стойност.
Премахване на ред (Remove Entire Row)	Премахва напълно всеки ред в набора от данни, който има една или повече липсващи стойности. Това е полезно, ако липсващата стойност може да се счита за случайно липсваща.
Премахване на колона (Remove Entire Column)	Премахва напълно всяка колона в набора от данни, в която има една или повече липсващи стойности.
Замяна с помощта на вероятности (Replace using Probabilistic PCA (Principal Component Analysis))	Заменя липсващите стойности с помощта на линеен модел, който анализира корелациите между колоните. Допълнителни предимства на PCA са подобрената визуализация на данните и оптимизирането на използваните ресурси от алгоритъма за обучение.

Избран подход:

За поставените в дисертацията задачи е важна последователността на данните и не се извършва сортиране на масива. Счита се, че липсващите данни са в резултат от

неизправност в работата на IoT устройствата, които са основен източник на данни, затова е избран метод за многовариантно дописване с помощта на верижно уравнение.

Обработка на екстремни стойности

Големите различия в стойностите ***(****outliers****)*** се наричат „екстремни стойности“ (Yang J., 2019), които са далеч от останалите наблюдения. Те оказват въздействие на прогнозите или оценките.

Причини за образуване на големите различия в стойностите са:

- човешка грешка;
- неизправност на измерителното оборудване;
- грешка при предаване на данни или транскрипция;
- специфично поведение на системата;
- естествени отклонения (outliers);
- грешка при извадката;

За определяне на екстремните стойности, получени при събирането на данни, се използва RANSAC алгоритъм (Kaspi O., 2017). Този алгоритъм предоставя статистическа оценка за вероятността за получаване на надеждни прогнози, т.е. прогнози в рамките на предварително определен брой стандартни отклонения от истинските стойности. Също така алгоритъмът може да се тълкува като метод за откриване на извънредни ситуации. Той е недетерминистичен алгоритъм – произвежда резултат само с определена вероятност, като тази вероятност се увеличава, защото се разрешават повторения. Алгоритъмът произвежда и валидира линеен QSAR (Количествено съотношение структура-дейност) модел (Consonni V., 2019), базиран на метода MLS (Minimum Least Square - метод на най-малките квадрати) чрез:

- отстраняване на зашумени проби (т.е. отклонения);
- избор на най-добрите характеристики (т.е. дескриптори);
- извличане на QSAR модел от набор от обучаващите примери;
- прогнозиране на активността на извадките от тестови набори.

RANSAC взима произволен брой от всички записи в базата данни и прилага линейна регресия. Входните данни могат да включват зашумени проби. Този алгоритъм осигурява статистическа оценка на вероятността за получаване на надеждни прогнози, т.е. вероятност в рамките на предварително определен брой стандартни отклонения от действителните стойности.

$$Y_{measured}(\bar{x}) = Y_{noice-free}(\bar{x}) + N,$$

където:

$Y_{noice-free}(\bar{x})$ очакваната активност в среда без шум и N е случайният вътрешен шум, който се подчинява на предположение, че има постоянно разпределение във всички стойности на активност.

2.1.3 Нормализиране на данни

Поради различният вид и диапазон на данните, получени от IoT системите, е необходимо те да бъдат нормализирани. Най-популярните методи (таблица 2.2) за нормализация са: Zscore, MinMax, LogNormal и TanH (Saif S., 2017) (Samariya D., 2020).

Таблица 2.2: Анализ на методи за нормализация

Метод	Описание
Zscore	Броят на стандартните отклонения от средната точка на данните. Zscore варира от (-3) стандартни отклонения (които биха паднали в най-лявата страна на нормалната крива на разпределение) до (+3) стандартни отклонения (които биха паднали в най-дясната страна на нормалната крива на разпределение).
MinMax	Линейно преоразмерява всяка функция в [0, 1] интервал. Пренасочването към интервала [0, 1] се извършва чрез преместване на стойностите на всяка от характеристиките така, че минималната стойност да бъде 0 и след това се раздели на новата максимална стойност, която е разликата между първоначалните максимални и минимални стойности.
LogNormal	Преобразува всички стойности в логаритмична скала.
TanH	Всички стойности се преобразуват в хиперболична тангента.

Избран подход:

Избраният в този дисертационен труд тип за нормализация на данните е MinMax. Той извършва нормализация чрез линейна трансформация към първоначалните данни, където min_a и max_a са минималните и максималните стойности

за атрибута *a*. MinMax нормализацията изобразява стойностите (map) *v* във *v'* в диапазон [$new\text{-}min_a$, $new\text{-}max_a$] със следната формула:

$$v' = [(v - min_a)/(max_a - min_a)] * (new\ max_a - new\ min_a) + new\ min_a$$

Чрез този тип нормализация се намалява корелацията между колоните, което подобрява работата на алгоритмите.

2.1.4 Проучване и анализ

Проучване, откриване, структуриране и анализ са операции, които са изключително полезни за опознаването на събраните данни. Изследването на набор от сурови данни подпомага избора на най-добър подход за извършване на аналитични проучвания (Waltenburg E., 2012). Това позволява откриването и осмислянето на отклонения (outliers) от данните. Този подход се използва, за да се генерализират данните и да се обобщят техните основни характеристики. Главните цели за прилагане на този подход върху извлечените данните от сензорите са:

- Да се създадат хипотези за причините за наблюдаваните явления.
- Да се подпомогне избора на подходящи статистически инструменти и техники.
- Да се изгради база за бъдещо събиране на данни чрез проучвания и експерименти.
- Да се определят връзките между променливите.

Перспективата на проучвателния анализ на данните е описана във формулата:

$$\boldsymbol{Data = Smooth + Rough}$$

Това се превръща в основно правило, че данните трябва да се разглеждат в две части. Първата част се нарича "гладка" и се отнася до закономерностите, които могат да бъдат извлечени от суровите данни при използване на различни техники. Техниките на анализа на изследваните данни (EDA) се фокусират върху извличане на „гладкото“ от всеки набор от данни. Това идва от данните и не произтича от нашите очаквания или предположения (хипотези) за данните. Една променлива може да има повече от единна структура (pattern) или „гладкост“. Извличането на гладкостта от необработените данни може да изисква повече от едно преминаване през набора данни и може да има повече от една закономерност, която данните съдържат.

Втората част на формулата се нарича „груба“. Тази част представлява остатъци, които не съдържат никаква схема или шаблон. Те са останали, след като всички шаблони, закономерности са извлечени от набора от данни. Много е важно обаче да се разгледа внимателно „грубата“ част, тъй като този набор от стойности може да

съдържа допълнителни шаблони и закономерности, на които трябва да бъде обърнато внимание.

Регулиране на дисбаланса в класовете

Небалансираните данни са често срещан проблем при класификация. Данните са небалансирани, ако класификационните категории не са представени в приблизително еднакво количествено. Често наборите от данни в реалния свят са съставени предимно от „нормални“ примери с малък процент от „необичайни“ или „интересни“ примери.

Най-честите практики за справяне с проблемите на дисбаланса в класовете са:

1. Синтез на нов екземпляр от малцинствения клас;
2. Вземане на проби от малцинствения клас;
3. По-малка извадка от мнозинството;
4. Създаване на функция за оценка с която да се направи погрешната класификация на малцинствените случаи по-важна от грешната класификация на мнозинството.

Препоръчаните в този дисертационен труд техники за регулиране на разпределението на класовете в набора от данни на IoT системи са:

- Синтез на нова инстанция от малцинствения клас (Synthetic Minority Oversampling Technique - SMOTE);
- Редуциране на извадката от мнозинството (Undersampaling of Majority Class).

Тези методи се използват както в статистическата извадка и методологията за проектиране на проучвания, така и в процеса на машинно обучение.

Двата метода са противоположни и приблизително еквивалентни техники, които се прилагат при наличието на определени условия, наблюдавани в данните.

- ***Синтез на нова инстанция от малцинствения клас***

Моделът „Синтез на нова инстанция от малцинствения клас“ е техника, която генерира синтетични проби от малцинствения клас. Той се използва за получаване на синтетичен клас, балансиран или почти балансиран тренировъчен набор, който след това се използва за обучение на класификатора. Пробите SMOTE са линейни комбинации от две подобни проби от малцинствения клас ($\mathbf{x}$ и $\mathbf{x}^R$) и са дефинирани като:

$$\mathbf{s} = \mathbf{x} + u \cdot (\mathbf{x}^R - \mathbf{x}) ,$$

където $0 \leq u \leq 1$; $\mathbf{x}^R$ е случайно избран сред 10 малцинствени класа на най-близките съседи на **x**.

Методът SMOTE е подход при изграждане на класификатори от небалансирани масиви от данни. Недостатъчната извадка от клас мнозинство е предложена, като добро средство за увеличаване на чувствителността на класификатора към малцинствения клас. Подходът синтезира нови случаи на малцинство между съществуващи случаи. Създават се вектори между съществуващите малцинства и се представят нови. По този начин дисбалансът се свива. Това е статистическа техника за увеличаване на броя на случаите в набора от данни по балансиран начин. Алгоритъмът взима проби от функционалното пространство за всеки целеви клас и неговите най-близки съседи и генерира нови примери, които съчетават характеристиките на целевия случай с характеристиките на неговите съседи.

- ***Редуциране на извадката от мнозинството***

Методът „Редуциране на извадката от мнозинството“ (Undersampaling of Majority Class) се характеризира с това, че класът от мнозинството бива редуциран на случаен (random) принцип. Чрез намаляване на записите в класа на мнозинството се постига баланс с класа на малцинството. Този метод подпомага балансирането на набора от данни (Chawla V., 2002), но съществуват редица рискове за нарушения в данните, като някои от тях са изтриване на важни данни, които показват аномалии в процеса или изтриване на група наблюдения, които водят до цялостно отклонение в резултатите.

Балансирането на данните е етап от процеса по обработка на данните, който се прилага в зависимост от дефинирания проблем. Изборът на метод за редуциране на дисбаланса в класовете се определя индивидуално в зависимост резултатите от направения анализ за състоянието на наличните данни.

2.2 Моделиране

В парадигмата за машинно обучение, моделът се отнася до математически израз на параметрите на модела заедно с входните характеристики за всяка прогноза, клас и действие за регресионни, класификационни и подсилващи (reinforcement) категории (Blagus R., 2013).

Моделирането се използва за прогнозиране – за същността му се изискват добри теоретични математически познания. Като се започне от класическия линеен модел на

регресия (ако трябва да се прогнозира или установят тенденции) и се стигне до категоризиране на класове за решаване на класификационни задачи.

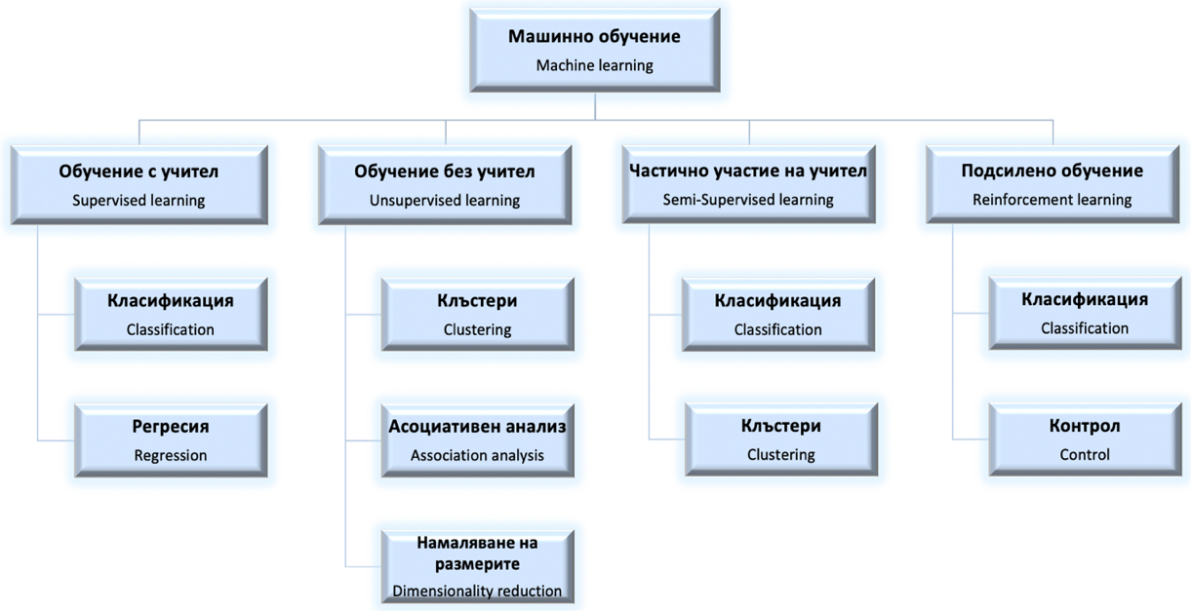
Генерираният модел приема входящи данни с предварително определена структура (подготвени, изчистени, нормализирани и т.н.).

2.2.1 Анализ на алгоритми за машинно обучение

Начинът за решаване на определени типове задачи често пъти се свежда до формулиране на последователност от указания, с които се описват определени действия. Такъв подход се нарича алгоритмичен, а системата от указания, с изпълнението на които се цели получаване на определен резултат – алгоритъм.

В машинното обучение съществуват различни групи алгоритми (таблица 2.3). Избирането на подходящ алгоритъм за решаване на даден проблем зависи основно както от вида и размера на данните, така и от тяхното качество и количество (Kolchakov K., 2018). Познаването на данните и ясно зададените цели са в основата за определяне на правилната за употреба на група от алгоритми (Tashev T., 2011), (Zhou Z., 2012), (Georgieva P., 2013), (Tashev T., 2014) и (Bonaccorso G., 2017).

Съществува разделение на алгоритмите за машинно обучение в различни типове категории според предназначението им (фиг. 2.4).



Фигура 2.4. Типове категории на алгоритми за машинно обучение

- Обучение с учител (Supervised Learning) – се използва за направа на прогнози на базата на примери. Алгоритъм за обучение с учител търси закономерности в етикетите на стойността.
- Обучение без учител (Unsupervised Learning) - данните нямат свързани с тях етикети. Целта на алгоритъма за обучение без учител е да се организират данните или да се опише тяхната структура.
- Обучение с частично участие на учител (Semi-Supervised Learning) - входните данни са смес от етикетирани и неетикетирани данни. Налице е проблем за прогнозиране, но моделът трябва да научи структурите за организиране на данните, както и да направи прогнози.
- Подсилено обучение (Reinforcement Learning) - дава възможност да се избере действие в отговор на всяка точка от данните. Алгоритъмът за обучение получава сигнал за поощрение, показвайки колко е добро решението. Въз основа на това алгоритъмът променя своята стратегия, за да постигне най-високото поощрение.

Разглеждаме няколко специфични вида (Dharmendra S., 2016), (Tomov P., 2019):

***Обучение с учител*:**

- Класификация (Classification) - данните се използват за прогнозиране на дадена категория;
- Регресия (Regression) – данните се използват за предвиждане на стойности.

***Обучение без учител*:**

- Клъстеризация (Clustering) – групиране на набор от обекти.
- Асоциативен анализ (Association Analysis) – метод базиран на правила за откриване на връзки между променливи и данни.
- Намаляване на размерността (Dimensionality Reduction).

***Обучение с частично участие на учител*:**

- Класификация (Classification) - данните се използват за оценка на риска.
- Клъстеризация (Clustering) – данните се използват за идентифициране на групи с висока плътност в данните.

Подсилено обучение с утвърждение:

- Класификация (Classification) - данните се използват за вземане на решения чрез дълбочинно обучение.
- Управление – използва се в системи за управление, като данните се преглеждат и подсилват чрез парадигми за решения.

Таблица 2.3 Сравнителен анализ на алгоритми за машинно обучение

Характеристика	Ползи	Недостатъци	Употреба
Обучение с учител (Supervised Learning)			
Linear Regression			
Принципът на този алгоритъм е да открие линейна зависимост в данните. След като тя е установена, се прави предсказване на нова стойност по отношение на тази връзка.	• лесен за прилагане • лесно се тренира голям набор от данни • не изисква много памет • доста бърз • лесен за интерпретиране • рядко се наблюдава прекомерно нагаждане (overfit)	• изисква линейно разпределение на данните • не може да улови нелинейни връзки без първо да трансформира входовете • нестабилен	• Прогнозни продажби на продукти в бъдеще въз основа на покупателното поведение в миналото. • Предсказване на икономическия растеж на държава. • Спортен анализатор – предвижда се броя на головете или целите, които даден играч ще отбележи в следващите мачове въз основа на предишни изпълнения. • Изчисляване на работна заплата въз основа на годините опит. • Анализи в строителството – предвиждане, колко къщи ще бъдат продадени през следващите месеци и на каква цена.
Naive Bayes			
Колекция от класификационни алгоритми, базирани на теоремата на Bayes. Класификаторите на Bayes приемат, че стойността на определена функция е независима от стойността на всяка друга характеристика. (Пример: счита се за ябълка, ако е червена, кръгла и с диаметър около 10 см. При наличието на тези характеристики се приема за ябълка, независимо от възможностите за корелация между цветове, обла форма и диаметър.)	• Лесен за разбиране и изграждане • Лесен за обучаване, дори и с малък набор от данни • Бърз • Заема малко памет.	• Неуспешно предсказване на редки събития. • Нереалистични предсказания.	• Прогнози в реално време. • Предсказване на много класове. • Спам филтриране. • Анализ на настроенията • Класификация на текста. • Препоръчителна система – предвижда се дали даден потребител би искал даден ресурс. • Разпознаване на лица.
Logistic Regression			

Алгоритъм за анализ на набор от данни, в които има една или повече независими променливи, които определят резултата. Има само два възможни резултата (дихотомна променлива). Целта на логистичната регресия е да бъде намерен най-подходящият модел, за да се опише връзката между променливите.	• по-малко параметри за учене. • ефективни реализации.	• нуждае се от голям брой примери за обучение. • понякога е трудна за интерпретация. • не се справя добре с голям брой променливи.	• Спам писма – предсказва дали имейла е спам. • Измама с кредитни карти – предсказва дали една транзакция с кредитна карта е измама или не. • Медицина - предсказва дали дадена маса от тъкани е доброкачествена или не. • Маркетинг • Банкиране – предсказва дали клиентът ще получи кредит.
Support Vector Machine (SVM)			
Алгоритъм за класификация и регресия. В този алгоритъм задаваме всяка точка от данните като точка в *n*-мерното пространство (където *n* е броят на признаци, характеристики), стойността на всяка характеристика е стойността на дадена координатна система. След това се извършва класификация чрез намиране на оптимална хипер равнина, която отличава двата класа много добре.	• Работи много добре с ясен марж на отделяне • Ефективен в пространства с големи размери • Автоматично се създава нелинейна функция • Ефективен, когато размерността *n* (dimensions) е по-голяма от броя на пробите. • Използва подгрупа от точки за обучение във функцията за вземане на решение. • Заема малко памет.	• Не работи добре, когато разполагаме с голям набор от данни, защото времето за обучение е по-високо. • Не работи добре при данни с повече шум. • Не се предоставя директно оценка на вероятностите.	• Популярен при обработката на изображения. • Често се използва за класифициране на текст, като например: определяне на категории, откриване на спам и анализ на настроенията. • Използва се и за разпознаване на ръкописи.
Decision Tree			
Алгоритъм за класификация и регресия. Разделя данните за трениране в райони със сходни характеристики. От голямо значение е задаването на правилен брой листа на дърво. Ако листата са малко, има риск от недостатъчно нагаждане, а ако листата са повече от необходимото – има риск от прекомерното нагаждане.	• Лесен за прилагане; • Лесен за поддръжка; • Добри резултати дори и с малко данни; • Необходими са няколко параметъра и са доста интуитивни; • Бърз;	• Заема много памет. • Склонен към прекомерно нагаждане (overfitting). • Генерира модели с висока вариация.	• Пускане на нов продукт – кога е най-подходящият период. • Реклама на продукт – кой тип знаменитост е най-подходящ за рекламата. • Предсказване търсенето на определени услуги. • Предсказване дали даден клиент ще желае да се включи към определена услуга.

Random Forest			
Алгоритъм за класификация и регресия. Тренират се няколко дървета за вземане на решения. Наборът от данни се разделя по случаен принцип на няколко подмножества с еднакви размери.	• Устойчив на свръхнатоварване. • Осигурява висока точност (рядко overfits). • Параметрите остават интуитивни. • Работи добре при голям брой променливи и голямо количество данни за учене. • Автоматично обработва липсващите стойности. • Няма нужда от трансформация на променливите.	• Заема много памет. • Обучението може да бъде бавно. • Не е възможно да се подобряват интерактивно генерираните модели. • Трудна интерпретация.	• В биоинформатиката. • В банковия сектор за определяне дали един клиент е лоялен или е измамник, като се идентифицират транзакциите. • Фармацевтичен сектор – определя се приемлива комбинация от лекарства. Също така може да се извърши идентифициране на болест на пациента чрез анализиране на медицинския му картон. • Финансов пазар – може да се анализира пазара на акции. Също така може да покаже очакваната загуба или печалба, която може да бъде произведена при закупуване на конкретен курс. • Електронна търговия – за препоръка на продукти при определен тип клиенти според техните интереси.
AdaBoost			
Алгоритъм за класификация. Може да се използва за увеличаване производителността на всеки слаб алгоритъм за машинно обучение.	• Подобрява точността при класификация. • Може да бъде използван в много области. • Лесен за имплементиране. • Сравнително добра генерализация. • Не се наблюдава прекомерно нагаждане на модела към данните.	• Необходима е голяма сигурност в качеството/ достоверността на данните за обучение. • Чувствителен към шумни данни и екстремни стойности (outliers).	• Може да бъде използван в широк кръг от области, като допълнение на друг алгоритъм.
Обучение без учител (Unsupervised Learning)			
K-Means Clustering			

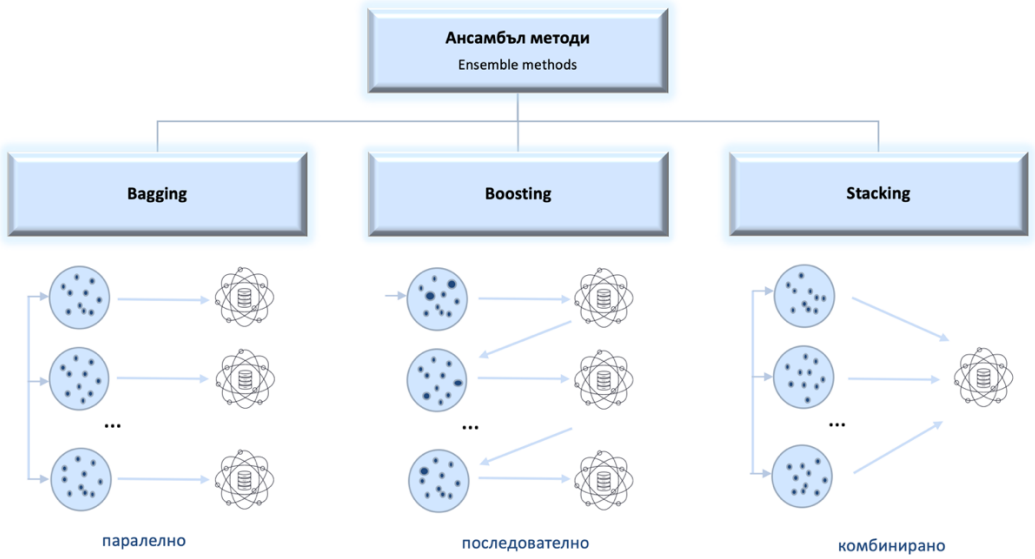
Има за цел да разпредели n наблюдения в k клъстери, при които всяко наблюдение принадлежи към най-близкия клъстер с най-близкия съсед (mean), служещ като прототип на клъстера. Това води до разделяне на пространството.	• Прост, гъвкав и ефикасен. • Лесна регулация при проблеми. • Добър при сегментирането на данни. • Лесна интерпретация. • Линейна сложност.	• Не позволява да се развие най-оптималният набор от клъстери и броят на клъстерите трябва да бъде определен преди анализите. • Избират се на случаен принцип няколко места, от които да се развиват клъстерите. Оттам центърът на клъстерите се преизчислява, докато се намери подходящ "център" за броя на заявените клъстери. • Редът на данните оказва влияние върху крайните резултати. • Не позволява работа с шумни данни.	• Класификация на документи • Оптимизация на доставки • Идентифициране на местата/районите за престъпления • Сегментиране на клиенти въз основа на историята на покупките • Анализ на статистиката в спорта • Разкриване на застрахователни измами • Анализ на трафика • Профилиране на киберпрестъпници • Анализ на детайлите на съобщение/разговор. • Категоризиране на ИТ сигнали/аларми и прогнозиране на средно време за ремонт.
Hierarchical Clustering			
Алгоритъм, който изгражда йерархия на клъстерите. Два съседни клъстера се свързват в единен клъстер. Този алгоритъм завършва, когато има само един клъстер.	• Приема шумни данни. • Не е необходимо предварително определяне на броя клъстери. • Алгоритъмът не е чувствителен към избор на показател за разстояние.	• Не може да обработва големи набори от данни. • Сложността на алгоритъма е $O(n^3)$.	• Икономически анализи на разходите за различни групи услуги.
DBSCAN Clustering			
Алгоритъм за клъстеризиране, използван като заместител на K-Means Clustering за прогнозни анализи. Това е алгоритъм за клъстеризация на базата на плътността: от точки в пространството се обединяват точки, които са тясно свързани помежду си (точки с много близки съседи), маркират се излишните точки, които са разположени сами в региони с ниска плътност. Изискват се два параметъра	• Не изисква предварително определяне на броя клъстери • Притежава ефект "single-link" – различни клъстери, свързани с тънка линия от точки. • Устойчив на свръхнатоварване. • Не е чувствителен към подредбата да данните. • Предназначен за използване с бази	• Не е напълно детерминиращ – граничните точки, които могат да бъдат достигнати от повече от един клъстер, могат да бъдат част от всеки клъстер, в зависимост от реда, по който се обработват данните. • Качеството на този алгоритъм зависи от "измерването на разстоянието", използвано във	Използва се за откриване на асоциации и структури данни, които са трудни за ръчно намиране, но могат да бъдат полезни за намиране на модели и предвиждане на тенденции. • В биологията • Медицината • Социални науки • Археология • Разпознаване на знаци

(ε (eps)-минимална дистанция между две точки и минимален брой точки) за образуване на плътна област. DBSCAN е един от най-често използваните алгоритми за клъстеризация.	данни, които могат да ускорят търсенето в региона.	функцията regionQuery. При Big data, намирането на подходяща стойност е затруднено. • Не може да групира голям набор от данни с големи разлики в плътността. • Ако данните и мащабът не са добре описани, изборът на смислен праг на разстояние може да бъде труден.	
Обучение с частично участие на учител (Semi-Supervised Learning (SSL))			
Self-Training Algorithm			
Първоначално се обучава с малък брой примери с етикети, след това класификаторът се използва за предсказване на етикетите на неетикетирани примери (стъпка на предсказване). След това подмножеството S (състои се от няколко неетикетирани примера с прогнози с висока точност), заедно с предсказаните етикети тренира нов класификатор (стъпка на селекция).	• Лесен за прилагане. • Интуитивен. • Произвежда много добри резултати.	• Добавените неетикетирани данни замърсяват оригиналните етикетирани данни.	• Категоризиране на снимки. • Обработка на естествен език (NLP).
Graph-Based Algorithms (graph traversal, min-cut и node ranking)			
Управление на връзките между субектите. Намират се обекти, които отговарят на определени структурни свойства, определени по отношение на други субекти. Намира се глобално оптимално решение, което има отношение между субектите.	• Ясна математическа рамка. • Висока ефективност. • Може да се разшири до насочени графи.	• Ако графът е лош, то е лошо и изпълнението. • Чувствителна структура.	• Намиране на сходства чрез дублиране на думи в съдържание. Например: Класифициране на места спрямо статии за пътуване. • Откриване на цитирания • NLP (Natural Language Processing)
Co-training Algorithm			
Алгоритъм, който се използва, когато има малки количества етикетирани данни и големи количества неетикетирани данни.	• Малко чувствителен. • Може да се прилага за почти всички	• Необходимо е повече време за трениране.	• SEO – текстообразуване за търсещи алгоритми като Google, Yandex, Yahoo • Класифициране на уеб страници(текстът в

	съществуващи класификатори.		хипервръзката на една страница може да даде информация за страницата, към която се свързва.)
Обучение с утвърждение (Reinforcement Learning (RL))			
Q-learning (Deep Q-learning and Double Q-learning)			
Целта на този алгоритъм е да се научи какво действие **a** е най-добре да бъде предприето в състояние **s.**	• Не изисква адаптация • Бърз • Може да учи от непълни данни. • Изчаква се само една стъпка напред, за да се актуализират прогнозните стойности.	• Ограничен брой стъпки.	• Изграждане на AI за игри • Виртуална реалност • Роботика • Индустриална автоматизация • Обобщаване на текстове • Системи за препоръки при онлайн пазаруване (обратна връзка)
Monte Carlo			
Методите на Монте Карло са подгрупа от изчислителни алгоритми, които използват процеса на повторно вземане на примери/проби, за да направят цифрови оценки на неизвестни параметри. Те позволяват моделиране на сложни ситуации, при които се включват много случайни променливи и оценка на риска.	• Работи отлично със сложни системи • Позволява анализ на чувствителността и оптимизация на реалната система • Поддържа добър контрол	• Работи само за епизодични задачи • Може да се учи само от пълни последователности • Необходимо е да се изчака до края на епизода • Аналитичните модели често са невъзможни	• Физика • Теория на игрите • Финанси – прогнозиране на фондови пазари

Методи за подсилване на алгоритми

Машинното обучение и науката за данни изискват нещо повече от просто използване на съществуващи алгоритми. Един от важните моменти е да се знае кои алгоритми могат да прилагат методи за подсилване (Mease D., 2007). Тези методи се използват за създаване на така наречените ансамблови модели, които могат да подпомогнат подобряването на точността на алгоритъма и да направят модела по-стабилен. Това се постига чрез комбиниране на слаби алгоритми с ансамбъл метод (Agarwal K., 2020), (Tewari S., 2020) и (Balabanov T., 2020).



Фигура 2.5. Типове методи за подсилване на алгоритми за машинно обучение

Съществуват 3 метода за подсилване на алгоритми (фиг. 2.5):

Bagging – много лесен и много мощен паралелен ансамблов метод. Използва се за алгоритми с висока вариация (high-variance). Предназначен да подобри стабилността и точността на алгоритмите за машинно обучение, използвани за класификация и регресия. Различните модели се изграждат паралелно. Всяко дърво се обучава върху нова проба, създадена чрез произволно изваждане на данни от оригиналния набор от данни с подмяна, докато се достигне до набор от данни с размера на оригинала. Всички модели правят предположения за финалната прогноза. Взима се най-често получения изход на изградените дървета. Всички модели получават равни тегла. Чрез този метод се намалява вариацията.

Bagging алгоритми: Random Forest, Bagged Decision Trees и Extra Trees.

Boosting – Последователен ансамблов метод. Може да се използва за увеличаване производителността на всеки слаб алгоритъм за машинно обучение. Обучава се последователно. Намира често приложение при Decision Trees, като едно дърво се научава се от предишното дърво, като се фокусира върху неговите неправилни наблюдения. Създава се нов модел с по-голямо тегло за правилните наблюдения от предишната последователност. По-успешните модели получават по-големи тегла. Чрез този метод се редуцират пристрастията (фалшиви положителни резултати).

Boosting алгоритми: AdaBoost, Gradient Boosting, XGBoost и Boosted Decision Tree.

Stacking – ансамблов метод, при който новия модел се обучава от комбинираните прогнози на два или повече предходни модели. Прогнозните модели се използват като входни данни за всеки последователен слой и се комбинират, за да образува нов набор от прогнози. Те могат да използват допълнителни слоеве или процесът да спре с краен резултат. Чрез този метод се подобряват предвижданията при прогнозирането.

Stacking алгоритми: Votting Ensemble.

Избрани ML алгоритми за класификационен анализ

След извършен сравнителен анализ в този дисертационен труд (таблица 2.3) на типовете алгоритми и насоките за тяхното правилно прилагане са избрани 4 класификационни алгоритъма за работа - Decision Tree, Bayes Point Machine, Logistic Regression и Support Vector Machines (SVM).

Тези модели са избрани спрямо целите на проучването и наличните данни в базата - данните се използват за прогнозиране на категория.

1. ***Two-Class Boosted Decision Tree*** – Алгоритъм способен да обработва както цифрови, така и категорийни данни. Целта му е да създаде модел, който да прогнозира стойността или типа на целевата променлива чрез изучаване на лесни правила за вземане на решения, изведени от характеристиките на данните. Дървото на решенията е структура, подобна на блок-схема, в която всеки вътрешен възел представлява "тест" на атрибут, всеки клон представлява резултат от тест и всеки възел на листата представлява клас (решение, взето след изчисляване на всички атрибути). Пътеките от корен до листо представляват правила за класификация (Rogozhnikov A., 2016).

Дърветата на решенията обикновено се използват в операционни изчисления, по-специално при анализа на решения, за да подпомогнат идентифицирането на стратегия за постигне на дадена цел.

За да се изгради дърво на решенията е необходимо да се изчислят два вида ентропия, като се използват таблици с честоти, както следва:

- Ентропия чрез използване на честотната таблица на един атрибут:

$$E(S) = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$$

където $\boldsymbol{p_i}$ е вероятност за клас i;

S – атрибут.

- Ентропия чрез използване на честотната таблица на два атрибута:

$$E(T, X) = \sum_{c \in X} P(c)E(c),$$

(T, X) – атрибути;

С цел по-добро генерализиране на данните от алгоритъма той се комбинира с ансамблов метод за подсилване – Boosting. Тази съвкупност води до намаляване на опасността от прекомерно нагаждане на модела към данните и получаване на точни резултати.

2. ***Two-Class Bayes Point Machine*** – този алгоритъм ефективно апроксимира теоретично оптималното байесово средноаритметично ниво на линейни класификатори, като избира един „среден" класификатор, точката на Bayes (Jena S., 2017). Bayes Point Machine е байесов класификационен модел, който не е склонен към прекомерно нагаждане на данните от обучението. Bayes Point Machine за трениране на теглата дава нов вход x_{n+1}, като предсказуемото разпределение може да бъде апроксимирано, както следва:

$$p(x_{n+1}|x_{n+1}, Y) = \int p(y_{n+1}|x_{n+1}, W)\, p(w|Y)dw \approx P(y_{n+1}|x_{n+1}, \prec w \succ),$$

където $\prec$ w $\succ$ е средна стойност на теглата и се нарича - Bayes Point.

Колкото по-голям е броят на итерации на обучение, толкова са по-точни прогнозите, но обучението става по-бавно.

3. ***Two-Class Logistic Regression*** – Алгоритъм за моделиране на двоична зависима променлива (Kost S., 2020). Входните стойности (x) се комбинират линейно, като се използват тегла или коефициенти стойности, за да се предскаже изходна стойност (y). Ключова разлика на логистичната от линейната регресия е, че изходната стойност, която се моделира, е двоична стойност (0 или 1), а не числова стойност.

$$\text{y} = \frac{\text{e}\hat{}(b0 + b1 * x)}{\left(1 + e\hat{}(b0 + b1 * x)\right)}$$

където ***y*** е прогнозираният изход, ***b0*** е отклонение или прихващане и ***b1*** е коефициент за единична входна стойност (x).

4. ***Two-Class Support Vector Machines (SVM)*** – Модел на обучение с учител, който изисква етикетирани данни. По време на тренировъчния процес, алгоритъмът (Rustam Z., 2018) анализира входящите данни и разпознава шаблоните в пространството с многомерни характеристики, наречено „хиперплан". Всички примери

са представени като точки в това пространство и са съпоставени към категориите на изхода по такъв начин, че те се разделят с възможно най-широка и ясна граница.

За този тип SVM обучението включва минимизиране на функцията за грешки:

$$\frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$$

при спазване на ограниченията:

$$y_i(w^T \phi(x_i) + b) \geq 0, i = 1, \ldots, N,$$

където ***C*** е константата на капацитета, ***w*** е векторът на коефициентите, ***b*** е константа, а ***ξ*** представлява параметри за обработка на неотделими данни (входове).

Избрани ML алгоритми за регресионен анализ

След извършен сравнителен анализ в този дисертационен труд за типовете алгоритми и случаите за тяхното правилно прилагане (таблица 2.3) са избрани 4 регресионни алгоритъма за прогнозиране на стойности - Linear Regression, Boosted Decision Tree Regression, Bayesian Linear Regression и Decision Forest Regression.

1. ***Linear Regression*** - Линейна регресия е алгоритъм за машинно обучение, основан на контролирано обучение. Решава регресионна задача. Регресията моделира целева стойност за прогнозиране, базирана на независими променливи (Naseem I., 2010). Използва се най-вече за установяване на връзката между променливите и прогнозирането. Различните модели на регресия се различават в зависимост от вида на връзката между зависимите и независимите променливи, които те обработват и броя на независимите променливи, които се използват.

Линейната регресия изпълнява задачата да предскаже зависима стойност на променлива (y) въз основа на дадена независима променлива (x). Тази техника на регресия открива линейна връзка между x (вход) и y (изход). Регресионната линия е най-подходящата линия за този модел.

$$y = \theta_1 + \theta_2 . x,$$

където ***x*** входни данни за обучение (унивариантна - една входна променлива - параметър),

y - етикети за данни (контролирано обучение),

θ_1 и θ_2, - коефициенти.

2. ***Boosted Decision Tree Regression*** - Алгоритъм за машинното обучение за разрешаване на регресионни задачи, която създава модел на прогнозиране под формата на ансамбъл от слаби прогнозни модели - дървета за решения (Poyarkov A., 2016). Той

изгражда модела поетапно, както правят другите стимулиращи методи, и ги обобщава, като позволява оптимизиране на произволна функция на различна загуба.

Всяко дърво е създадено итеративно:

- На изхода на дървото (h_t (x)) се дава тегло (w) спрямо неговата точност,
- Изходът на ансамбъла е претеглената сума:

$$\hat{y}(x) = \sum_t w_t \, h_t(x)$$

След всяка итерация на всяка проба от данни се дава тежест въз основа на нейната правилна класификация. Колкото по-често дадена извадка от данни е класифицирана, толкова по-важна става.

Целта е да се сведе до минимум функцията:

$$O(x) = \sum_i l(\hat{y}_i, y_i) + \sum_t \Omega\,(f_t),$$

където:

$\boldsymbol{l(\hat{y}_i, y_i)}$ е функция на загубата – дистанцията между истината и прогнозата на i-тата проба,

$\boldsymbol{\Omega(f_t)}$ е функция на регуларизация – отговаря за сложността на t-то дърво.

3. ***Bayesian Linear Regression*** - алгоритъм за линейна регресия, в който статистическият анализ се извършва в контекста на байесов извод - използвайки разпределения на вероятностите, а не оценка на точките (Ali A., 2019). Моделът, изведен от нормалното разпределение, е:

$$y \sim N(\beta^T X, \Omega^2 I),$$

където изходът **y** се генерира от нормалното разпределение, характеризиращо се със средна стойност и дисперсия. Средната стойност за линейна регресия е транспонирането на матрицата на теглото, умножена по матрицата на предиктора. Вариацията е квадратът на стандартното отклонение σ.

Целта на Bayesian Linear Regression е да определи следващото разпределение на параметрите на модела (Shekaramiz M., 2018). Това разпределение на параметрите се базира на входовете и изходите:

$$P(\beta|y, X) = \frac{P(y|\beta, X) * P(\beta|X)}{P(y|X)}$$

P (β|y,X) представлява следващото разпределение на вероятността на параметрите на модела, като се имат предвид входовете и изходите. Това е равно на вероятността на

данните P (y|β,X), умножена по предварителната вероятност на параметрите и разделена на константа за нормализиране.

4. ***Decision Forest Regression*** – алгоритъм за надграждащо обучение за класификация и регресия, който комбинира резултати от множество прогнози. Всяко дърво взима произволна извадка от оригиналния набор от данни и генерира разделянията си (Criminisi A., 2012), добавяйки допълнителен елемент на случайност, който намалява вероятността за фалшиви прогнози. Броят на променливите, които могат да бъдат разделени на всеки възел, е ограничен до определен процент от общия брой (който е известен като хиперпараметър). Това гарантира, че моделът на ансамбъла не разчита твърде много на всяка отделна характеристика и прави справедливо използване на всички потенциално предсказуеми такива.

След обучението, прогнозните проби x' могат да бъдат направени чрез осредняване на прогнозите от всички отделни регресионни дървета на x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

където **B** е брой дървета, а **b** принадлежи на интервала {1,…,B}.

Това води до по-добра производителност на модела, тъй като намалява дисперсията (отклонението от математическото очакване), без да увеличава пристрастията. Това означава, че докато прогнозите за едно дърво са силно чувствителни към шума в тренировъчния му набор, средната стойност на много дървета не е, стига дърветата да не са свързани. Простото обучение на много дървета в един набор от тренировки би дало силно свързани дървета (или дори едно и също дърво много пъти, ако алгоритъмът за обучение е детерминиран).

2.2.2 Анализ на методи за валидиране

Валидирането измерва точността на модела. Тя се получава чрез разделяне на оригиналната проба в тренировъчен комплект за обучение и тест.

Методите за трениране и тестване (Таблица 2.4) използват различни подходи за разделяне на данните (Vanwinkelen G., 2012), (Balabanov T., 2018).

Таблица 2.4 Анализ на съществуващите методи за валидиране на модел

Метод	Описание	Употреба
Train-Test Split	Разделя данните на две части в процентно съотношение 80:20 или 70:30. По-голямата част се използва за трениране, а по-малката за тестване.	Традиционен метод за трениране и тестване на данни. Ефективен при набори от данни съдържащи малко на брой променливи.
Cross-Validation	Разделя набора от данни на *n* подмножества (сгъвки). Изгражда модел на всяко подмножество и след това връща набор от статистически данни за точността на всяка сгъвка. Използва целия набор от данни за обучение и тестване, а не само част от него.	Когато е необходимо да се оцени както набора от данни, така и модела. Ефективен за преодоляване на прекомерното нагаждане на моделите.
Early Stopping	Метод, позволяващ оказването на голям брой тренировъчни итерации, които могат да бъдат прекъснати, след като производителността на модела спре да подобрява данните за валидиране. Този метод не позволява прекомерно нагаждане на модела към данните.	Използва се за ранно спиране на обучението на невронни мрежи, за да се избегне прекомерно нагаждане на модела към данните и да се достигне до най-висока ефективност.

Избран подход:

Малкият брой колони на данните позволяват използването на традиционният метод - Train-Test Split, който има вградена стратификация на данните, чрез която данните се разбъркват на случаен принцип преди да бъдат разделени на две части – за трениране и тестване в съотношение 70:30.

2.2.3 Валидиране на избрания модел

Чрез валидиране на модела се получава реална оценка за продуктивността и устойчивостта на обучения модел. То се извършва чрез различни метрики, характерни за отделните категории на обучение.

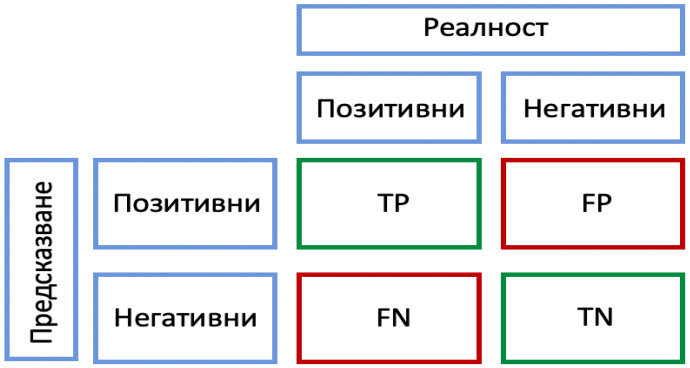
Метрики за оценка на класификационни модели

Изборът на показатели влияе върху начина, по който се измерва и сравнява работата на алгоритмите за машинно обучение.

За описание на работата на класификационния модел върху набор от тестови данни, за които са известни истинските стойности се прилага матрица за оценка на класификацията (фиг. 2.6).

Матрицата за оценка е един от най-интуитивните показатели, използвани за намиране на точността, прецизността, чувствителността и цялостното представяне на модела. Използва се за решаване на класификационни задачи, когато изходът може да бъде от два или повече класа (Singh H., 2019) (Joshi V., 2020). Чрез нея се визуализира производителността на модела.

Матрицата показва начините, по които моделът на класификация се обърква, когато прави прогнози. Тя ни дава представа не само за честотата на грешките, допуснати от класификатора, но и за видовете грешки, които се правят.



Фигура 2.6 Матрица за оценка на класификацията

Матрицата съдържа четири основни квадранта:

Истински положителни резултати (TP): Истински положителни са случаите, когато действителният клас на точката от данни е бил 1 (Истина) и прогнозираният също е 1 (Истина).

Истински отрицателни (TN): Истински негативи са случаите, когато действителният клас на точката от данни е бил 0 (Грешно) и прогнозираният клас също е 0 (Грешно).

Фалшиви положителни резултати (FP): Грешни положителни резултати са случаите, когато действителният клас на точката от данни е бил 0 (Грешно), а прогнозираният клас е 1 (Истина).

Фалшиви негативи резултати (FN): Фалшиви негативи резултати са случаите, когато действителният клас на точката от данни е бил 1 (Истина), а прогнозираният клас е 0 (Грешно).

Матрицата на оценка на класификацията не е мярка за ефективност, но всички показатели за ефективност са базирани на нея и стойностите в нея.

Основните метрики за ефективност базирани на матрицата са: точност, прецизност, чувствителност и средно претеглена стойност.

Точността е главният показател за оценка на моделите за класификация. Тя представя общия процент на правилните прогнози получени от модела. За бинарна класификация, точността може да се изчисли и като положително и отрицателно класифицирани признаци, както следва:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

където TP = True Positives, TN = True Negatives, FP = False Positives and FN = False Negatives.

Точността е най-интуитивната мярка за оценка изградена от съотношение на правилно прогнозираното наблюдение към общите наблюдения. Тя е част от общия набор критерии, които трябва да бъдат следени в процеса на определяне на правилния модел.

Прецизност - съотношението на правилно прогнозираните положителни наблюдения към общите прогнозни положителни наблюдения. Високата прецизност е свързана с ниската степен на фалшиво предсказани положителни резултати.

$$Precision = \frac{TP}{TP + FP}$$

Чувствителност - това е съотношението на правилно прогнозираните положителни наблюдения към всички наблюдения в реалния клас.

$$Recall = \frac{TP}{TP + FN}$$

Средна хармонична стойност (F1 Score) - Това е средно претеглената стойност между точност и чувствителност. Следователно, този резултат взема под внимание както фалшивите положителни резултати, така и фалшивите негативи.

$$F1\ Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

Освен метриките за оценка на ефективността на модела е важно да се обърне внимание и на графиките, които визуализират получените резултати след трениране на модела и ясно онагледяват неговата ефективност.

ROC curve (Receiver Operating Characteristic Curve) – графика, показваща ефективността на модел за класификация (Vasanthi R., 2019). Тази крива очертава два параметъра: Истински положителен размер (True Positive Rate - TPR) и Фалшив положителен размер (False Positive Rate - FPR). В случай на случайни предположения верните положителни предположения ще имат равен брой с фалшивите положителни предположения (TPR = FPR).

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{TN + FP}$$

AUC (accuracy) – площ под кривата ROC. Осигурява обобщена мярка за сигурност във всички възможни прагове за класификация (Ling C., 2005). Варира в стойност от 0 до 1.

Lift curve – мярка за ефективността на прогнозен модел. Изчислена като съотношение между резултатите, получени с и без прогнозния модел. Колкото по-голяма е площта между кривата на повдигане и основната линия, толкова по-добър е моделът.

$$lift = \frac{TP/(TP + FN)}{(TP + FP/(TP + FN + FP + TN))}$$

Добрият анализ на изходните данни след обучение на различни модели за класификация е в основата на правилния избор на модел. Този избор трябва да бъде добре съобразен с целта на поставената задача, за да се получат истински положителни резултати в бъдеще.

Метрики за оценка на регресионни модели

Най-важните показатели за оценка на модела са: средна абсолютна грешка, средна квадратична грешка, относителна абсолютна грешка, относителна квадратична грешка, средна нулева единична грешка и коефициентът на определяне. В зависимост

от получените стойности на тези показатели се намира най-добрият модел, който представя данните. Счита се, че моделът се вписва добре в данните, ако разликата между наблюдаваните и прогнозираните стойности е малка.

• **Средна абсолютна грешка (Mean Absolute Error - MAE)** измерва колко близо са прогнозите до действителните резултати. Прогнозите, клонящи към 0, отразяват по-добрите резултати. Средната абсолютна грешка (Boiroju N., 2011) има същата мерна единица като тази на оригиналните данни и може да сравни само тези модели, чиито грешки се измерват в едни и същи единици.

$$MAE = \frac{\sum_{i=1}^{n}|p_i - a_i|}{n}$$

a_i - реални резултати;

p_i - прогнозни резултати;

n – брой периоди използвани при изчисление.

• **Средна квадратична грешка (Root Mean Square Error - RMSE)** създава единична стойност, която обобщава грешката в модела. Чрез изравняване на разликата, метриката пренебрегва разликата между свръх-предсказване и недостатъчно предсказване. Средна Квадратична Грешка (Ozawa, 2019) е популярен начин за измерване размера на грешката при регресионни модели. Използва се само за модели, чиито грешки се измерват в едни и същи единици.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}}$$

a_i - реални резултати;

p_i - предсказани резултати;

n – брой периоди използвани при изчисление.

• **Относителна абсолютна грешка (Relative Absolute Error - RAE)** е относителната абсолютна разлика между очакваните и действителните стойности. Тази грешка е относителна, защото средната разлика се разделя на средната аритметична стойност.

$$RAE = \frac{\sum_{i=1}^{n}|p_i - a_i|}{\sum_{i=1}^{n}|\bar{a} - a_i|}$$

• **Остатъчна стандартна грешка (Residual Standard Error (RSE)** нормализира общата квадратична грешка на прогнозните стойности чрез разделяне на квадрата на общата грешка на действителните стойности.

$$RSE = \frac{\sum_{i=1}^{n}(p_i - a_i)^2}{\sum_{i=1}^{n}(\bar{a} - a_i)^2}$$

• **Коефициент на определяне**, често наричан R2, представлява предсказващата сила на модела като стойност между 0 и 1. „Нула“ означава, че моделът е случаен (обяснява нищо); „Едно“ означава, че има перфектно пасване. Необходимо е да се внимава при тълкуването на стойностите на R2, тъй като ниските стойности могат да бъдат напълно нормални, а високите стойности могат да бъдат подозрителни.
Коефициентът на определяне (R2) обобщава силата на регресионния модел и се изчислява чрез сумите на квадратите от стойностите на зависимата променлива.

R2 описва пропорцията на дисперсията на зависимата променлива, обяснена от регресионния модел. Ако регресионният модел е "перфектен", SSE е 0, а R2 е 1. Ако регресионният модел е „случаен“, SSE е равен на SST.

$$\mathrm{R}^2 = \frac{\mathrm{SSR}}{\mathrm{SST}} = 1 - \frac{\mathrm{SSE}}{\mathrm{SST}}$$

$$SST = \sum (y - \bar{y})^2$$

$$SSR = \sum (y' - \bar{y}')^2$$

$$SSE = \sum (y - y')^2$$

R^2 (Coefficient of determination) - коефициент на детерминация;
SST (Sum of Squares Total) – обща сума на квадратите;
SSR (Sum of Squares Regression) – сума на квадрати на остатъците от регресия;
SSE (Sum of Squares Error) – сума на квадрати от грешката.

2.2.4 Подобряване на избрания модел

Моделът, който оформя данните за обучение твърде добре, обикновено извършва прекомерно нагаждане към данните (overfitting). Прекомерно нагаждане се случва, когато модел научи детайлите и шума в тренировъчните данни до степен, която влияе негативно върху работата на модела върху нови данни. Това означава, че шумът или случайните колебания в тренировъчните данни се вземат и усвояват като

концепции от модела. Тези концепции не се прилагат за нови данни и влияят негативно върху способността на моделите да се обобщават. В този случай е необходимо обновяване на модела, чрез което да бъде преодоляно прекомерното нагаждане.

Най-често прилаганите подходи за справяне с прекомерното нагаждане на модел към данни са:

1. Кръстосано валидиране (Cross-validation);
2. Обучение с повече данни;
3. Минимизиране на характеристики чрез регуларизация – някои алгоритми имат вграден избор на характеристики. За тези, които не го правят, може ръчно да бъде подобрена тяхната генерализация, като се премахнат неподходящите входни променливи;
4. Избор на променливи.

Спецификите на тези подходи са:

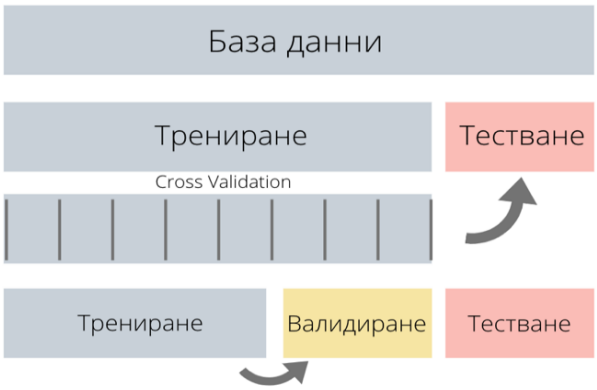
- **Кръстосано валидиране (Cross-validation)**

Кръстосаното валидиране е техника за оценяване на предсказуеми модели чрез разделяне на оригиналната проба в тренировъчен комплект за обучение и тест (фиг. 2.7).

При кръстосано валидиране, оригиналната извадка се разпределя на случаен принцип в подмножества k с еднакъв размер. От пробите k се запазва една подпроба като данни за валидиране и тестване на модела, а останалите подпроби $k-1$ се използват като данни за обучение. Процесът на кръстосано валидиране след това се повтаря k пъти, всяко k се използва само веднъж като данни за валидиране. Резултатът k може да бъде осреднен (или комбиниран по друг начин), за да се получи единична оценка. Предимството на този метод е, че всички наблюдения се използват както за обучение, така и за валидиране и всяко наблюдение се използва за валидиране точно веднъж.

Целта на кръстосаното валидиране (Liu Y., 2020) е да се провери способността на модела да предсказва нови данни, които не са били използвани при оценката му, за да се отбележи проблем като прекомерно нагаждане (overfitting) и да се даде представа за това как моделът ще се обобщи към независим набор от данни.

За задачи за класификация и регресия е най-подходящо използването на Cross-validation testing (Fushiki T., 2011).



Фигура 2.7 Кръстосано валидиране

- **Обучение на модела с повече данни**

Моделите могат да обработват голям обем от данни. При добавяне на нови данни моделът задължително ги генерализира, за да постигне напредък. Добавянето на данни към съществуващото множество води до повишена точност на модела, като същевременно намалява шанса за прекомерно нагаждане (overfitting).

Събирането на данни е скъп процес, затова в практиката се прилага лесна техника за увеличаване на данните (Schnaubelt M., 2020). При тази техника всеки път, когато една проба се обработва от модела, тя трябва да бъде по-различна от предходната. Това затруднява модела да научи параметрите за всяка проба. Друга практика е да се добави шум (Sengupta U., 2020):

- Към входа: това работи за направата на модела по-здрав към естествени смущения, които биха могли да се срещнат в природата.
- Към изхода: това прави обучението по-разнообразно.

И в двата случая за добавяне на шум към данните е необходимо силата на шума да не бъде твърде голяма. В противен случай има опасност данните да бъдат заглушени от изкуствено добавения шум на входа или да се направи изхода неточен. И двете ще попречат на процеса на обучение.

Балансирането на данни (Gulowaty B., 2019) и увеличаването на размера им чрез повишаване на малцинствен клас също е полезна техника за избягване на нежеланото нагаждане на модела към данните. Прилагането на модел над балансирани и небалансирани данни дава отклонение в получените резултати за оценка ефективността на модела.

- **Минимизиране на характеристиките чрез регуларизация**

Регуляризирането в машинното обучение е важна концепция и решава проблема с обучението на добър модел. Това е метод за добавяне на някои допълнителни ограничения към условие, за да се реши некоректно поставен проблем или да се предотврати прекомерното нагаждане.

Регуларизацията е основният начин да се направи баланс между *Bias* (показва колко сме далеч от целта) и *Variance* (показва колко близо сме до целта). Баланс между това да вкараме твърде много предположения в модела и това да го оставим да прави каквото си иска (Minka T., 2001).

Чрез регуляризирането се извършва контрол на сложността на модела.

С цел добро обучение на избраните модели и постигане на максимално точни реални резултати се прилага метода **Elastic Net** (Oleszak M., 2019). Elastic Net е комбинация от два типа регуларизация – L1 и L2, чрез които се определя каква норма (мярка за разстояние) ще се използва.

Elastic Net има за цел да минимизира функциите на загуба чрез:

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i-1}^{n}(y_i - x_i'\hat{\beta})}{2n} + \lambda(\frac{1-a}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + a\sum_{j=1}^{m}|\hat{\beta}_j|),$$

където **a** е параметър на смесване между R1 (a = 0) и L1 (a = 1).

$L_{enet}(\hat{\beta})$ интерполира между L1 норма на β и L2 норма на β.

λ – за регуларизация.

n – брой наблюдения.

y_i - отговор при наблюдение i.

x_i - вектор на p стойности при наблюдение i.

L1 (Least Absolute Shrinkage and Selection Operator **- LASSO**) Оператор с най-малко абсолютно свиване и селекция е оператор (Hans C., 2009), който свива коефициентите и ги свежда до 0, т.е. избира някой от тях по абсолютната им стойност – премахват се най-малките абсолютни стойности. Някои от променливите стават излишни.

L2 (**Ridge Regularization**) – Евклидово разстояние. Стреми се от големи коефициенти да направи малки т.е. стреми се всички коефициенти да клонят към 0.

Отстраняването на частично релевантните променливи значително подобрява точността и бързината за изпълнение на модела (Cortes C., 2012). Изборът на правилна техника за регуларизация е важна част от преодоляването на прекомерното нагаждане.

- **Избор на променливи**

Изборът на променливи е една от основните концепции в машинното обучение,

която оказва съществено влияние върху производителността на модела. Необходимо е да бъдат отстранени незначителните или частично релевантните променливи, защото те могат да окажат отрицателно въздействие върху ефективността на модела. Наличието на неподходящи променливи в данните намалява точността на модела, променя желания изход и той бива обучен въз основа на неподходящи променливи.

Изборът на правилни променливи преди моделиране на данните помага за:

- Намаляване на прекомерното натоварване – по-малко шум;
- Подобряване на точността;
- Намаляване времето за обучение – по-малко точки от данни водят до намаляване на сложността на алгоритмите и по-бързото им обучение;

Методи за избор на променливи

Корелация на Пиърсън (Pearson Correlation)

Коефициентът на корелация на Пиърсън (Samuels P., 2014) представлява статистически модел известен, като **r** стойност. За всеки две променливи се връща стойност, която показва силата на корелация. Коефициентът на корелация на Пиърсън се изчислява, като се вземе ковариацията на две променливи и се раздели на произведението на техните стандартни отклонения. Коефициентът не се влияе от промените в мащаба на двете променливи.

$$r = \frac{n\sum xy - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

n - брой двойки резултати;

$\sum xy$ - сума от произведенията на сдвоени резултати;

$\sum x$ - сума от *x* резултати;

$\sum y$ - сума от *y* резултати;

$\sum x^2$ - сума от квадратите на x;

$\sum y^2$ - сума от квадратите на y;

Корелация на Спирмън (Spearman Correlation)

Коефициентът на Спирман е непараметрична мярка за статистическа зависимост между две променливи. Коефициентът на Спирмън изразява степента (Kotlyar M., 2002), до която две променливи са монотонно свързани. Тя се нарича още Спирмън ранг корелация – r_s. Предназначена е за използване върху непараметрични и небалансирани данни.

$$r_s = 1 - \frac{6\sum d^2}{n(n^2-1)}$$

r_s - коефициент на корелация на Spearman Rank;

$\sum d^2$ - сбор от квадратите на разликите между рангове;

n - брой двойки наблюдения в извадката.

Корелация на Кендъл (Kendall Correlation)

Корелацията на Кендъл измерва връзката между класирането на различни ординални променливи или различни класации на същата променлива. Корелацията на Кендъл (Croux Ch., 2010) е непараметричен тест, който измерва силата на зависимостта между две променливи. Ако разгледаме две проби, *a* и *b*, където всеки размер на извадката е *n*, знаем, че общият брой на двойките с *b* е *n (n-1) / 2*.

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$$

n_c - брой съгласувани;

n_d - брой разминаващи се.

Chi- Squared

Двупосочният chi-squared тест е статистически метод, който измерва колко близки са очакваните стойности към действителните резултати. Методът предполага, че променливите са случайни и са извлечени от адекватна извадка от независими променливи. Получената chi-squared статистика (Mengyu X., 2019) показва колко далеч са резултатите от очаквания (случаен) резултат.

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

O - наблюдаваните честоти;

E - очакваните честоти;

Оценка на Фишер (Fisher Score)

Методът на Фишер е информационна оценка, представляваща количеството информация, която една променлива предоставя за някой неизвестен параметър. Резултатът се изчислява чрез измерване на разликата между очакваната стойност на информацията и наблюдаваната стойност. Когато вариацията е сведена до минимум, информацията е максимална, тъй като очакването на резултата е нула, информацията на Фишер е също вариация на резултата (Luo Q., 2020).

2.3 Интегриране на данни с обучен модел в работна среда

След създаване на успешен модел той бива архивиран и експортиран в структура, която съдържа обект за класификация/регресия и функция за прогнозиране. Този модел се разархивира и имплементира в работна среда, за да прави прогнози, използвайки нови данни.

2.3.1 Нови данни

Добавянето на нови данни към вече обучен модел е необходимо да се извършва при определени условия:

- Новите данни трябва да съдържат същите имена на променливите (колоните), каквито са били използвани при обучението. Функцията за прогнозиране игнорира допълнителни променливи, ако има такива.
- Форматът и видът на данните също трябва да съответстват на оригиналните данни за обучение.

2.3.2 Почистване и нормализиране

Подаването на нови данни към обучения модел трябва да се извърши след почистване на всички невалидни, липсващи и екстремни стойности. В случай на пропускане на този етап от обработка на данните, резултатите след повторното обучение на модела ще дадат отклонения от първоначалните резултати.

Друго важно условие при добавяне на нови данни е те да бъдат нормализирани. Нормализацията трябва да се извърши със същия метод, който е използван при обучението. Причините за това са различните интервали, в които се разпределят данните и различния подход за нормализиране, който е заложен при всеки метод. В случай на промяна на метода за нормализация на новите данни ще се получи съобщение за несъвместимост.

2.3.3 Използване на обучен модел

Едно от ключовите изисквания за един обучен модел е, че той трябва да е преносим, за да бъде интегриран в реална работна среда. Поради тази причина е важен дигиталният формат, в който ще бъде съхранен. Съществуват различни формати представени в (Lunga D., 2020) (Hackeling G., 2014).

Използваните формати в разработваната в този дисертационен труд системата са:

- ***Json* (JavaScript Object Notation) –** стандартен файлов формат за обмен на данни, който използва текст за съхранение и предаване на обекти.

- ***Pickle*** – стандартна библиотека в Python. Използва се за модели изградени от scikt-learn.

2.3.4 Валидиране

След имплементиране на един модел в реална среда е важно да се проследи неговата ефективност.

В жизнения цикъл на моделите е необходимо да има създаден процес, който измерва производителността на модела спрямо новите данни, за да се тества точността му и да се провери неговата актуалност (колко точен е все още). Често използвани метрики за това са Area Under the Receiver Operating Characteristic (AUROC) и Average Precision (AP) (Kantardzic, 2011). В зависимост от получените резултати се определя неговото развитие.

- Ако моделът падне под приемливия праг на производителност се инициира нов процес за преквалификация на модела – връща се към фаза „моделиране“.
- Ако моделът е в приемливия праг на производителност, резултатите от извършеното предсказване или класифициране биват визуализирани и интерпретирани, което представлява финална стъпка за извличане на бизнес стойност от придобитите данни.

2.3.5 Тълкуване и визуализиране

Тълкувателната фаза трябва да отговори напълно или частично на въпросите, които предизвикват моделирането на данните. Това е етапът по време на който се използва всичко научено от събраните и обработени данни. Тази стъпка включва:

- Извличане на заключения от данните;
- Оценка на резултатите;
- Разпространение на резултатите.

Тълкуването на резултатите от изследванията е от първостепенно значение за познаване ефективността на изследването. Необходимо е ясно да се опишат и визуализират резултатите по начин, по който други изследователи могат да ги сравняват със собствените си резултати. За правилното им тълкуване е необходимо те да са ясни, недвусмислени и добре представени графично или таблично. Резултатите се анализират, като се използват подходящи статистически методи, за да се определи вероятността те да са били случайни и да не могат да бъдат възпроизведени в по-големи изследвания.

Резултатите трябва да бъдат интерпретирани по обективен и критичен начин, преди да се преценят последиците от тях и да се направят изводи.

2.4 Изводи

В тази глава е предложена нова систематизирана методология за обработка, моделиране и интеграция на хетерогенни данни придобити от IoT устройства. Тя описва последователност от логически етапи и стъпки, което я прави лесна за разбиране и използване. Анализирани са предложените методи или подходи, които са най-подходящи за нейното изпълнение в зависимост от типа на разглеждания проблем и количеството и качеството на придобитите хетерогенни данни.

В резултат на това са направени следните заключения:

1. Методологията обхваща всички аспекти от работния процес – от дефиниране на проблема и поставяне на задачите, до интегриране на готово решение в работна среда.
2. Осъществяването на всеки един от етапите в методологията се извършва чрез решаване на редица по-малки задачи (стъпки), чрез което лесно може да се идентифицира наличието на проблем и да се предприемат навременни действия за неговото отстраняване.
3. Методологията е предназначена за работа с хетерогенни данни и предлага няколко метода за тяхното нормализиране и обединяване в единна хомогенна структура.
4. Методологията позволява създаване на автоматизирани процеси за работа с данни.
5. Методологията може да бъде прилагана от специалисти в различни области, работещи с хетерогенни данни.

Съдържанието на тази глава е отразено в публикациите:

[1] Dineva, K., Atanasova, T., Methodology for Data Processing in Modular IoT System. 22-st International Conference, DCCN 2019, Springer Nature Switzerland AG 2019, LNCS 11965, pp. 457–468, 2019. https://doi.org/10.1007/978-3-030-36614-8_35, **SJR:0.283**

[6] Динева К., Атанасова Т. Подходи и методи за анализ и обработка на данните в мониторингова система за пчелни кошери, Годишник на департамент „Телекомуникации“, НБУ, 2018, ISSN: 2534-854 X (online) No: 5, pp. 37-46.

[8] Dineva, K., Atanasova, T. OSEMN process for working over data acquired by IoT devices mounted in beehives. Current Trends in Natural Sciences, 7, 13, University of Pitesti, 2018, ISSN:2284-953X, 47-53.

Глава 3. Архитектура на модулна IoT система

В тази глава на дисертацията се представя пълното разработване на система за събиране, обработка и моделиране на данни в реално или през дефинирани интервали от време. Тя е съставена от отдалечени IoT устройства (Задача 2) и разпределено облачно базирано софтуерно приложение (Задача 3).

Извличането на данни се осъществява от модулна IoT устройства, които са интегрирани в пчелни кошери. Събраните данни се изпращат в софтуерно приложение, където данните се обработват, анализират и моделират, а резултатите могат да бъдат достъпни през потребителски интерфейси.

Мониторингът на био процесите е важен проблем и предизвикателство за изследователи и специалисти по информационни технологии. Пчеларството е един от подсекторите на селското стопанство, където могат да бъдат приложени техниките и методите за интелигентен мониторинг (Beecham, 2017). Интегрирането на информационните технологии в пчеларския процес може да подобри знанията на пчеларите за поведението на отделните пчелни колонии. Една от актуалните цели в областта на пчелния мониторинг е разработването на инструменти за непрекъснато наблюдение на пчелите в реално време, като се използват автоматични решения, избягващи излагане на пчелите на допълнителен стрес или непродуктивни дейности. Целта на тези технически средства не е да заменят, а по-скоро да подкрепят пчеларя.

3.1 Архитектура на модулна хардуерна IoT система

3.1.1 Изисквания към системата

Правилното определяне на нуждите на системата е най-важната стъпка преди планирането и разработката на хардуерната архитектура. По своята същност архитектурата се отнася до идентифицирането на физическите компоненти на системата и техните взаимовръзки.

В тази глава се проектира и изгражда IoT система, която поддържа постоянна връзка с облачно базиран контролен блок. Целта на системата е да събира данни от различни IoT устройства през дефинирани интервали или в режим на реално време. Данните могат да бъдат събирани, агрегирани, обработвани и съхранявани според нуждите и текущия режим на работа на системата и след това предавани чрез използване на различни методи към облачно (Cloud) базирана база от данни. Системата предоставя възможност за следене и докладване на поведението и състоянието си

(автодиагностика). Също така открива нередности и приема отдалечени команди, с които да осъществява промени в конфигурацията си, бизнес логиката и режимите си на работа. Системата е от отворен тип, като няма ограничение за броя на IoT устройствата, с които тя може да работи. IoT устройствата поддържат съвместимост и са лесни за подмяна и обновяване. Системата е с автономно захранване и може да работи без нуждата от пряка човешка дейност. Има изградено високо ниво на сигурност разгледано подробно в (Dineva K., 2019).

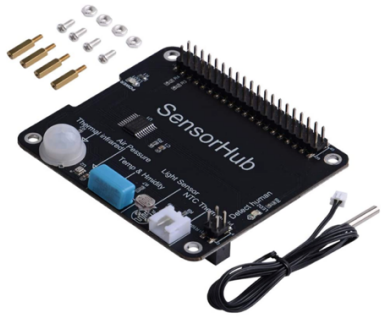
3.1.2 Хардуерни компоненти – съвместимост и заменяемост

Съвместимостта и лесната заменяемост на хардуерните компоненти е важна характеристика на системата. Това е и едно от основните изисквания към нея, което е спазено по време на процеса на проектирането й. Всички компоненти използвани в този дисертационен труд са специално подбрани по критерий за съвместимост, така че те лесно да бъдат подменени с други подобни при наличието на такава нужда. Пример за такава нужда може да бъде дефектирало устройство или нуждата от негово обновяване (upgrade). Внимание е отделено и на ценовия диапазон, в който се намират хардуерните компоненти с цел да не се променя себестойността на системата, ако се налага един компонент да бъде подменен с друг.

Системата използва следните четири основни категории хардуерни компоненти – сензори, логически блокове, комуникационни компоненти и захранващи елементи:

- **Сензори -** Сензорът е първичен преобразовател на физични или химични въздействия в удобен за използване електрически сигнал. Той обикновено е част от система и не работи самостоятелно. Група от сензори образуват сензорен хъб. Сензорният хъб представлява приложно ориентирани сензорни групи (ASSN (Application Specific Sensor Nodes)), които комбинират различни сензори и предоставят значително по точни и надеждни данни в сравнение с тези, които могат да бъдат получени от самостоятелни сензори. Това е от важно значение при външни условия, защото всеки вид сензори има своя собствена шумова характеристика, а чрез комбинирането им в хъб се компенсират недостатъците на отделните сензори. Също така те са значително по-малки и по-ефективни от обикновените решения.

Docker Pi Sensor Hub	Измерване на външна температура, термистор за откриване на температурен диапазон между -30 °C ~ 127 °C Вътрешна температурна детекция в диапазон -40°C

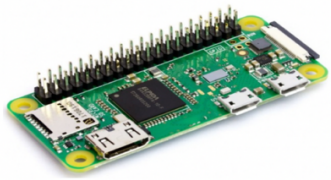
	~ 80℃. Автодиагностична температурна детекция DHT11 - 20℃ ~ 60℃. Измерване на влажност, обхват на откриване в диапазон 20% Rh ~ 95% Rh. Детекция на интензивност на светлината, обхват на откриване: 0Lux ~ 1800Lux. Детекция на налягане, диапазон на откриване: 300 Pa ~ 1100 hPa. Детекция на движение с максимален полезен диапазон 12 м.

Docker Pi Sensor Hub е устройство с групирани в него сензори за външна и вътрешна температура, атмосферно налягане, осветеност и реакция на движение, притежаващо възможност за автодиагностика. Устройството използва стандартен 40 GPIO pin header и I2C интерфейс за комуникация, което го прави съвместим с повечето логически блокове, които се предлагат на пазара. Притежава още 12 волтов 4 pin header, което му позволява да предава захранване и на други устройства, като по този начин лесно може да се образува група от устройства, които работят заедно и използват единен захранващ блок. За интеграцията му в разработваната в този дисертационен труд система се използва език за програмиране Python и пакет за разработка Python3-smbus. Отличителна черта на това устройство е способността му да следи стойностите на собствената си температура, като по този начин позволява да се прихващат гранични стойности при преминаването на които, устройството може да изпадне в състояние, в което да не може да продължи да функционира коректно. В такива случаи следва да му бъде изключено захранването, за да се запази целостта на системата.

- **Логически устройства -** Логическите устройства са едноплаткови компютри, върху които може да бъде инсталирана операционна система от типа на Linux. Също така предлага набор от GPIO (общо предназначение за вход/изход), което позволява да бъдат контролирани електронни компоненти за физически изчисления и да изследват устройствата от “Интернет на нещата”.

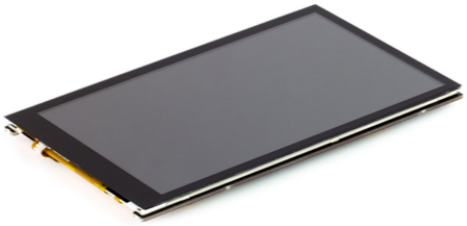
Разработваната IoT система в този дисертационен труд използва два вида логически устройства с цел оптимизация на консумация на енергия и пространството, което е необходимо на устройството за вграждане в пчелни кошери. Устройствата са напълно взаимнозаменяеми и с минимални усилия всяко едно от тях може да

изпълнява функциите на другото. Те са избрани преди всичко затова, че са компактни, мощни, надеждни и отлично разпознаваеми. Притежават отлична поддръжка и имат силно развита общност, където се обменят идеи и потребителят може да получи напътствия ако среща трудности. Мощният четириядрен процесор позволява изпълнение на сложни задачи свързани с работата с данни. Поддържането на различни комуникационни протоколи дава необходимата гъвкавост и разнообразие на подходи при реализацията на обмен на данни между устройствата.

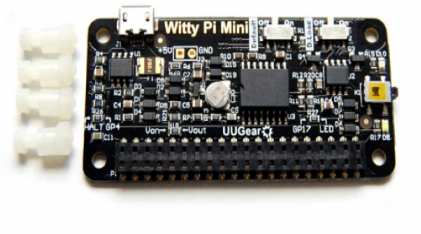
Raspberry Pi 4 	Broadcom BCM2711, четириядрен Cortex-A72 (ARM v8) 64-битов SoC @ 1.5GHz 2GB LPDDR4-3200 SDRAM 5.0 GHz IEEE 802.11ac безжична връзка, Bluetooth 5.0 2 USB 3.0 порта; 2 порта USB 2.0. Raspberry Pi стандарт 40-пинов GPIO 2 × микро-HDMI порта (поддържа се до 4kp60) 2-лентов MIPI DSI дисплей порт. 2-лентов MIPI CSI порт за камера. H.265 (4kp60 декодиране). OpenGL ES 3.0 графика. Слот за Micro-SD карта за зареждане на операционна система и съхранение на данни. 5V DC през USB-C конектор (минимум 3A). 5V DC чрез GPIO заглавие (минимум 3A). Работна температура: 0 - 50 градуса в околна среда. Консумират по-малко от 500mA.
Raspberry Pi Zero WH 	Bluetooth с ниска енергия (BLE), Bluetooth 4.1 802.11b / g / n безжична локална мрежа CSI конектор за камера Микро USB захранване USB порт в движение Мини HDMI 512MB RAM 40-пинов GPIO заглавка

- **Сензорен екран -** Сензорният екран е периферно устройство - дисплей, който реагира на докосване. Дисплеят играе роля едновременно на входно и на изходно устройство. Използваният в разработената IoT система сензорен екран е с висока резолюция и е удобен за визуализация и функционално управление на логически

устройства. Съвместим с почти всички логически устройства от типа на Raspberry PI 4, които имат 40 пин шина за комуникация.

HyperPixel 4.0 – High resolution Touch Capable TFT Display, 800x480 pixesl 	Висикоскоростен DPI интерфейс 800x480 pixels (~235 PPI) 18-bit дълбочина на цвят (262,144 цвята) 4.0" IPS (широкоъгълен, 160°) дисплей (86.4x51.8mm) 60 FPS кадъра в секунда Контраст: 500:1, Капацитивен тъч 40-pin female шина за комуникация Крепежни елементи за устойчиво закрепване

- **Управление на захранване -** За управление на захранването в системата е използвано последно поколение устройство, което притежава вграден часовник за реално време (RTC) и позволява управление на електрозахранването на логическия блок, създаването на различни режими на работа при различни нива на заряд на вградена батерия. Високата точност на часовника, под 160 секунди грешка на годишна база, улеснява синхронизацията на голяма група устройства, което е съществено предимство за отдалечени автономни системи. Вграденият кондензатор дава възможност за запазване на състоянието на вградения часовник за повече от 17 часа, което спомага за лесното преодоляване на ситуации на нарушено електрическо захранване към модула.

Witty Pi Mini: RTC + Power Management for Raspberry Pi 	Преобразувател позволяващ подаване на различни напрежения - DC 5.3V~26V Старт/Стоп ключ с едно натискане. Пълен контрол върху подавания ток към периферните устройства. Точен часовник без нужда от интернет. Възможност за създаване на график за старт/стоп на логическо устройство. Възможност за създаване на скрипт управляващ сложни Старт/Стоп сценарии. Възможност за прекъсване на захранване при дефинирани нива на заряд на батерия. Възможност за събуждане на устройство при достигане на нужните нива на заряд на батерията. Работна влажност - 10% ~ 90% (без конденз) Работна температура: -30°C ~ +80°C

	Захранване - 5V±5%, Размери - 65x56x19 mm

Всички описани до този момент компоненти са специално избрани и проучени с цел изпълнение на поставените изисквания, на които трябва да отговаря разработваната в този дисертационен труд хардуерна IoT система. На тази база са разработени прототипи на IoT устройства от модулен тип (**Приложение 1**), които са изградени от предложените хардуерни компоненти и покриват нужните изисквания.

- **Комуникации -** Системата се нуждае от WiFi изградена мрежа, за да може да се осъществява необходимият обмен на данни в различните посоки на комуникация между компонентите. Използвано е комбинираното решение на ALFA. При него всички необходими елементи идват, като единен пакет, което гарантира неговата оперативна съвместимост. Решението е изцяло насочено за употреба при външни условия, при които се намират пчелните кошери. Притежава отлични възможности за свързване към мрежите на мобилните оператори с цел да приема и предава данни чрез интернет. Рутерът, който е част от решението, образува WiFi мрежа с отлично покритие и притежава всички необходими протоколи за сигурност. Лесно се конфигурира. Модулният характер на решението е полезен за бъдещи обновявания, когато 5G мрежите ще станат стандарт при преноса на мобилни данни. В този случай ще се наложи да се смени единствено антената. Ниското захранващо напрежение (само 5 волта, както на всички останали устройства в системата) му дава допълнително предимство при изграждането на автономни и отдалечени WiFi мрежи. Решението може да бъде лесно монтирано и демонтирано при нужда от неговото преместване.

ALFA WiFi 4G Camp Pro 2 + Long Range Repeater 	Цялостно комбинирано устойчиво на вода и влага решение за външна употреба Удължител за рутер, 4G модем, Омни антена с 4G характеристика на насоченост. Тегло 993 гр. Размери - 93x70x26 mm SIM Slots: Mini - 2FF (включени адаптери). Омни антена - Tube-U4G модем и AOA-4G-5AM антена: 7.2” x 1.9” x 1" Мобилна комуникация – 802.11b: до 11 Mbps; 802.11g: до 54 Mbps; 802.11n: до 300 Mbps. Тип споделена комуникация - WiFi 802.11 b/g/n/. WiFi Сигурност - 64/128-bit WEP, WPA-PSK, WPA-EAP (PEAP/TTLS), WPA2-PSK, WPA2-EAP, WPA mixed, 802.1x аутетинкация. Портове – Micro USB & USB-A (за свързване на

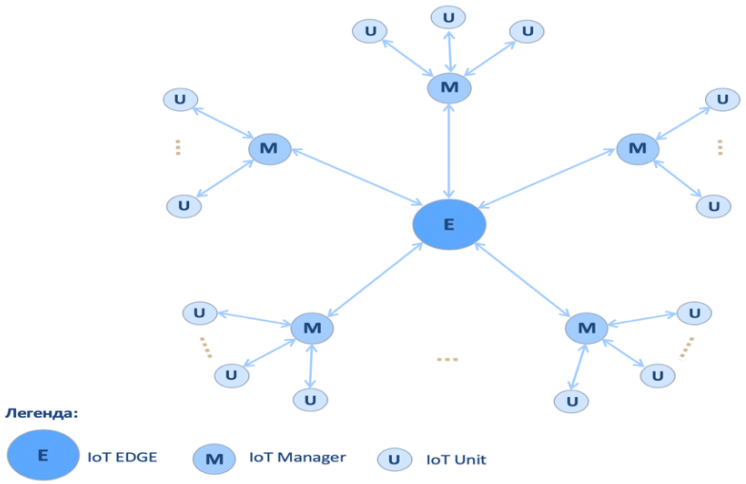
	модема и рутера). Работна влажност - 10% ~ 90% (без конденз). Работна температура: -20°C~+40°C Захранване - 5V±5%

Предложеният компонент не е задължителна част към разработваната IoT система. Неговото прилагане е нужно само в случаи, при които няма вече изградена Wifi мрежа.

3.1.3 Групиране и йерархия между хардуерните компоненти

Описаните хардуерни компоненти са групирани физически и логически по специфичен за разработената в този дисертационен труд система начин. Различните практически конфигурации на устройствата в системата позволяват да бъдат извършени разнообразни групирания, но тяхната логическа структура е постоянна. Физическото групиране на компонентите е продиктувано от конкретно разположение на кошерите вътре в пчелина. Известно е, че пчелините се различават по брой на пчелни кошери в тях и тяхното конкретно разположение. То зависи от вижданията на потребителя, както и от характеристиките на терена, където е разположен пчелина.

Логическото групиране от своя страна е преди всичко зависимо от типа роля, която компонентът изпълнява в работата на системата. За нуждите на системата е предложена логическа схема от тип Снежинка (Snowflake) (фиг. 3.1) за групиране на IoT устройства. Тя следва строга йерархична структура, която показва ясно логическата принадлежност и свързаност на устройствата, което спомага за по-лесната поддръжка на системата, добавянето на нови или премахването на съществуващи устройства в нея.



Фигура: 3.1: Логическа схема на групиране от тип Снежинка (Snowflake)

В системата са изградени следните видове участници:

- **IoT Unit** - Съвкупността между групираните сензори и логическият блок в този дисертационен труд се нарича IoT Unit. Това е IoT устройство, което има пълен контрол над работния процес, за който отговаря. Притежава необходимият капацитет, за да бъде пълноценен участник в комуникационни вериги и е способен да контролира и спазва различни работни режими свързани с оптимизация на консумацията на електрическа енергия.

В системата IoT Unit е съставен от устройството Sensor Hub с групираните в него сензори, което е предназначено за измерване на параметрите вътре в пчелния кошер. За да може това устройство да бъде пълноценен участник в системата и да взаимодейства с останалите компоненти е необходимо да бъде монтирано към логическо устройство (Raspberry Pi), което има хардуерен капацитет да изпълнява конкретната си роля, като участник в системата. IoT Unit е устройство, което не предава данни извън границите на системата. То има възможност да комуникира само с един участник, което го прави крайно вътрешно устройство.

- **IoT Manager -** IoT Manager представлява хардуерно устройство, което притежава капацитета да служи като мениджър на няколко на брой IoT Units. Всеки мениджър е пряко отговорен за наблюдението на работния процес на IoT Units. Той контролира работата им чрез изпълнение на подходящ софтуер, който е инсталиран в неговата трайна памет. Правилното изпълнение на този софтуер се контролира от Линукс операционна система. IoT Manager още е отговорен за временното съхраняване на данните от сензорите в IoT Units и тяхната първична обработка и филтриране - при наличието на такава необходимост. Пример за такава обработка може да бъде, ако сензори за температура имат стойности в различен формат (°F $\rightarrow$ °C), то тези стойности се конвертират в един формат, за да се получи хомогенност на данните. При зададено разписание или постъпване на такава заявка, той изпраща необходимите данни към следващите участници в системата. Чрез него се предават командите към крайните вътрешни устройства и пак чрез него се докладва статуса на всяко едно устройство към останалите участници в системата. IoT Manager не може да общува с устройства извън границите на системата. Той може да комуникира както с IoT Units, така и с Edge и затова се наричано междинно устройство.

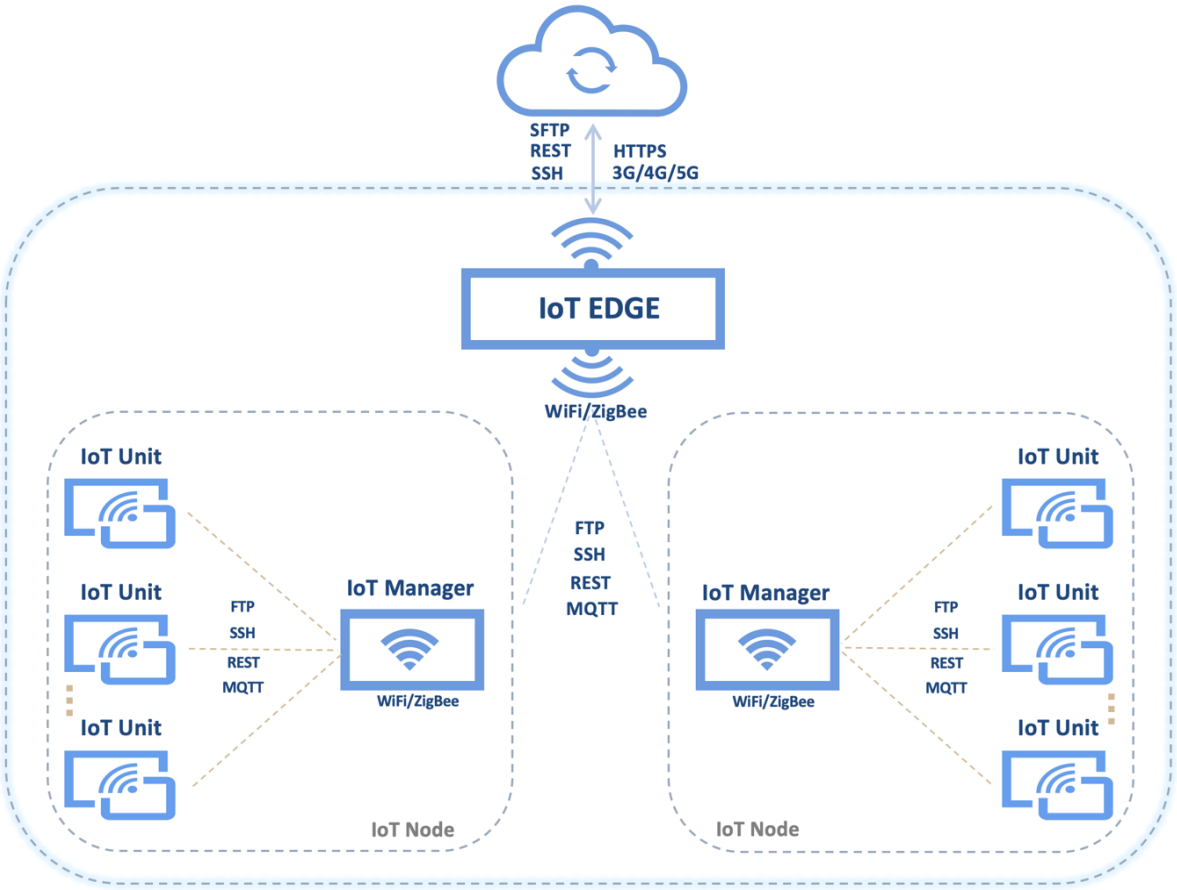
- **IoT Node -** Съвкупността от всички IoT Units, за които е отговорен даден IoT Manager, се нарича IoT Node в този дисертационен труд. Той може да притежава уникални настройки и правила, които участниците в него (IoT Units, IoT Manager), като

по този начин да бъдат изпълнени дейностите в системата, за които този IoT Node отговаря.

• **IoT Edge -** Това е хардуерно устройство, което е предназначено да изпълнява ролята на медиатор. Чрез него се управлява цялата комуникация на системата с Облачната среда. Затова IoT Edge се нарича външно крайно устройство за системата.

Всички IoT Nodes изпращат събраните сензорни данни и информация за състоянието си към IoT Edge. Също така IoT Edge има възможност самостоятелно да вземе данните от IoT Nodes. Това е направено с цел да се повиши стабилността при работа на системата, като се поддържа повече от един комуникационен канал между участниците в нея. От своя страна, IoT Edge на базата на дефинирани от потребителя правила, изпраща събраните данните към Облачната среда (фиг. 3.2).

IoT Edge може да получава команди от Облачната среда, тези команди могат да бъдат адресирани пряко към него, или към конкретен IoT Node. В този случай, Edge препраща информацията към конкретния IoT Node.



Фигура 3.2: Архитектура на разработената модулна IoT система

Предложеното в дисертацията ясно разграничение на комуникацията в хардуерната система от тази на системата в Облачната среда е от важно значение,

защото при възникване на нужда от промени, в която и да е от двете, няма да засегне другата. Добавянето или премахването на IoT Nodes от системата, няма да промени установените комуникационни правила. Сигурността на системата значително се повишава, защото Edge е единствена точка на достъп с външния свят.

3.1.4 Комуникация

Комуникацията в хардуерната система се основава на безжична свързаност (WiFi). Това е технология, която използва радио сигнали, за да предава информация между устройствата, които я поддържат. На база описаните системни изисквания в 3.1.1, се определя, че WiFi е най-подходящата технология за осъществяване на комуникация между устройствата. Тази технология позволява лесна работа с различни протоколи за обмен на данни, което е от съществено значение за отдалечени системи, които трябва да имат свойството да бъдат управлявани, чрез команди от разстояние. WiFi позволява използване на силно криптиращи алгоритми, с които сигурността на системата да бъде от най-високо ниво. WiFi може да поддържа таблица с локални адреси, което я прави много удобна за изграждане на вътрешни мрежи, където могат да се прилагат разнообразни правила за комуникация.

Съществуват и други подходи и технологии, които позволяват общуване на устройствата помежду им. Най-популярни от тях са ZigBee и Z-Wave. Всички те имат своите предимства и недостатъци в зависимост от техните характеристики и конкретни нужди, при които се налага използването им. В **Приложение 2** е изложен разработеният в дисертацията програмен код за осъществяване на комуникация чрез използване на ZigBee технология, което показва възможността на системата да използва и други подходи за обмяна на данни едновременно с WiFi.

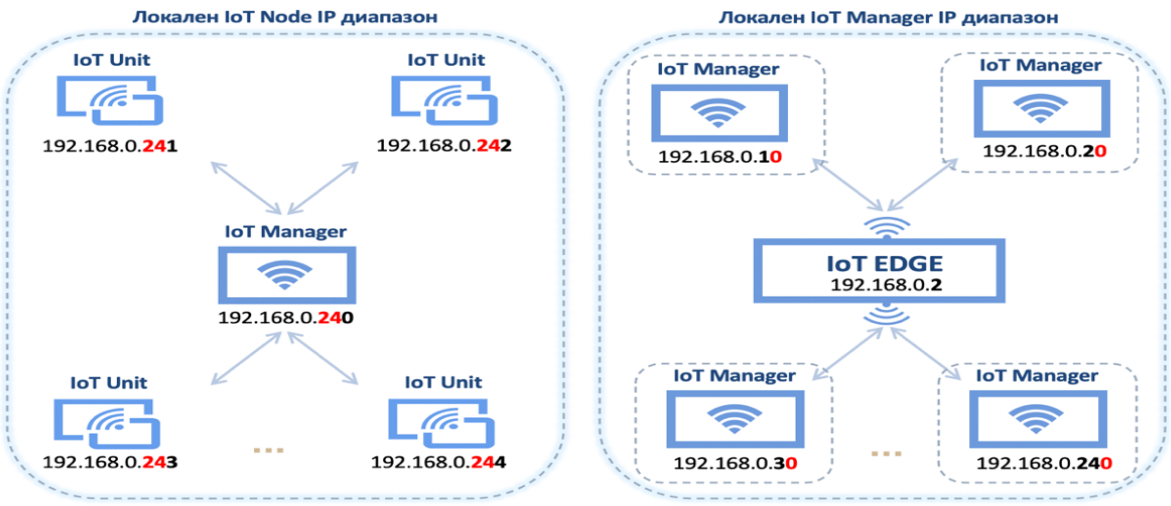
Системата се характеризира с централизирана топология в центъра на която се намира рутер, който още изпълнява ролята и на шлюз по подразбиране (Default Gateway). В такава топология, той свързва системната мрежа с външни мрежи и интернет. Топологията определя и типовете комуникации, от които системата се нуждае – устройство към устройство, устройство към Edge, Edge към Cloud.

Всички устройства, участници в системната мрежа, притежават IP адрес, който е част от адресното пространство дефинирано от подмаска 255.255.255.0 (Microsoft, 2019). Всички устройства имат статични IP адреси, които се указват при процеса на тяхната първоначална инициализация. За добрата организация на устройствата и семантична яснота на логическата връзка между всяко устройство с неговия IP адрес е

предложен **нов метод** за комуникация между устройствата, основан на IP адресация сегментирана в конкретни диапазони. Като централно устройство и шлюз по подразбиране, рутера винаги има IP адрес 192.168.0.1. Всички адреси в пространството 192.168.0.**2** – 192.168.0.**9** са резервирани само за външни крайни устройства, като IoT Edge и се наричат Edge IP диапазон (Edge IP Range). IP адреси отговарящи на шаблона/правилото, 192.168.0.**[1-24]**0 са резервирани за IoT Managers и се наричат Managers IP диапазон (Managers IP Range). Адреси в пространството 192.168.0.**250** – 192.168.0.**255** са резервирани за специални случаи, не се използват в системата и се наричат Резервиран IP диапазон (Reserved IP Range). Всички останали свободни IP адреси, които не са засегнати от описаните досега правила, са резервирани само за IoT Units. За всяко едно такова устройство е изградена ясно дефинирана конвенция, която е обяснена със следващите примери:

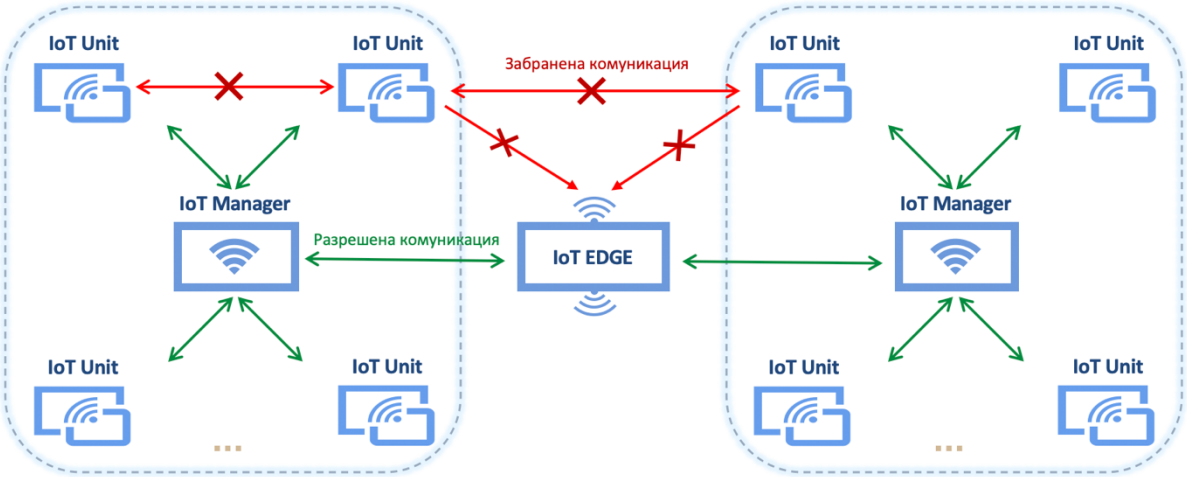
- 192.168.0.**14** – IoT Unit с този IP адрес е част от IoT Node, който има IoT Manager с присвоен IP адрес 192.168.0.**10**. Този IoT Unit е четвърто поред устройство в йерархията на този Node, а всички IP адреси в диапазона 192.168.0.**[10-19]** са дефинирани, като Локален Node IP диапазон **10** (Local Node IP Range 10).
- 192.168.0.**249** - IoT Unit с този IP адрес е част от IoT Node с IoT Manager, който има присвоен IP адрес 192.168.0.**240**. Този IoT Unit е девето поред устройство в йерархията на този Node, а всички IP адреси в диапазона 192.168.0.**[240-249]** са дефинирани, като Локален Node IP диапазон **240** (Local Node IP Range 240).

Тази конвенция определя, че максималният брой устройства в един IoT Node е 10, като едно от тях е логически блок, а девет са IoT Units (фиг. 3.3).



Фигура 3.3: Метод за комуникация чрез IP адресация между участниците в разработената IoT системата

Така създадената в този дисертационен труд системна мрежа и наложената конвенция за IP адресация, определя следните посоки комуникации, които се извършват в системата (фиг. 3.4):



Фигура 3.4: Комуникация между участниците в разработената IoT системата

- **IoT Node**

Комуникацията във всеки един IoT Node е от тип устройство към устройство (device-to-device). Всички IoT Units в даден IoT Node имат строго наложени правила от защитната стена (Firewall). При извършване на първоначална инициализация, всички входящи заявки са забранени. След това входящи заявки (requests) се разрешават само за тези, идващи от IP адреса на IoT Manager на този Node, и IP адреси от Edge IP диапазон. Изходящи заявки са разрешени само към IP адреса на IoT Manager на този Node. Не се предвижда IoT Units да могат да общуват помежду си, както в рамките на своя Node, така и с които и да е други IoT Units. По този начин се гарантира строга йерархичната структура на комуникация в системата и се дава възможност за получаване на директни команди за преконфигурация на даден IoT Unit в случай, че IoT Manager на неговия Node е спрял да функционира по някаква причина. В този случай, неговият IP адрес може да бъде променен от разстояние и след подадена команда за рестартиране, устройството може автоматично да бъде присъединено към друг IoT Node.

- **IoT Manager – IoT Edge**

Комуникацията е от тип устройство към gateway (device-to-gateway). Всеки IoT Manager задължително поддържа двупосочна комуникация с IoT Edge. Затова при неговата инициализация се установяват правила разрешаващи двупосочна комуникация само с IP адреси, които са зададени на устройства принадлежащи на Локален Node IP диапазон, и на IP адреси в Edge IP диапазон. В разглеждания в

момента случай, системата има само един IoT Edge, което означава, че е наличен само един IP активен адрес в Edge IP диапазон. В случай, че в системата има повече от един IoT Edge, тогава лесно могат да бъдат зададени конвенции, които да удовлетворяват правилото, че един IoT Node, може да общува само с един Edge.

От съществено значение е да се има предвид, че при изпращането на данните от IoT Manager към IoT Edge, задължително се изпраща и мета информация, която удостоверява броя на устройствата в този Node и тяхното хардуерно състояние. Тази информация се съхранява в трайната памет на Edge под формата на документна база от данни. Така се гарантира възможността да се направи моментален преглед на текущо състояние (Snapshot) на системата с цел анализ на активните участници в нея. Една от ползите на този анализ е в случаите, когато трябва да бъде присъединено ново устройство към системата. В анализа ясно се виждат IP адресите, които са резервирани към момента за активни устройства и следвайки наложената конвенция, лесно се установява правилния IP адрес за новото устройство. То може да бъде инициализирано предварително и при вграждането му в системата, да започне да работи веднага без да е нужно да се правят промени по настройките на което и да е друго активно устройство.

- **IoT Edge – Cloud**

Комуникацията е от тип устройство към Cloud (device-to-cloud). Като външно крайно устройство за системата, IoT Edge изпълнява важна роля на медиатор. Всички външни за системата комуникации се контролират от него. Това определя високото ниво на сигурност, което той трябва да притежава. Всички външни заявки към IoT Edge, които също така са външни за системата са криптирани чрез използване на TLS/SSL. Всички изходящи заявки, които са външни за системата също трябва да са криптирани.

Edge събира информацията, която му е предоставена от IoT Managers, и я съхранява до момента за нейното изпращане в Облачната среда. Edge изпълнява важната роля да редуцира и трансформира информацията в нужния вид и обем, като по този начин значително намалява размера на изходящия интернет трафик от системата. А това е важно предимство при надеждната работа на отдалечени IoT системи. За тази цел Edge поддържа протоколи, като HTTPS, SFTP, SSH и MQTT.

IoT Edge приема команди от облачна среда чрез използване на отделен комуникационен канал, който е различен от този, по който се предават данните. С тези команди, Edge може да променя поведението и начина на работа на конкретен IoT Unit,

IoT Manager или на цял IoT Node, прави промени в техните конфигурационни файлове и рестартира, чрез използване на модула за управление на захранването на всяко едно устройство. Това е огромно предимство пред останалите системи, които не могат да си позволят функционалности на обновяване и актуализация (Carter J., 2016), които поради хардуерната си специфика, не могат да разчитат на софтуерно базирани инструкции за иницииране на промяна в тяхното поведение.

Всички хардуерни компоненти са избрани така, че да отговарят на функционалните изисквания на заданието и същевременно да са лесно заменяеми с техни аналози.

Предложените решения са с оптимизирана консумация на електроенергия благодарение на модули за управление на захранването, които разрешават Sleep състояние за всяко едно устройство. Вграденият часовник притежава висока точност и прави лесна синхронизацията на устройствата.

Наличието на Линукс операционна система в устройствата позволява използване на езика за програмиране Python, чрез който се задават за изпълнение всички функционални изисквания към поведението на устройствата - това позволява да бъдат взаимнозаменяеми.

IoT Manager може лесно да бъде конфигуриран да изпълнява ролята на външно крайно устройство Edge. Също така позволява лесното добавяне на теоретично неограничен брой други устройства (IoT Units, IoT Managers, Edge), които да бъдат пълноценни участници в системата. Повишените софтуерни възможности позволяват лесното конвертиране на разнородни по тип стойности данни, така че те да имат накрая необходимия хомогенен формат.

Всички устройства поддържат възможност за комуникация с устройства от тип микроконтролери (Arduino), което дава възможност за съвместимост на системата с други IoT сензорни модули, които вече съществуват, но имат хардуерно ограничени възможности. Това позволява системата да се използва за надграждане над други подобни системи, които са били хардуерно или софтуерно ограничени при разширяване на функционалните им възможности.

Възможността на системата да използва WiFi, ZigBee или Z-Wave я прави платформено независима при избор на подходящата технология за комуникация с отдалечени системи и облачни среди.

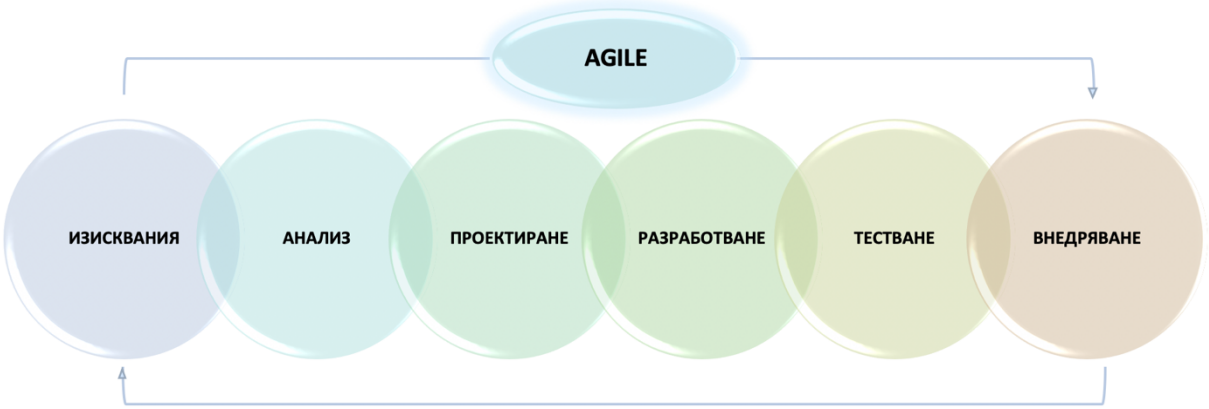
Системата притежава високо ниво на сигурност, като се:

- използва единствена точка за достъп, която общува с интернет само чрез криптирани TLS/SSL комуникационни канали (Преодоляват се проблемите с липсата на криптиране и несигурен уеб интерфейс (описани в глава 1)).
- използват строго рестриктивни правила за комуникация между устройствата (Преодолява се проблема с имплицитното доверие (описан в глава 1)).
- използват устройства, които позволяват работа на Линукс операционна система и поддържат функции за обновяване и актуализация (Преодоляват се проблемите с паметта на устройствата, поддръжката и актуализациите на ОС (описани в глава 1)).

3.2 Архитектура на облачна софтуерна система

Софтуерната архитектура се отнася до основните структури на софтуерната система и дисциплината на създаване на такива структури и системи. Всяка структура съдържа софтуерни елементи, отношения между тях и свойства както на елементи, така и на отношенията.

При разработването на софтуерната система в тази дисертация е следвана утвърдената методология – Agile (фиг. 3.5). Нейната употреба предоставя редица ползи, като определя: точна последователност на дейностите, добро прогнозиране на сроковете за започване и приключване на всеки етап, ясна информираност за минали, настоящи и следващи задачи, откриване на възможни проблеми в ранен етап, лесно проследяване на дейностите и използваните технологии и методи за изпълнението им, определяне на възможността на задачите за паралелно или последователно изпълнение и други.



Фигура 3.5: Agile методология за разработка на софтуерна система

Agile е гъвкава методология, при която разработката на софтуерна система се изгражда на итерации, при които изпълнението на етапите се повтаря последователно (Cohn, 2019) (Manel D., 2019). Итеративните версии дават възможност за ранно откриване и отстраняване на греши и получаване на междинни резултати, като така значително се подобрява ефективността на работния процес.

В този дисертационен труд разработката на системата се изпълнява в последователност от 6 етапа следвайки Agile методологията:

1. Изисквания – уточняват се изискванията, на които системата трябва да отговоря. Това е важен етап от целия процес, защото в него се дефинират макро рамките на системата.
2. Анализ и изисквания – уточняват се функционални, нефункционални и технически изисквания, инфраструктура, програмни езици, библиотеки и инструменти, срокове на изпълнение;
3. Архитектура и проектиране – съставят се указания за проектиране, архитектура, преглед на проектни решения, проверка и контрол върху проектирането на сигурността;
4. Разработване – разработване на отделните части в указаната последователност;
5. Тестване – тестване при разработване, функционално тестване, тестване при интегриране, системно тестване;
6. Внедряване и поддръжка – извършва се проверка и контрол върху процеса на внедряване и последващо обновяване;

3.2.1 Сървърна облачно базирана MSA архитектура

3.2.1.1 Определяне на изискванията

Ясното и правилно дефиниране на изискванията към разработката на софтуерна архитектура е първата и основополагаща стъпка в целия процес. На база на тези изисквания се определя необходимия тип архитектура и всички необходими елементи в нея. Като главни изисквания към разработваната сървърна архитектура в този дисертационен труд могат да бъдат определени:

- Услуги – ясно разграничени услуги, всяка услуга да изпълнява определена задача без да се намесва в задачите на другите услуги.
- Поддръжка – лесно отстраняване на грешките чрез изолация на неизправностите, този процес да не възпрепятства работата на цялата система.

- Гъвкавост – лесно добавяне или отстраняване на елементи от системата без това да наруши нейната цялост.

3.2.1.2 Анализ

Ключов момент при разработката на софтуерната система е правилното анализиране и определяне на функционалностите и необходимия технологичен стек за изпълнението им.

Очакваните поддържани функционалности от системата са:

- Поддържане на потребители - създаване, изтриване и редактиране на потребители и всички данните свързани с тях;
- Работа с данни - внедряване и контролиране на устойчиви процеси по придобиване на хетерогенни данни от крайни отдалечени устройства. Трансформация на нови данни и на вече съществуващи такива;
- Моделиране - внедряване, употреба и мониторинг на обучен модел за машинно обучение;
- Крайни устройства – управление на процесите на работа на крайните устройства чрез изпращане на команди към тях от упълномощени потребители;
- Журнал - записване на събития в системата, които показват кога и как те са настъпили и резултата след тяхното настъпване.

На база описаните функционалности се определя технологичния стек, чрез които ще се постигне тяхното изпълнение. Технологичният стек е съвкупността от всички инфраструктурни и софтуерни инструменти и технологии, които се използват в процеса на създаване на софтуер. Избирането на правилен технологичен стек е от голямо значение за получаването на успешен краен резултат. За постигане на целта и изпълнение на задачите в този дисертационен труд са използвани следните технологии:

- ***Инфраструктура*** – системата използва облачно базирани виртуални сървъри (VPS - Virtual Private Server) с операционна система Дебиан 9 (R. Hertzog, 2015). Дебиан е Линукс операционна система с отворен код, която е популярна при разработването на софтуерни системи от различен тип. Предоставя сигурност и надеждност от високо ниво (при правилни настройки на системата). Свързването към отдалечените виртуални сървъри става чрез използване на SSH (Secure Shell) протокол.
- ***Програмни езици*** – системата изпълнява задачи от разнороден характер. Различните програмни езици са подходящи за конкретни задачи (Belani, 2020). Избраните програмни езици са:

- Python – скриптов език за програмиране от високо ниво с общо предназначение.
- C# - обектно ориентиран програмен език от високо ниво с общо предназначение създаден от Майкрософт, част от платформата .NET.
- Typescript - скриптов програмен език с отворен код създаден от Майкрософт. Построен върху JavaScript, като добавя редица подобрения, които правят процеса на разработка значително улеснен. В системата се използва при създаването на потребителските уеб интерфейси (UI).

- ***Технологии и библиотеки*** – в системата се използват разнообразни (безплатни) технологии и библиотеки: Numpy, Pandas, Scikt-learn, FastAPI, SqlAlchemy, Asp.NET Core, SignalR, Ocelot, Angular, Kafka, MongoDB, Postgres и други.
- ***Инструменти за разработка*** (IDE) – продуктите на JetBrains (Kovaleva, 2019) са от най-високо ниво и позволяват решаването на всички задачи поставени пред системата.
  - PyCharm – интелигентна среда за разработка на Python базирани уеб и ML приложения.
  - Raider/Visual Studio – интелигентна среда за разработка на C# базирани приложения.
  - DataGrip – интелигентна среда за работа с бази от данни.
  - Visual Studio Code – интелигентна среда за разработка на софтуерни приложения, разработена от Microsoft. Използва се за разработването на потребителския интерфейс на системата.

3.2.1.3 Архитектура и Проектиране

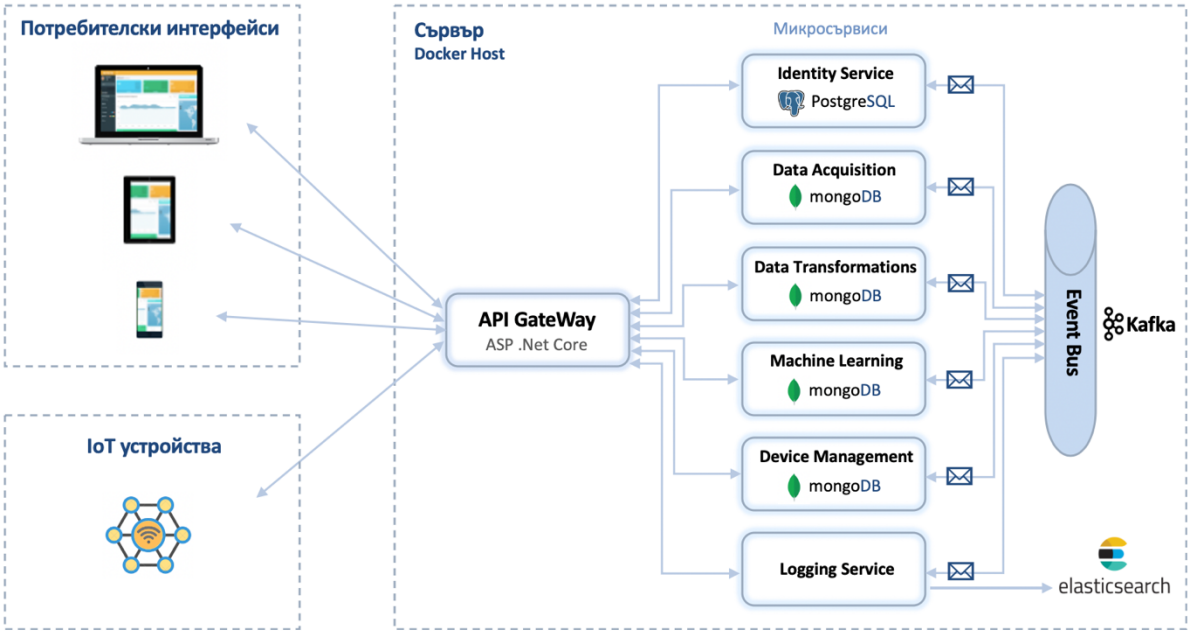
Проектирането се отнася до създаването на софтуерната архитектура. Тя от своя страна определя цялостната структура на софтуерната система, софтуерните компоненти, свойства и отношения между тях.

Има различни видове софтуерни сървърни архитектури (Saraswathi, 2020) – Single tier, Single layer, Traditional n-layer, Microservice architecture, DDD (Domain Driven Design), EDA (Event Driven Architecture) и други. Всички те имат своите предимства и недостатъци. След определяне на функционалните изисквания и технологичния стек за разработване на системата в този дисертационен труд се определя, че най-подходяща е архитектура основана на използването на микросървиси (Microservice architecture - MSA). Тя позволява създаване и поддържане на дистрибутирани софтуерни системи, работа с различни бази от данни и създаване на

самостоятелни потребителски интерфейси, които могат да общуват със сървърната част. Микросървисната архитектура е с висока сложност (Waller M., 2019). Изискват се специфични знания и възможности, за да бъде имплементирана по правилен начин в реална среда. По същество тя представлява съвкупност от няколко или много самостоятелни софтуерни приложения, като всяко едно от тях изпълнява своите специфични задачи. Те могат да общуват помежду си и да обменят информация.

За разлика от монолитните и други сървърни архитектури, микросървисната притежава редица характеристики, които й дават стратегическо предимство пред останалите.

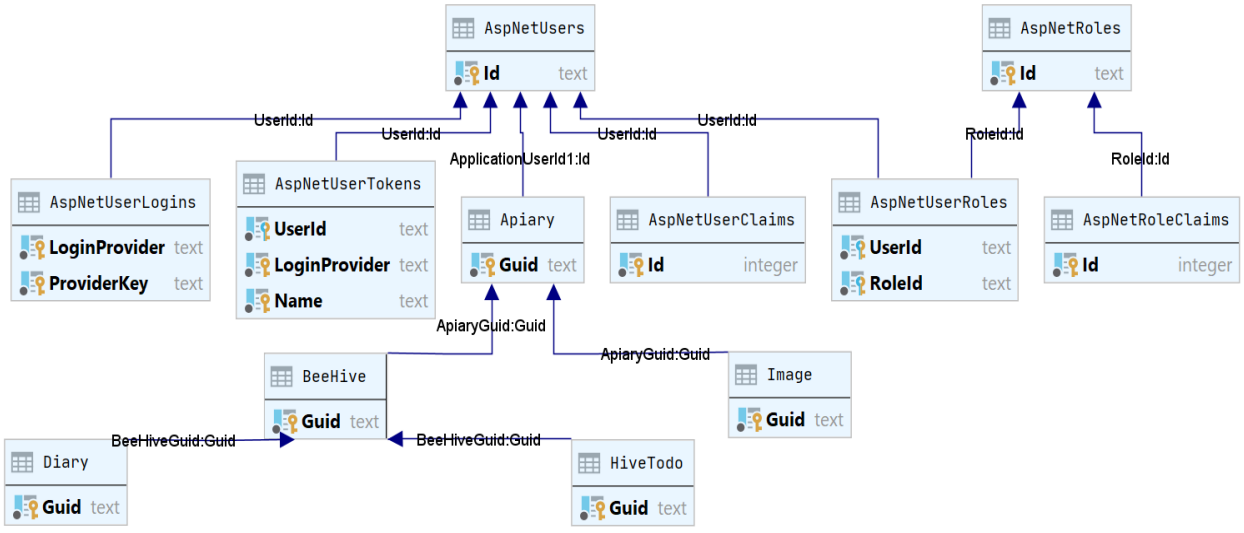
- Редуциране на сложността – по същество, бизнес логиката на софтуерната система е разпределена между необходимият брой микросървиси, което води до липса на зависимост между отделните елементи в системата. Всеки един представлява самостоятелно приложение и носи отговорност за конкретен аспект от работата на системата. По този начин цялата сложност в имплементирането на бизнес логиката на системата е разпределена на малки, конкретно дефинирани задачи, които са делегирани на конкретен микросървис.
- Всеки микросървис се разгръща и се прилага самостоятелно. Това е важно, защото изискването за 24/7 достъпност на системата предполага тя да не бъде спирана по време на обновяване или процес на поддръжка.
- Всеки микросървис се скалира индивидуално. Това е задължителна характеристика за дистрибутирани софтуерни приложения.



Фигура 3.6: Дизайн на предложената и реализирана в системата MSA архитектура

За изпълнение на поставените задачи е приложен **нов подход** за организация на услуги за интелигентна обработка на данни, състоящ се от следните реализирани микросървиси (Фиг. 3.6):

Identity Service отговаря за всички операции, свързани с потребителя, като създаване на нов, редакция на съществуващ, изтриване, смяна на потребителска парола, възстановяване на забравена парола, задаване на потребителски роли и други. Всички тези действия, които са възможни към всеки един потребител позволяват той да бъде строго идентифициран, да му бъдат зададени нужните права на достъп до системата и да се осигури нужното ниво на защита на собствените му данни. Използваната технология е Asp.NET Core 3.1 (Microsoft, 2016), която е последна стабилна версия към този момент. За база от данни се използва Postgres, която работи в Линукс работна среда. За връзка между двете технологии се използва Entity Framework Core, който представлява ORM (Object-relational mapping) за работа с бази от данни и приложения използващи Asp.NET Core. EF Core, което позволява създаването на mapping, където софтуерни обекти отговарят на конкретни таблици в базата от данни. Това дава нужното ниво на абстракция и прави процеса на разработване и поддържане на софтуерно приложение много по-лесен и разбираем за разработчиците.



Фигура 3.7: Схема на релационна база данни за разработения Identity Service

На фиг. 3.7 е представена обща схема на релационната база данни за Identity Service. Тя съдържа 12 таблици с връзки между тях от тип 1:1 и 1:n (пълната схема на релационната база може да бъде намерена в **Приложение 3**).

- ***Data Acquisition*** отговаря за всички data pipelines (съвкупност от всички предприети стъпки в процеса на устойчиво преместване на данни между отдалечени физически различни локации), чрез които системата събира данните от отдалечени IoT устройства. Системата има способност да инициира процес за самостоятелно вземане на данните чрез SFTP (Secure File Transfer Protocol), позволява данните да бъдат вкарани в системата чрез използване на REST (Representational state transfer) и/или Streams и двупосочен обмен на данни за размяна на съобщения (Naik N., 2017). След като данните са успешно записани в системата и са валидирани, като коректна транзакция, друг микросървис бива уведомен, че са налице нови данни и той може да започне работа с тях. Програмният език е Python, а използваните технологии са Numpy, Pandas, Kafka, FastAPI, SqlAlchemy (Bell J., 2020) (McKinney W., 2012). Всички данни, влезли в системата се записват в MongoDB база от данни. Тя е избрана защото е база, която може лесно да се адаптира ако има промяна в схемата на данните – нещо, което е често явление при работата с неструктурирани хетерогенни данни.
- ***Data Transformations*** носи отговорност за всички трансформации върху новопостъпилите сурови данни докато те придобият нужният вид, подходящ за анализ и прилагане на модели за машинно обучение върху тях. За постигане на устойчиви резултати в процеса на предсказване от модела за машинно обучение, всички трансформации върху данните са консистентни с тези, които са предприети при процеса на обучение на модела за машинно обучение (описани в методологията в Глава 2). Това е важно условие, за да се постигне максимална достоверност на предсказаните резултати. Поради тази причина в структурата на този микросървис е създаден микро слой, който лесно да може да бъде променян, ако се налагат промени в процеса на трансформация на входящите данни. Програмният език е Python. Технологичният стек включва библиотеките Pandas и Numpy, а за база данни се използва MongoDB.
- ***Machine Learning Service*** отговаря за прилагането на готови, тренирани модели за машинно обучение за предсказване на стойности и вероятни събития. В модела се подават само нови и напълно трансформирани данни. Това е изключително важно условие, за да се постигне скорост на предсказване в почти реално време. Микросървисът има модулна структура, което прави лесно промяната на съществуващ модел с друг, както и внедряването на нов модел в този сървис. Микросървисът не държи информация за статуса на данните. Той работи, като на подадени данни, дава резултат без да записва в база от данни нито броя на извършените предсказвания, нито

резултатите от самото предсказване. Програмният език е Python, технологичният стек включва библиотеката scikt-learn и streaming платформата Kafka (Moon J., 2020).

- ***Device Management Service*** е отговорен за предаването на команди към IoT устройствата от потребителя. Тези команди се инициират от потребителя чрез потребителския интерфейс. Този сървис използва отделен комуникационен канал, за да се осигури автономност и независимост при работата на системата. Дори и комуникационните канали отговарящи за събирането на данни да са максимално натоварени, чрез използването на специализиран комуникационен канал се гарантира сигурност на потребителя, че подадените команди ще пристигнат максимално бързо до целевите IoT устройства.
- ***Logging Service*** отговаря за събирането и анализа на събитията, които са настъпили по време на работата на системата. Това е много важна дейност за всяко едно софтуерно приложение, което има изискване да бъде надеждно и устойчиво. Данните събрани от този сървис се използват за 4 основни нужди:

- Софтуерна поддръжка – събраните данни могат да показват, къде в софтуерните части на системата има проблеми и от какъв тип са те.
- Системна поддръжка – чрез анализ на събраните данни се вижда ясно, къде системата работи бавно или неустойчиво и направените изводи дават важни насоки към подобряване на цялостната архитектура на системата.
- Сигурност – с централизирано събиране на данни за работата на системата, те могат да бъдат филтрирани и записите, които се отнасят пряко към сигурността на данните в системата могат да бъдат бързо анализирани, за да бъдат предприети своевременни мерки. От важно значение е, че тази информация може да бъде проследена чрез времевата линия на записите. Това от своя страна дава важни изводи за векторите на атаки, които са били използвани в процеса на атака върху системата.
- Предсказване – постигането на бързина в процеса на предсказване от модели за машинно обучение е важна характеристика на системата. Анализът на записите ясно показва, кои модели се използват най-често, и кои модели работят с най-големи обеми от данни. Направените изводи са важни при оптимизацията, която се прави при работата на микросървис отговарящ за извършване на задълбочени анализи (analytics).

API Gateway Service се използва за разделяне на потребителските интерфейси от системата. В архитектура изградена чрез микросървиси, една клиентска заявка обикновено се нуждае от информация от няколко микросървиси. Тази заявка не трябва да се извършва директно към всички сървиси, защото клиентът трябва да поддържа

едновременно няколко комуникационни канала, а това води до значителни усложнения при неговата работа. Освен това, клиентът трябва да се идентифицира към всеки един сървис, което прави всяка една система нестабилна и несигурна. API Gateway поема всички тези отговорности върху себе си – клиентът се идентифицира само на едно място в системата и всички заявки за данни се агрегират на едно място. Използваните технологии са Asp.NET Core, Ocelot, IdentityServer4, Entity Framework Core. Базата от данни е Postgres, схемата е релационна.

Event Bus модул се грижи за предаването на съобщенията до нужния получател (Alekseev I., 2015). Използваният Event Bus в системата е Apache Kafka (Hesse G., 2020). Kafka е система за съобщения, но за разлика от други системи, тя е пригодена за случаи на използване при нужда от висока пропускателна способност, при която е необходимо да се преместват големи количества данни по мащабируем и несъвместим начин.

Когато един микросървис желае да комуникира с друг или други сървиси, той изпраща съобщение до Event Bus. Всички съобщения в системата, които използват Event Bus представляват имплементация на модел на команда (command pattern), която е вид модел на поведение (behavioral pattern). При нея в един обект се енкапсулира както самата команда, така и цялата необходима информация, нужна за изпълнението на командата до нейния край. Информацията включва инициатора на командата, името на метода, който трябва да се изпълни и аргументите, от които този метод се нуждае. Към командата още може да бъде включен *callback* метод, който да бъде извикан при завършването на изпълнението на командата. Този метод може да инициира нова команда, а може и само да уведоми част от системата за промяна в статус на данни или сървис.

Поддържаните типове комуникации от Event Bus в системата са:

- *Публикуване – Абониране (Publish - Subscribe)* – Сървисите могат да се абонират за определени комуникационни канали. Всеки път, когато сървис публикува съобщение до Event Bus, то бива доставено на всички сървиси, които са се абонирани за него. Този тип комуникация се използва в Data Acquisition сървис, при успешно получаване на данни се изпраща две съобщения – за успешно свършена работа и за наличие на нови данни. Първото се изпраща до Logging Service, второто до Data Transformation Service. И двете съобщения се изпращат през Event Bus, сървисите получават само това съобщение, което е изпратено през комуникационния канал, за който те са абонирани.

- *Излъчване (Broadcast)* – Съобщението бива доставено до всички сървиси. Този тип комуникация се използва при успешно влизане в системата на регистриран потребител. При настъпването на това събитие, всички микросървиси се уведомяват за това и към тях се изпращат Id на потребителя заедно с Id на неговия WebSocket.

Всеки от описаните модули и микросървиси има ясно изразени функционални граници, които се дефинират преди процеса на разработка. Съвкупността от микросървиси образуват единна бизнес логика, която отговаря на функционалните нуждите на системата. Промяната на един от микросървисите няма да доведе до неустойчивост на системата. Това е постигнато, като се предоставя обратна съвместимост (*backward compatibility*) между всички обновени микросървиси с непроменените такива.

3.1.1.4 Разработване

Процесът по разработване на микросървиси в този дисертационен труд започва с определяне на три основни аспекта: Ограничение на контекста, Типовете комуникации и Консистентност на данните.

- **Ограничение на контекста** (*Bounded Context - Boundaries*) – всеки микросървис трябва да има определени логически граници, за да се постигне гъвкава архитектура с липса на директни зависимости между сървисите. *Bounded Context* (Henry A., 2020) е ключов подход при разработването на дистрибутирани приложения. Чрез този подход, големи и сложни бизнес модели се разделят на по-малки, но логически смислени и групирани такива, които са наречени “Boundaries”. Всеки един сървис в разработваната система се занимава само с един “Bounded Context”, като те са дефинирани и изпълнени по начин, по който няма зависимост между тях. Промяната на всеки един от тях, не води до директна причина за промяна, в който и да е от другите сървиси.
- **Типовете комуникации в системата** - създаването на непрекъсваем и стабилен комуникационен канал за даден сървис в разработваната система е водещо при избора на неговия тип (Gadge S., 2017). Съществуват два типа комуникационни канала – синхронен и асинхронен.
  - *Синхронен*
    - Клиент – API Gateway – Сървис

Комуникациите се извършат чрез HTTPS заявка-отговор (request-response) (Фиг. 3.8). Започнатите задачи могат да бъдат завършени след отговор към клиента. Този тип

комуникация се прилага, когато потребителската заявка може да получи бързо необходимата информация за отговор, като се обръща само към един микросървис.

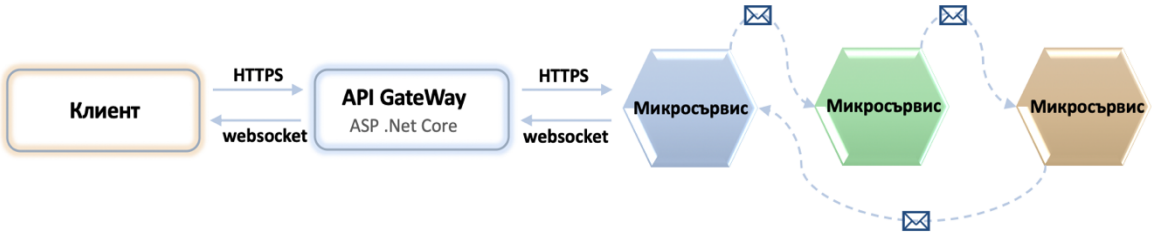
В разработваната система в този дисертационен труд се избягва употребата на синхронна комуникация.

Фигура 3.8: Синхронна комуникация Клиент – API Gateway – Сървис

- *Асинхронен*

- Сървис – клиент

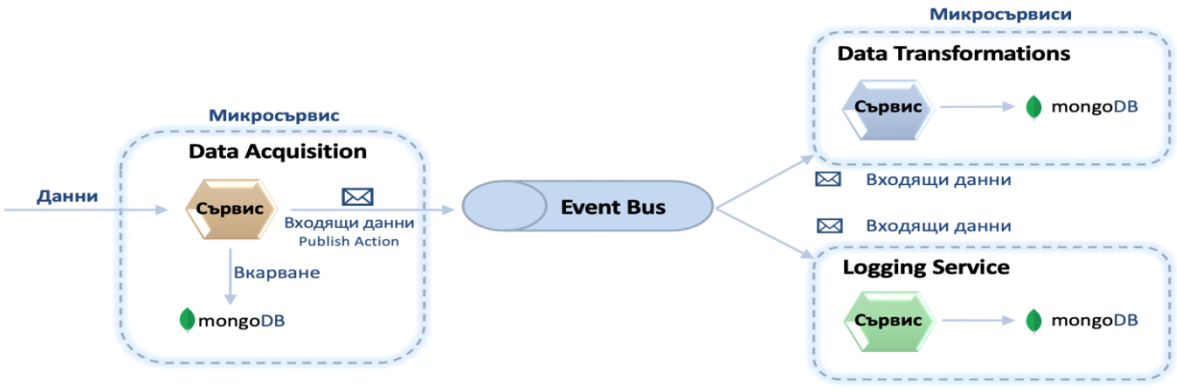
Комуникациите използват WebSocket (Miu A., 2020), който представлява разширение на HTTP протокол и се създава директна връзка между клиента и сървъра (фиг. 3.9). Чрез този канал става обновяването на информацията в клиента (браузъра) в реално време.



Фигура 3.9: Синхронна комуникация Сървис - Клиент

- Сървис – сървис

Комуникациите използват асинхронна размяна на съобщения като инициаторът на комуникацията не получава отговор. Системата извършва комуникации с един или много получатели в зависимост от конкретната нужда. Обърнато е специално внимание на устойчивост на връзката и продължителността (*durability*) на предаваните съобщения.



Фигура 3.10: Комуникация Сървис- Сървис

- **Консистентност на данните при работа с микросървиси**

o Заявка на данни от повече от един микросървис се среща често при работата на системата. За да се избегне прекалено честото общуване между сървисите (което е лоша практика), част от най-често исканите данни са дублирани в Cold Database (Ullah M., 2008) за бързо достъпване от страна на клиента. Този подход се използва само за статични данни, защото не е подходящ за работа с данни в реално време.

o Сегментация на отговорностите на заявките (CQRS - Command Query Responsibility Segregation) е архитектурен модел (Barnkob M., 2018), който разделя четенето и писането на две различни операции. Това означава, че всяка операция трябва да бъде команда, която изпълнява действие, или заявка, която връща данни. Командата не може да върне данни и заявката не може да промени данните. Този модел се използва в разработваната в този дисертационен труд система за генериране авансово на данни само за четене (read only). Този подход подобрява значително работата на системата, като се отчита, че има възможност за работа с неконсистентни данни за кратък период от време.

o Системата допуска извършване на анализ за заявките, които са направени към нея. Този анализ позволява установяване на случаи, когато има нужда от твърде често агрегиране на данни от различни микросървиси. Това е знак, че архитектурата на системата може да бъде оптимизирана.

3.1.1.5 Тестване

Разработваната в този дисертационен труд софтуерна система трябва да бъде сигурна, достъпна 24/7, скалируема и устойчива. За да се гарантират тези характеристики е необходимо да бъдат извършени редица тестове на различните части участващи в изграждането системата.

Микросървисите представляват най-критичната част от софтуерната система и като такива, е необходимо да бъдат надеждни и да работят без прекъсвания. Само автоматизираните тестове (Sotomayor J., 2019) дават това ниво на увереност, че всички грешки, а не само критичните, могат да бъдат намерени и отстранени още по време на процеса на разработване. Поведението на системата се тества по няколко различни начина, всеки от който играе своята важна роля за създаване на софтуерно приложение, което работи устойчиво и безаварийно в реална среда (инсталирано).

Видове приложени тестове:

• *Единични тестове (Unit tests*) се прилагат в системата, за да се удостовери, че всички индивидуални компоненти работят, както се очаква от тях. Всеки микросървис разполага със собствен набор от единични тестове, които не зависят от тези на останалите микросървиси.

• *Интеграционни тестове (Integration tests)* се прилагат в системата, за да се удостовери, че всички компоненти работят по очаквания начин с външни елементи като бази от данни, брокери на съобщения (Wittek K., 2019) и/или входно-изходни операции.

• *Функционални тестове (Functional tests)* се прилага за всеки микросървис, за да се даде нужната гаранция, че той работи според изискванията за своята функционалност.

• *Тестове за обслужване (Service tests)* се използват в системата за проследяване правилното изпълнение на различни случаи (use-cases), които са част от бизнес логиката на системата (Meinke K., 2015). По време на изпълнението на тези тестове, няколко микросървиси комуникират и работят едновременно, а крайният резултат се проверява за достоверност с очаквания такъв. Тези тестове най-много се доближават до работата на системата в реална среда.

3.1.1.6 Внедряване и Поддръжка

Софтуерното внедряване е процес на инсталиране на завършено софтуерно приложение. Този етап се извършва след извършено тестване, за да се гарантира, че всички грешки и отклонения са идентифицирани и отстранени. Софтуерната поддръжка е непрекъсната услуга, която гарантира поддържането на персонализираното софтуерно решение. Този етап включва коригиране на грешки, оптимизиране и актуализиране на версии.

В изградената в този дисертационен труд микросървисна архитектура е от важно значение поддържането на всеки микросървис. Те са различни по своята функционалност и значимост за системата. За да се постигне максимална гъвкавост при процеса на инсталиране на нови версии без да се компрометира непрекъсваемостта на работата на системата, се използват два подхода (Yang B., 2018) за обновяване на сървисите:

- *Blue-Green deployment* се използва в системата за инсталиране на обновени микросървиси, които са критично важни за работата на системата. По време на този процес в работната среда за определен момент съществува както обновения сървис,

така и неговата предишна версия. Както новата, така и старата версия са напълно функционални и работещи. За лесно разпознаване на текущата версия на всеки един микросървис в системата се използва версия, базирана на маршрутизация (Routing based versioning). В URL (Uniform Resource Locator) на всяка една заявка, която този сървис може да обработи, конфигурира и обнови неговата версия. В сегмента „/v1/users“ се указва, че текущата версия на сървиса е „1“. По време на процеса на обновяване на сървисите за определен момент се поддържат двете версии, за да може промените лесно да се върнат обратно (roll back) ако е налице такава нужда. Трафикът, който върви към тях първоначално е насочен в по-голямата си част към старата версия. С времето, когато има пълна увереност, че новата версия отговаря напълно на изискванията за функционалност и сигурност, трафикът се пренасочва изцяло към нея, а старата версия се премахва от системата.

Този начин на обновяване на сървисите в системата се прилага върху Identity Service, Machine Learning Service, Data Management Service и Logging Service.

- *Canary deployment* се използва от системата за обновяване в системата на микросървиси, които имат голямо натоварване върху себе си. При този подход (Tseitlin A., 2015), промените първоначално засягат малка част от функционалността на сървиса, след това по-голяма, докато стигне момент, когато целият микросървис е обновен напълно. Този начин на обновяване се прилага върху Data Acquisition Service и Data Transformation Service.

3.2.2 Потребителски интерфейс

Архитектурата на потребителските интерфейси (UI (User Interface)) обхваща планирането и проектирането на техническите, функционалните и визуалните компоненти на един уебсайт – преди той да бъде разработен и внедрен.

Интерфейсът е точка на взаимодействие между потребителя и системата. Той е от важно значение за визуализацията на събраните и анализирани сензорни данни от IoT устройства и начинът, по който те биват представени на потребителя. Затова е необходимо да бъде извършена цялостна организация на съдържанието (IA (Information Architecture)) в UI. Чрез IA се цели структуриране на информацията за създаване на интуитивно разпределение и позициониране на елементите, което подобрява потребителското изживяване (UX (User Experience)).

Разработката на UI архитектурата в този дисертационен труд се извършва по утвърдената „Agile“ методология (разгледана в 3.2) за създаване на софтуерна

архитектура, като се следват основните етапи на определяне на изискванията, анализ, проектиране, разработване, тестване, внедряване и поддръжка.

3.2.2.1 Определяне на изискванията

На база изискванията за разработката на потребителски интерфейс се определя избора на технологии, функционалности и начин на работа. Главните изисквания по време на разработката на потребителския интерфейс в този дисертационен труд са:

- Покриване на потребителски изисквания;
- Бързина при зареждане и обновяване;
- Дизайн за съвместимост с различни устройства и браузъри;
- Добре организирано и смислово подредено съдържание;
- Интуитивна и разбираема навигационна структура;
- Гъвкавост при добавяне на нови функционалности;
- Стабилност и непрекъсваемост на работа;
- Сигурност на потребителските данни;
- Лесна за поддръжка и обновяване;

Съвкупността от описаните изисквания към потребителския интерфейс са ключови за правилното определяне на технологичния стек, необходим за изпълнение на поставените изисквания към системата в този дисертационен труд.

3.2.2.2 Анализ

След определяне на изискванията към потребителският интерфейс в този дисертационен труд, следва да бъдат определени и функционалните изисквания, преди да бъде направен избор на технологичен стек, чрез който да бъде изградена UI частта.

Функционалните изисквания, които потребителския интерфейс предлага, дават възможност на потребителя на разработваната платформа да:

- Въвежда, редактира или изтрива лична информация;
- Въвежда, редактира или изтрива информация за пчелни кошери;
- Извършва мониторинг на своите кошери в реално време;
- Подава команди към IoT устройствата, които са под контрол на потребителя;
- Достъпва статистически анализи;
- Достъпва резултати за прогнозите извършени от ML моделите;
- Получава и реагира на известия;

Чрез анализ и описание на функционалните изисквания на потребителския интерфейс се определя технологичния стек, чрез който да бъдат изпълнени

необходимите функционалности. Използваният технологичен стек за изграждане на потребителският интерфейс се състои от:

- Платформа: Angular 8, Node.js;
- Програмни езици: Typescript, HTML5, CSS3;
- Библиотеки: Bootstrap4, Morris.js, LineChart.js.

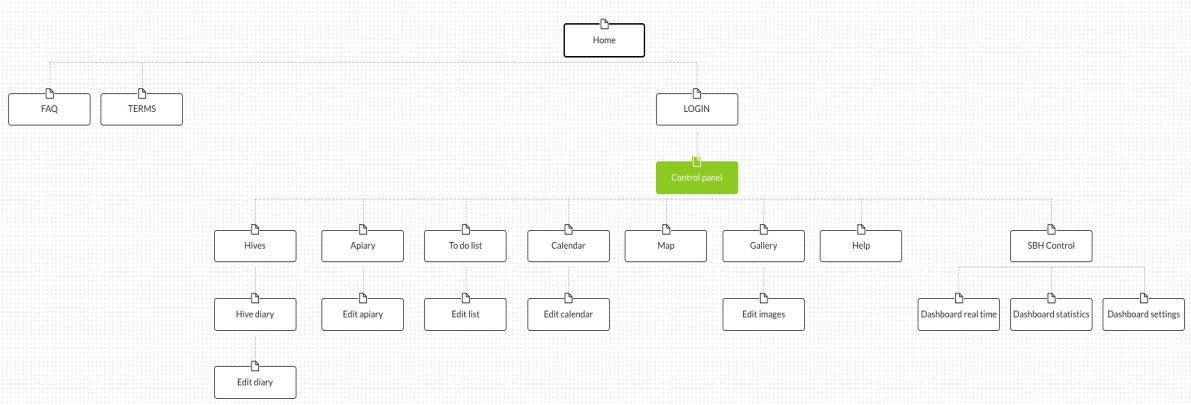
3.2.2.3 Архитектура и проектиране

Организационната схема на разработената в този дисертационен труд платформа, може да бъде определена като "субективна". Субективните схеми категоризират информацията по начин, специфичен за областта. Този тип категоризация подпомага потребителите да разберат и да начертаят връзки между отделните части. Субективните схеми на разработената платформата включват:

- Схеми по теми – организират съдържание въз основа на конкретния предмет.
- Схеми по задачите – организират съдържанието, като се отчитат нуждите, действията, въпросите и процесите, които интересуват потребителите.

Организационната схема определя общите характеристики на компонентите на съдържанието и влияе върху логическото групиране на тези компоненти. Освен изграждането на организационна схема е важно и изграждането на организационна структура (Bakanova N., 2018). Чрез организационната структура се определят видовете взаимоотношения между компонентите на съдържанието и групите.

Организационната структура на платформата е "Hierarchical structure – top-down" (фиг. 3.11). Това е добре позната и лесна за ориентиране интуитивна структура. Чрез нея се изгражда баланс в ширината и дълбочината на платформата спрямо спецификата ѝ. Този подход позволява добавяне на съдържание в различните части на платформата без да бъде необходимо извършване на преструктуриране, което е от съществено значение за постигане на поставените цели.



Фигура 3.11: Организационна структура на разработения потребителски интерфейс

Потребителският интерфейс в този дисертационен труд е тематично разделен на две части:

- Презентационна (Landing page)
- Контролен панел (Control panel)
    - Описателна част
    - Контролна част

Всички части поддържат различни функционалности, които са консистентни и се допълват взаимно. Презентационната част запознава потребителя с характеристиките, спецификите и функционалностите на системата. Има обособена част с отговори на най-често задаваните въпроси по реализацията и употребата на системата.

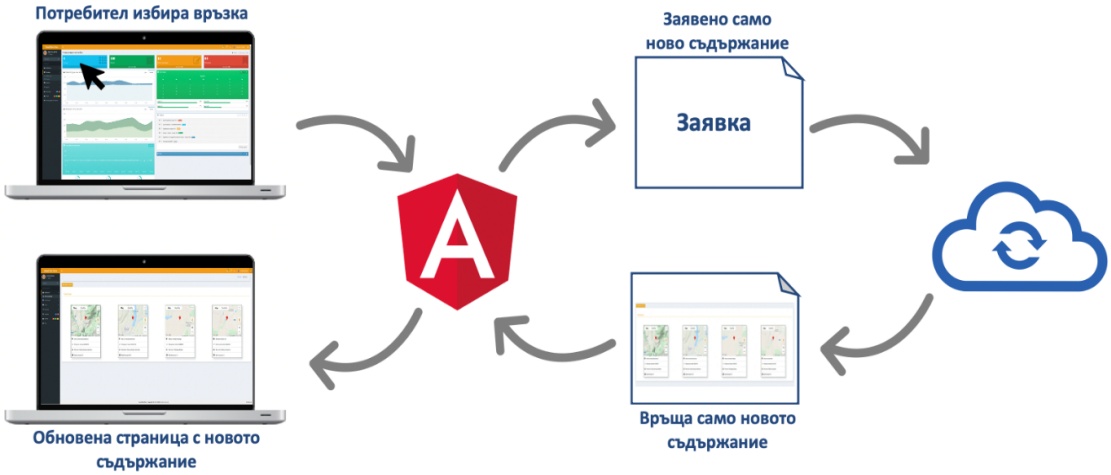
Контролният панел е разделен на две части – описателна и контролна. Описателната част позволява на потребителите да създават, въвеждат, редактират и изтриват информация за своите пчелини и кошери в тях. В тази част се предоставя възможност за създаване на пчелини, добавяне на кошери в тези пчелини, водене на електронен дневник за състоянието на всеки от въведените кошери, лист с минали, бъдещи и в процес на изпълнение задачи, попълване на календар с дейности и събития, задаване на GPS координати и галерия.

Изграденият електронен пчеларски дневник съдържа таблици, в които може да се въвеждат данни за смяна на пчелна майка, дата и вид на третиране на кошера, болести, честота на подхранване на кошера и количество добит мед. Чрез този дневник се цели да се улеснят потребителите при проследяване развитието на всеки кошер през годините чрез опции за филтриране и графично визуализиране.

Контролната част е достъпна за потребители, разположили IoT системата в своите пчелини. Те имат възможност да извършват real-time мониторинг на своите кошери и да виждат пълен набор от статистически данни визуализирани чрез диаграми. В контролната част се визуализират резултати получени след обработката на събираните данни, които се моделират, за да се извърши предсказване на събития за бъдещо здравословно състояние на пчелното семейство и бъдещо количество събран мед. Изграждат се шаблони (patterns). При засичане на отклонения от нормалните данни в пчелното семейство се изпраща в платформата съобщение на потребителя с информация за наличния проблем.

3.2.2.4 Разработка

Потребителският интерфейс в този дисертационен труд е разработен на платформата Angular, която е създадена специално за изграждане на клиентски SPA приложения (Japikse P., 2020). Наричат се SPA (Single-Page Application), защото цялото съдържание на приложението е разработено под формата на самостоятелни компоненти, които се зареждат динамично в една уебстраница, като създават впечатление, че приложението всъщност може да навигира между повече от една страница (фиг. 3.12).



Фигура 3.12: Процес на обновяване на уеб съдържание в разработеният потребителски интерфейс

SPA приложенията се характеризират с редуцирано количество динамични опреснявания на страниците при получаване на данни. Това е постигнато чрез използване на AJAX комуникация със сървъра. Опресняванията се извършват не на целия DOM (Document Object Model), както при нормалните уеб приложения, а само на необходимия раздел, който съдържа нови данни за визуализация. Благодарение на това се намалява режийното зареждане на приложението, което го прави леко и бързо за употреба, а наличието на преизползваеми компоненти намалява значително повторенията на софтуерен код.

Основни градивни елементи на Angular са модули, компоненти и маршрутизация.

- **Модули**

Angular приложението е от модулен тип. За да поддържа своята модулност е изграден AppModule, който е в основата на приложението (коренен модул). При нужда от разширяване лесно могат да бъдат добавени нови модули с допълнителни функционалности. Модулът е клас с @NgModule декоратор.

Декораторът се използва за прикачване на метаданните към класовете, за да се знае тяхната конфигурация и как те работят.

- **Компоненти**

Компоненти се използват за определяне и контролиране на желаните от потребителя изгледи (уеб съдържание).

Приложението има един основен компонент - AppComponent и множество допълнителни. Основния компонент се зарежда в модула и се импортират неговите зависимости (допълнителните компоненти). Към тях се добавя @NgModule декоратора съдържащ необходимите метаданни. Всеки компонент съдържа шаблон, услуги и зависимости и данни.

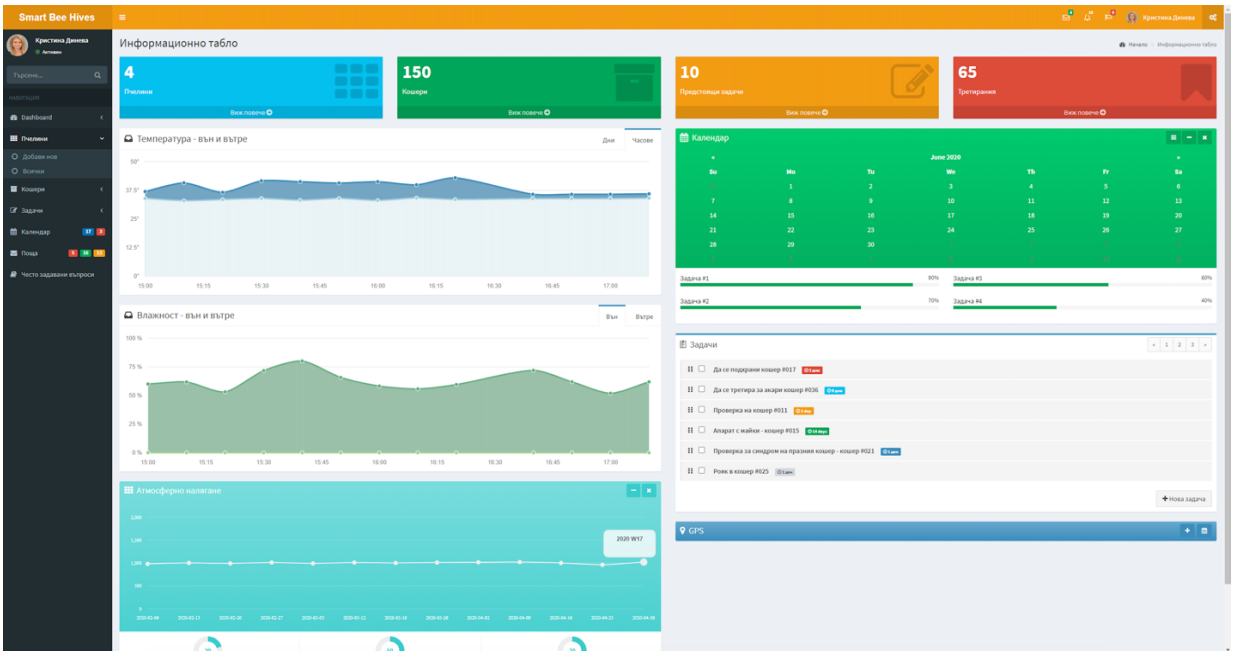
- Шаблон – HTML съдържание, което определя визуалната част на компонента. Логиката на шаблона се осигурява от директиви (променят поведението на съществуващ елемент) и свързаните с приложението данни към DOM дървото на страницата.
- Вкарване на зависимости към външни услуги (Dependency Injection) – разрешава се достъп до определени, външни за компонента услуги при наличие на абонамент за тях.
- Услуги (services) – представляват самостоятелен софтуерен код, който извършва някаква конкретна операция. Пример за такива операции представляват запис на данни в кеш, четене на данни от кеш, заявки към база от данни и редица други. Тези услуги могат да бъдат консумирани от различни компоненти на приложението.
- Свързване на данни (Data Binding) – в приложението се използват два вида свързвания:
  - Свързване на събития – използва се за актуализиране на данните в приложението при настъпване на събитие, което е извършено от потребителя.
  - Свързване на свойства – използва се за предаване на данни от компонента. Улеснява интерполирането на стойности, които се изчисляват от приложението и се връщат обратно в HTML.

- **Маршрутизация**

Чрез маршрутизация се определя навигационния път между различните елементи в приложението, като се преобразува URL пътят към съвкупност от компоненти, а не към страници. При извършено навигиране от потребителя, маршрутизатора прихваща поведението на браузъра и показва или скрива йерархиите на компонента. Това прави приложението стабилно по своя характер.

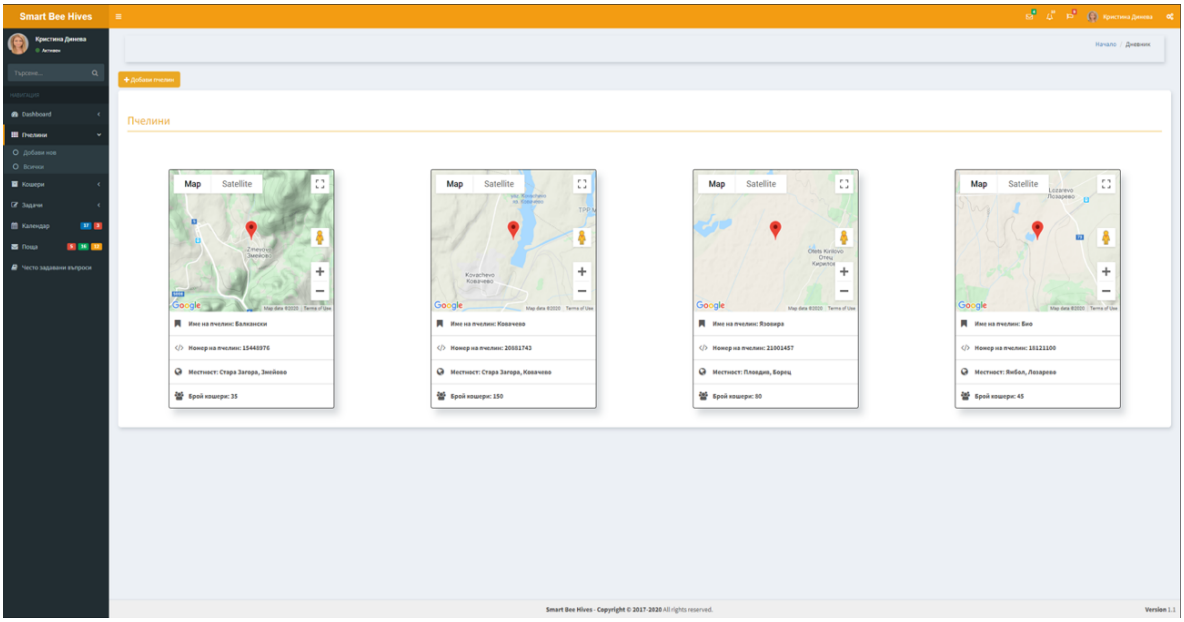
На фиг. 4.13 „Информационно табло“ е показано уеб съдържанието, което се достъпва в приложението чрез адрес за навигация “https://smartbeehives/dashboard“. То се формира чрез извикване и зареждане на компонентите, които са отговорни за конкретните секции:

- Хедър – хоризонтална информационна лента с елементи за нотификация;
- Вертикална навигационна лента – предоставя бърз достъп до всички навигационни маршрути;
- Главно съдържание
  - Помощна навигационна лента – предоставя информация и бърз достъп до най-често ползваните компоненти в приложението;
  - Контейнер за групиране на информационни елементи – съдържа множество информационни компоненти; Те са тематично подредени спрямо съдържанието, което визуализират.
- Футър – хоризонтална информационна лента очертаваща края на страницата;



Фигура 3.13: Потребителски интерфейс - информационно табло

При действие от страна на потребителя да достъпи ново съдържание се изпраща заявка към сървъра за обновяване на съответните секции в страницата. Сървъра връща към клиента само новото съдържание под формата на нови компоненти и обновява страницата с тях без да бъдат повторно зареждани съществуващите елементи (фиг. 3.14), които остават непроменени.



Фигура 3.14: Потребителски интерфейс – пчелини

Този принцип на работа предоставя бързина при зареждане и обновяване на информационното съдържание и лесно добавяне на нови функционалности – чрез създаване на нови компоненти и/или цели модули.

3.2.2.5 Тестване

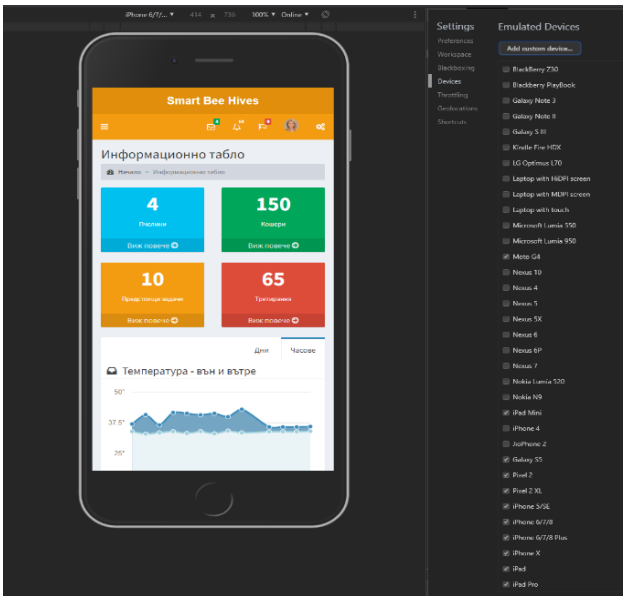
Софтуерното тестване е формален процес на проверка върху разработения потребителски интерфейс, с цел получаване на информация за качеството му. За тестване на изграденият интерфейс в този дисертационен труд се извършва функционално тестване, което проверява коректността на системата от гледна точна на потребителя. Обхвата на тестовете включва всички нива от изградената организационна структура.

Потребителските действия, които са важни за тестване, могат да бъдат групирани в категориите:

- Навигация;
- Попълване на входни полета;
- Работа с файлове;
- Работа с бисквитки (cookies);
- Съобщения;

Разработения интерфейс се тества за поддръжка от браузърите: Интернет експлорър 11.0.11, Гугъл хром 81.0.4044.129 , Файърфокс 75.0, Опера 68.0.3618.63 и Сафари 5.1.7. Главно изискване към потребителския интерфейс е той да бъде достъпен

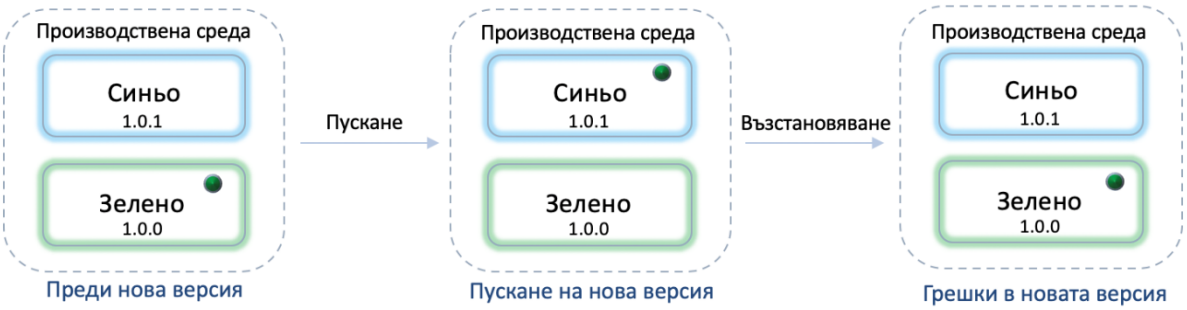
през всички приложения. За изпълнение на целта се извършва първично тестване на гъвкавостта на дизайна (responsive design) на предоставените симулатори от браузърите (фиг. 3.15). Тестването обхваща резолюциите от 320px до 2560px (4K).



Фигура 3.15: Тестване на интерфейсна визуализация в симулатор

3.2.2.6 Внедряване и поддръжка

Потребителският интерфейс е внедрен по модела „Синьо – зелено“. Това е модел, използван за намаляване на времето на престоя чрез използване на две идентични производствени среди, така че едната от тях да е активна в момента със стабилна производствена версия, а другата е бездействаща – тя служи като „временно място“ за внедряване на стабилна версия.



Фигура 3.16: Процес на внедряване на софтуерна версия

На фиг. 3.16 са представени различните фази, през които се преминава в процеса на работа на модела „Синьо – зелено“. Съществуват две производствени среди „Синя“ и „Зелена“.

При фаза „Преди нова версия“ в производствената среда е активна „Зелена“ версия 1.0.0, а синята е неактивна. След постигане на желаната стабилност на версия

1.0.1 се пристъпва към втора фаза „Пускане на нова версия“, където активна става „Синя“ производствена среда с новата версия, а старата спира да се поддържа. В случай на неуспех с новата версия (грешки, нестабилност) старата версия се запазва и се преминава към следваща фаза „Грешки в новата версия“, където се възстановява старата, но стабилна „Зелена“ производствена среда с версия 1.0.0 до отстраняване на грешките във версия 1.0.1.

3.3 Изводи

В тази глава е предложена иновативна система, изградена от IoT устройства, облачна сървърна част и потребителски уеб интерфейс, които си взаимодействат и обменят данни помежду си.

Разработената IoT хардуерна архитектура и предложеният и реализиран нов метод за комуникация между IoT устройства, основан на йерархична IP адресация, дават значителни предимства на системата:

1. Ефективна работа с различни по вид хардуерни компоненти и устройства, лесна заменяемост на вече вградените или добавяне на нови такива при нужда от промяна в мащаба или функционалностите на системата.
2. Поддържка на различни комуникационни канали от IoT устройствата, което позволява работа с различни хардуерни компоненти в зависимост от конкретните нужди на потребителя, спецификите на терена и количеството използвани устройства.
3. Голяма част от логиката изпълнявана от устройствата може да бъде контролирана, променяна и обновявана от разстояние чрез интегрирани софтуерни подходи.
4. Справяне с по-високата консумация на енергия в системата чрез използване на хардуерни устройства за интелигентно управление на захранването и вграден часовник за реално време.
5. Хардуерната система има само една точка на достъп, което я прави с високо ниво на сигурност.

Софтуерната архитектура е от MSA тип. Тя се състои от група микросървиси, които са реализирани чрез прилагане на нов подход за организация на услуги за интелигентна обработка и обмен на данни. Това дават следните предимства на системата:

1. Съсредоточеност върху функционалностите, а не върху технологиите. Микросървисите нямат ограничения при използването на програмни езици и бази от данни за тяхното създаване и успешно функциониране. Всеки микросървис е насочен да изпълнява определена функционалност. Това прави тази архитектура адаптивна за

използване в множество други процеси и в различни канали в зависимост от нуждите. Всеки микросървис е отговорен за конкретна услуга, което води до изграждане на интелигентна и многофункционална архитектура.

2. Подобрена производителност и скорост. Микросървисите могат да работят едновременно без да е необходимо да се изчакват.

3. Лесна поддръжка, гъвкавост и скалируемост на цялата система. Всеки микросървис може да се управлява независимо. При нужда и бъдещо развитие на системата може лесно да бъдат добавени нови независими едни от други микросървиси. При промяна на един микросървис или добавяне на нов не се нарушава функционирането на останалите участници в системата.

4. Автономност и мултифункционалност. Микросървисната архитектура предоставя независимост и автоматична работа на микросървисите.

Потребителският интерфейс е изграден като SPA приложение, което е построено от серия компоненти. Този вид архитектура има следните предимства:

1. Постига се висока бързина и гъвкавост чрез зареждане само на необходимото съдържание (компонент), а не на цяла страница. Непроменението съдържание се съхранява в кеша.

2. Постига се високо ниво на преизползваемост на код – създадени веднъж компоненти, могат да бъдат използвани на различни места в приложението.

3. Съществуващият интерфейс лесно може да бъде надграден и обогатен с допълнителни функционалности, като се създават нови компоненти, които ще бъдат извикани при необходимост и визуализирани в главния компонент.

Съдържанието от тази глава са отразени в публикациите:

[4] Dineva, К., Atanasova, T., Security in IoT Systems, SGEM 2019*,* Vol. 19, Issue 2.1, pp. 576-577, ISBN 978-619-7408-79-9, ISSN 1314-2704, DOI:10.5593/sgem2019/2.1 **SJR:0.209**

[9] Dineva, K., Atanasova, T., Computer System Using Internet of Things for Monitoring of Beehives, SGEM GeoConference, ISSN:1314-2704, DOI:10.5593/SGEM_GeoConference, 2017, Vol. 17, Issue 63, pp. 169-176. **SJR:0.209**

[10] Dineva, K., Atanasova, T., Model of Modular IoT-based Bee-Keeping System. ESM'2017, EUROSIS-ETI, 2017, ISBN:978-492859-00-6, 404-406. **Scopus**

[12] Dineva, K., Internet of Things in Help of Sustainable Agricultural Development, International Conference AUTOMATICS AND INFORMATICS'2017, Sofia, Bulgaria, 2017, ISSN:1313-1850, pp.309-312.

Глава 4. Експериментални резултати

В тази глава основна цел е прилагане и валидиране (Задача 4) на предложената във втора глава методология за обработка, моделиране и интеграция на хетерогенни данни получени от разпределени IoT устройства. Преминава се през всички необходими стъпки от етапите „Подготовка на данни“ и „Моделиране“ до получаване на валидни и ефективни обучени модели, които се интегрират в разработената в този дисертационен труд MSA софтуерна платформа.

4.1 Прилагане на разработената методология

Последователността на действията е важна част от изследователския процес, за да може експериментите в случай на необходимост да бъдат повторени и в бъдеще подобрени. За правилната организация на цялостния процес от зареждане на базата, през почистване, нормализиране и разделяне на данните, до прилагане, трениране и оценка на моделите, е следвана разработената методология.

След обработка на данните, от съществено значение е определянето на максимално подходящ алгоритъм за машинно обучение. Това е сложен процес, който зависи от размера, качеството и естеството на събраните данни.

Работният процес за постигане на целите на експеримента се извършва в облачна изчислителна среда - Microsoft Machine Learning Azure Studio (Abraham T., 2018), (Azure, 2020).

Целият работен процес за валидиране на методологията се изпълнява в последователност от девет основни стъпки:

1. Зареждане на базата с данни.
2. Избор на работни колони (Select Columns in Dataset).
3. Редактиране на метаданни. Избира се колоната с наблюдения, която се превръща в категориен тип (Edit Metadata).
4. Премахване на стойностите Null, NA и NAN от данните. Маркират се колоните, в които има липсващи стойности и се избира метод на почистване, който представлява начина, по който да бъдат обработени липсващите стойности.
5. Нормализация на данните. Избира се подходящ метод за нормализация на данните, чрез който те се разпределят в интервал от стойности. Целта на нормализацията на данни е те да станат лесно сравними и да бъде намалена корелацията между колоните (Normalize Data).

6. Разделяне на данните на две части - 70% за трениране и 30% за тестване. На този етап се извършва и стратификация на данните (Split Data).
7. Трениране на модел с избраните алгоритми (Train model).
8. Оценка на всеки модел (Score Model).
9. Визуализиране на резултатите след извършено трениране на модел (Evaluate Model).

4.2 Избор на алгоритъм за машинно обучение

4.2.1 Класификационен анализ

С цел получаване на валиден класификационен модел за машинно обучение се прилагат алгоритми за класификация върху хетерогенни данни от IoT устройства. Данните са придобити от център за данни за изследване на медоносни пчели "The Data Center for Honeybee Research". Моделите биват тренирани за предсказване на един от общо два класа – **0** – пчелното семейство няма достатъчно условия да оцелее или **1** – пчелното семейство има достатъчно условия да оцелее. Тренирането се извършва на база определени параметри – вътрешна и външна температура при пчелните кошери и процент влажност т.е. ще бъде извършена биномиална класификация (binomial classification).

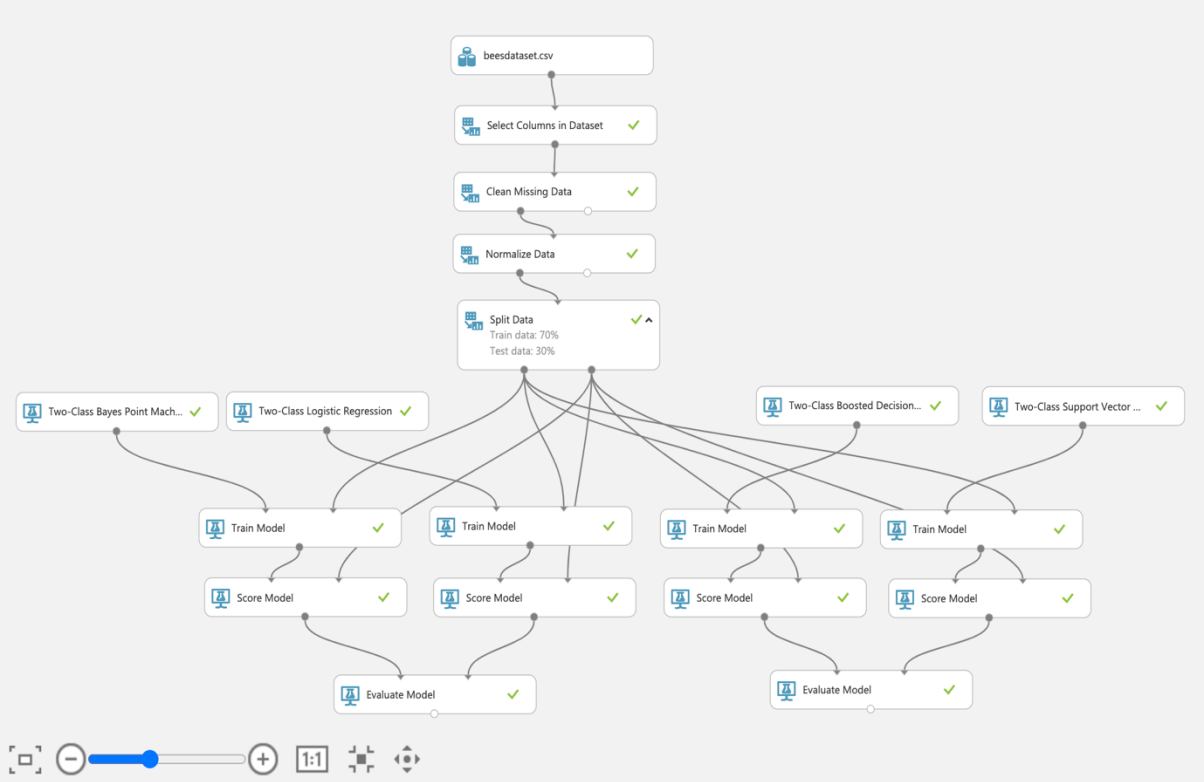
4.2.1.1 Трениране и тестване на модели

След зареждане на базата с данни, селектиране на необходимите колони, почистване и нормализиране на данните, те биват разделени на две части 70:30. 70% от данните се използват за трениране на моделите, а 30% се използват за тестване. При разделяне на данните се прилага стратификация.

Прилагаме четири вида алгоритми, избрани след извършен сравнителен анализ в Глава 2 (таблица 2.3), които са:

1. Two-class Boosted Decision Tree,
2. Two-class Bayes Point Machine,
3. Two-class Logistic Regression,
4. Two-class Support Vector Machine;

Схемата на работния процес е показана на фиг. 4.1.



Фигура 4.1 Схема на работен процес

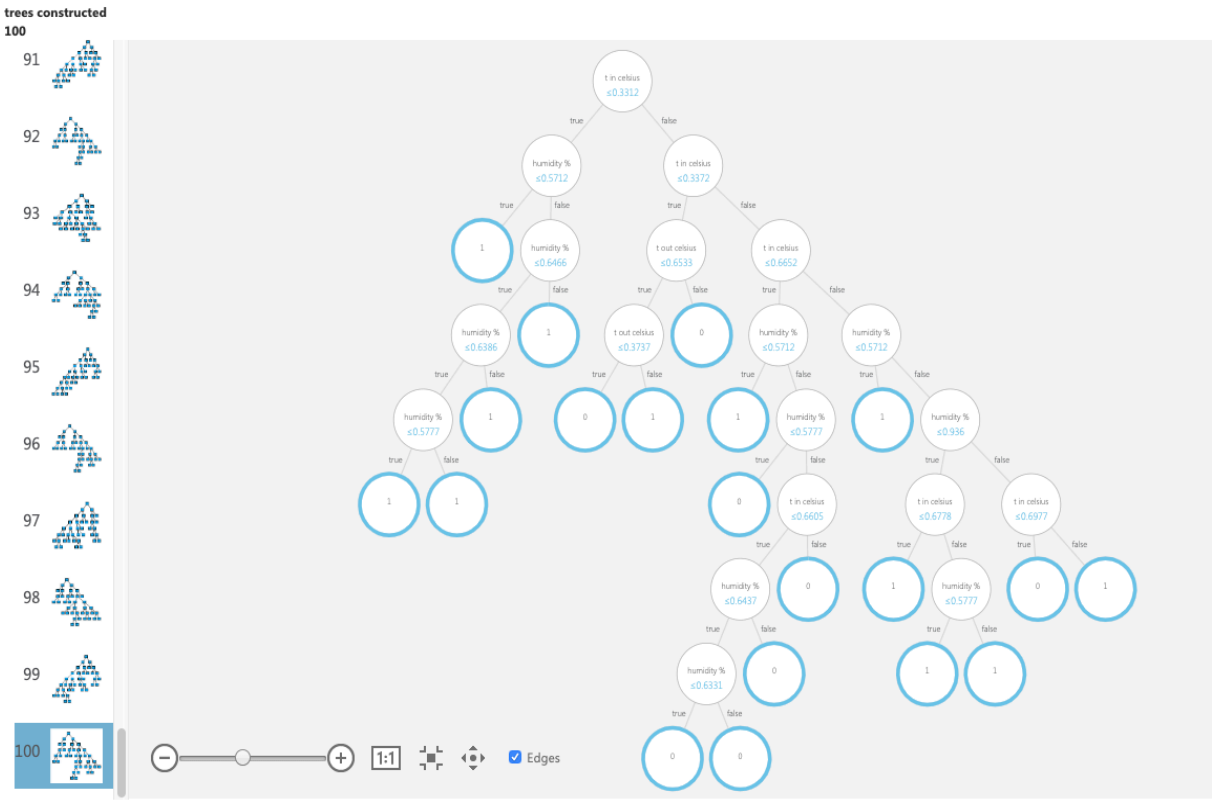
Всеки един алгоритъм е резултат от дадена моделираща функция, която определя начина, по който работи алгоритъма и от там и начина, по който се получават резултатите, затова е от важно значение правилната конфигурация (hyperparameter tuning) на всеки алгоритъм.

1. Two-class Boosted Decision Tree

Надграждащ се алгоритъм, при който се изграждат *n*-брой дървета, като второто дърво коригира грешките на първото дърво, третото дърво коригира грешките на първото и второто дърво и така до крайният брой итерации. Поради тази причина е от съществено значение правилната настройка на брой дървета, брой възли и стъпки за обучение. Прогнозите се базират на резултатите получени от цялата съвкупност от дървета.

Конфигуриране на алгоритъм

За правилната конфигурация на Boosted Decision Tree е важно да бъдат определени точно: брой итерации (брой построени дървета), брой крайни възли и стъпка на обучение на алгоритъма.



Фигура 4.2: Дърво на решенията

На фигура 4.2 е представено дърво от итерация номер 100 от модела Boosted Decision Tree. Общият брой изградени дървета за вземане на решения е 100. Броят на дърветата е обратно пропорционален на времето за обучение - създавайки повече дървета на решенията се постига по-добро покритие (всички дървета участват в определянето на прогнозата), но времето за обучение се увеличава.

Максималният брой крайни възли (листа) в този експеримент е 20, което представлява оптимален вариант при изграждане на този вид дървовидни структури. Задаването на твърде голям брой крайни възли крие риск за точността на модела и времето на обучение, а задаването на твърде малък брой крайни възли представлява риск за правилното построяване на структурата и получените крайни резултати.

За изграждане на дърветата е зададена стъпка на обучение 0.5, която определя размера на стъпката по време на обучение. Коефициентът на обучение определя колко бързо или бавно модела се приближава до най-доброто решение. Ако размерът на стъпката е твърде голям, може да се пренебрегне оптималното решение, а ако размерът на стъпката е твърде малък – ще бъде необходимо повече време за приближаване.

Всяко едно от изградените дървета се обучава последователно, като се научава от предходното и се фокусира върху неговите неправилни наблюдения (boosting метод). По този начин се създава нов модел с по-голямо тегло. По-успешните модели

получават по-големи тегла, които участват в редуцирането на пристрастията и образуването на краен резултат.

Резултати след извършено трениране на модел

На фигура 4.3 е представена част от таблица с резултати от извършеното обучение. Тя съдържа шест колони. Първите четири колони са променливите за обучението. Последните две колони са „Scored Labels“ и „Scored Probabilities“. Колоната „Scored Labels“ показва прогнозния клас. Колоната „Scored Probabilities“ показва вероятността едно пчелно семейство да принадлежи към прогнозния клас. Прогнозата в „Scored Labels“ се основава на колоната „Scored Probabilities“, където ако прогнозата за едно пчелно семейство да има добри условия за оцеляване е над 0,5 се прогнозира клас 1.

rows 7000 columns 6

view as

t in celsius	t out celsius	humidity %	observation	Scored Labels	Scored Probabilities
0.98375	0.331044	0.678311	0	0	0.00027
0.914	0.802934	0.11494	0	0	0.000337
0.33225	0.238279	0.685369	1	1	0.997997
0.96125	0.774191	0.847356	0	0	0.000361
0.662	0.362788	0.085066	0	0	0.00027
0.452	0.163254	0.668838	1	1	0.997997
0.5795	0.807202	0.703056	0	0	0.000403
0.67825	0.180527	0.396925	0	0	0.00027
0.01175	0.358453	0.661266	0	0	0.00039
0.26775	0.38146	0.720186	1	1	0.997997
0.20175	0.375992	0.250754	0	0	0.00032
0.02725	0.052551	0.002872	0	0	0.00032
0.1755	0.757853	0.339062	0	0	0.000337
0.1755	0.347182	0.090466	0	0	0.00032
0.862	0.458486	0.291686	0	0	0.000267
0.119	0.462154	0.511537	0	0	0.000317
0.32875	0.866689	0.909691	1	1	0.997756
0.08925	0.862354	0.188205	0	0	0.000337

Фигура 4.3: Оценка на вероятностите при Boosted Decision Tree

Резултатите след съпоставяне на колоните „Observation“ и “Scored Labels” са представени в матрица за оценка на класификацията (таблица 4.1), в която са представени резултатите след извършено обучение на Two-class Boosted Decision Tree, които показват, че общият брой на верните предположения е в размер на 6269 – за отрицателните и 722 за положителните, а за грешните предположения е 4 – за положителните и 5 за отрицателните.

Таблица 4.1. Матрица за оценка на класификацията при Boosted Decision Tree

	Predicted 1	Predicted 0
True 1	TP = 722	FN = 5
True 0	FP = 4	TN = 6269

Матрицата за оценка на класификацията служи за изчисление на всички необходими показатели за установяване точността, чувствителността и прецизността на модела. Тя показва видовете грешки допуснати от класификатора.

2. Two-class Bayes Point Machine

Алгоритъм с байесов подход към линейна класификация, който се характеризира с ефективното сближаване на оптималната байесова средна стойност на линейните класификатори чрез избиране на „Байесова точка“ като класификатор.

Конфигуриране на алгоритъм

Броят зададени итерации (повторения над данните за обучение) при конфигуриране на модела е 30. Този брой тренировъчни повторения е достатъчен, за да може алгоритъмът да направи точни прогнози с набори от данни при свързани променливи.

От друга страна включването на байес при трениране на модела подобрява неговата прецизност и намалява шансовете за получаване на неверни положителни резултати от модела.

Резултати след извършено трениране на модел

rows 7000 columns 6

t in celsius	t out celsius	humidity %	observation	Scored Labels	Scored Probabilities
0.98375	0.331044	0.678311	0	0	0.034878
0.914	0.802934	0.11494	0	0	0.000443
0.33225	0.238279	0.685369	1	0	0.153348
0.96125	0.774191	0.847356	0	0	0.105623
0.662	0.362788	0.085066	0	0	0.000795
0.452	0.163254	0.668838	1	0	0.11093
0.5795	0.807202	0.703056	0	0	0.112954
0.67825	0.180527	0.396925	0	0	0.011161
0.01175	0.358453	0.661266	0	0	0.243432
0.26775	0.38146	0.720186	1	0	0.203561
0.20175	0.375992	0.250754	0	0	0.016943
0.02725	0.052551	0.002872	0	0	0.003819
0.1755	0.757853	0.339062	0	0	0.036448
0.1755	0.347182	0.090466	0	0	0.005218
0.862	0.458486	0.291686	0	0	0.002692
0.119	0.462154	0.511537	0	0	0.105374
0.32875	0.866689	0.909691	1	0	0.381372
0.08925	0.862354	0.188205	0	0	0.017026

Фигура 4.4: Оценка на вероятностите при Bayes Point Machine

След трениране на алгоритъма Bayes Point Machine, получените резултати са представени в колона “Score Probabilities” и „Score Labels“ (фиг. 3.4), които съответно показват резултатите в интервал [0 , 1] и обобщеният резултат 0 или 1, който бива съпоставен с истинските данни представени в колона „Observation“. Броят на несъответствията между резултатите е представен в таблица 4.2.

Таблица 4.2. Матрица за оценка на класификацията при Bayes Point Machine

	Predicted 1	Predicted 0
True 1	TP = 3	FN = 724
True 0	FP = 89	TN = 6184

В матрицата за оценка на класификацията е представен броят на верните предположения в размер на 6184 – за отрицателните и 3 за положителните, а броят на грешните предположения е 89 – за положителните и 724 за отрицателните.

3. *Two-class Logistic Regression*

Алгоритъм, който в основната си форма използва логистична функция за моделиране на двоична зависима променлива. Прогнозира вероятността от възникване на събития чрез приспособяване на данни към функцията.

Конфигуриране на алгоритъм

За толеранс на оптимизацията е избрана стойността 0.0000001>= Double.Epsilon, която служи за праг при оптимизиране на модела. Ако подобрението между итерациите падне под този определен праг, се счита, че алгоритъмът се е сближил с решението и обучението спира.

Добавянето на регуларизация за превенция на модела от прекомерно нагаждане на резултатите е задължително при употребата на логистична регресия. С цел по-добро обучение на модела и постигане на максимално точни резултати е приложена комбинация от два типа регуларизация – L1 и L2 с тежест 1. (Elastic Net).

По този начин е изграден баланс между това да бъдат вкарани твърде много предположения в модела и това да се остави модела да извърши прекомерно нагаждане на резултатите (overfitting).

За възпроизводимост на същите резултати от трениране и тестване на модела е важно да бъде посочена “random seed” стойност в произволен размер, която да бъде

използвана в същия си размер и при следващи опити. Стойността на Random seed в този експеримент е зададена да бъде 123.

Резултати след извършено трениране на модел

Фигура 4.5 отразява резултатите получени след трениране на модела Logistic Regression. Пълните резултати са отразени в колона “Scored Probabilities”, а в колона “Scored Labels” се съдържа крайният класификатор 0 или 1, който може да бъде лесно сравним с колона „Observation”, в която се съхраняват реалните резултати. Верните и грешните предположения след извършено сравнение на резултатите от двете колони са систематизирани в матрица за оценка на класификацията.

rows 7000 columns 6

t in celsius	t out celsius	humidity %	observation	Scored Labels	Scored Probabilities
0.98375	0.331044	0.678311	0	0	0.041563
0.914	0.802934	0.11494	0	0	0.002567
0.33225	0.238279	0.685369	1	0	0.139729
0.96125	0.774191	0.847356	0	0	0.112837
0.662	0.362788	0.085066	0	0	0.003244
0.452	0.163254	0.668838	1	0	0.102865
0.5795	0.807202	0.703056	0	0	0.11123
0.67825	0.180527	0.396925	0	0	0.016296
0.01175	0.358453	0.661266	0	0	0.217814
0.26775	0.38146	0.720186	1	0	0.187933
0.20175	0.375992	0.250754	0	0	0.019884
0.02725	0.052551	0.002872	0	0	0.00682
0.1755	0.757853	0.339062	0	0	0.036607
0.1755	0.347182	0.090466	0	0	0.008767
0.862	0.458486	0.291686	0	0	0.006854
0.119	0.462154	0.511537	0	0	0.092269
0.32875	0.866689	0.909691	1	0	0.393483
0.08925	0.862354	0.188205	0	0	0.019907

Фигура 4.5: Оценка на вероятностите при Logistic Regression

В матрицата за оценка на класификацията (таблица 4.3) се визуализира производителността на модела чрез представяне на резултатите за всички наблюдения.

Таблица 4.3. Матрица за оценка на класификацията при Logistic Regression

	Predicted 1	Predicted 0
True 1	TP = 6	FN = 721
True 0	FP = 108	TN = 6165

Матрицата представя броят на верните предположения, които са в размер на 6165 – за отрицателните и 6 за положителните, а броят на грешните предположения е 108 – за положителните и 721 за отрицателните.

4. *Two-class Support Vector Machine*

Алгоритъм, решаващ класификационни и регресионни проблеми. В процеса на обучение се анализират входните данни и се идентифицират шаблони в многомерното пространство по категории.

Конфигуриране на алгоритъм

Зададеният брой повторения използвани при изграждането на модела е 3 >= 0. Този параметър се използва за контрол в съотношението скорост – точност.

Използваната ламбда стойност е 0.001 >= Double.Epsilon, която се използва като тегло за L1 регуларизация. Задаването на по-голяма стойност би довело до намаляване на точността на модела.

Извършената вътрешна нормализация на променливите е от тип MinMax, при която точките с данни се центрират на 0 и се мащабират, за да имат една единица стандартно отклонение.

Резултати след извършено трениране на модел

Резултатите след извършеното трениране на модела Support Vector Machine са представени в табличен вид (фиг. 4.6). В таблицата са представени променливите, с които е обучен модела и изходните резултати след обучението, които са представени в колона “Scored Labels”.

rows 7000 columns 6

t in celsius	t out celsius	humidity %	observation	Scored Labels	Scored Probabilities
0.98375	0.331044	0.678311	0	0	0.050739
0.914	0.802934	0.11494	0	0	0.003554
0.33225	0.238279	0.685369	1	0	0.120241
0.96125	0.774191	0.847356	0	0	0.157465
0.662	0.362788	0.085066	0	0	0.003293
0.452	0.163254	0.668838	1	0	0.090626
0.5795	0.807202	0.703056	0	0	0.131111
0.67825	0.180527	0.396925	0	0	0.015819
0.01175	0.358453	0.661266	0	0	0.171622
0.26775	0.38146	0.720186	1	0	0.166474
0.20175	0.375992	0.250754	0	0	0.016127
0.02725	0.052551	0.002872	0	0	0.004358
0.1755	0.757853	0.339062	0	0	0.034251
0.1755	0.347182	0.090466	0	0	0.006851
0.862	0.458486	0.291686	0	0	0.008129
0.119	0.462154	0.511537	0	0	0.07636
0.32875	0.866689	0.909691	1	0	0.415339
0.08925	0.862354	0.188205	0	0	0.018367

Фигура 4.6: Оценка на вероятностите при Support Vector Machine

Получените резултати след съпоставяне на реалните наблюдения с предсказаните стойности са представени в матрица за оценка на класификацията (Таблица 4.4).

Таблица 4.4. Матрица за оценка на класификацията при Support Vector Machine

	Predicted 1	Predicted 0
True 1	TP = 12	FN = 715
True 0	FP = 70	TN = 6203

Матрицата представя броят на верните предположения е размер на 6203 – за отрицателните и 12 за положителните, а броят на грешните предположения е 70 – за положителните и 715 за отрицателните.

4.2.1.2 Оценка на моделите

За определяне успеваемостта на един модел е необходима употребата на различни метрики за оценяването му. Няма универсална формула за това – всичко зависи от конкретните данни и целите свързани с тях.

Избраните метрики за този експеримент са: точност (accuracy), прецизност (precision), чувствителност (recall) и хармонична средна стойност (F1 Score).

Изчислението на метриките за оценка на моделите е въз основа на изградените матрици за оценка на класификацията (таблица 4.1, 4.2, 4.3 и 4.4), където са представени стойностите на всички верни и грешни, положителни и отрицателни резултати. След изчисляване на стойностите от матриците, получаваме коефициенти за всяка метрика на всеки отделен модел (таблица 4.5).

Таблица 4.5: Оценка на моделите

	Boosted Decision Tree	**Bayes Point Machine**	**Logistic Regression**	**SVM**
Точност	0.998	0.884	0.882	0.888
Прецизност	0.994	0.033	0.053	0.146
Чувствителност	0.993	0.004	0.008	0.017
Хармонична средна стойност	0.994	0.007	0.014	0.030

Сравнителната таблица съдържа изчислените коефициенти за всеки един от приложените модели. Boosted Decision Tree е с най-висока точност – 0.998, а Logistic regression с най-ниска точност в размер на 0.882. Хармоничната средна стойност между прецизност и чувствителност при Bayes Point Machine, Logistic Regression и SVM е под 0.5, а при Boosted Decision Tree, коефициентите са съответно 0.999 и 0.988, което означава, че от всички модели само Boosted Decision Tree описва най-добре данните.

След извършеното сравнение на получените метрики за оценка на ефективността на моделите, моделът Two-Class Boosted Decision Tree се откроява, като най-добър класификатор спрямо останалите.

Високият резултат успеваемост на този модел, съчетан с висок резултат на чувствителност и прецизност, налага проверка на достоверността на получените резултати чрез прилагане на подходи за преодоляване наличието на недостоверни резултати, в резултат на прекомерно нагаждане (overfitting).

Подходи за преодоляване на прекомерното нагаждане (overfitting) на модела

Прекомерното нагаждане на модел към данни е моделираща грешка, която възниква, когато дадена функция приляга твърде плътно към набора от данни. Обикновено това се случва, когато е създаден твърде сложен модел. Моделът научава детайлите и шума в тренировъчните данни, чрез което се постига висок процент точност върху тестовите данни. При добавяне на нови наблюдения към модела, той не може да приложи правилно усвоените концепции върху тях и това влияе негативно върху способността на модела да обобщава.

За да се избегне извършването на прекомерно нагаждане на модела се прилагат три основни подхода:

- Балансиране на данните;
- Избор на променливи;
- Кръстосано валидиране;

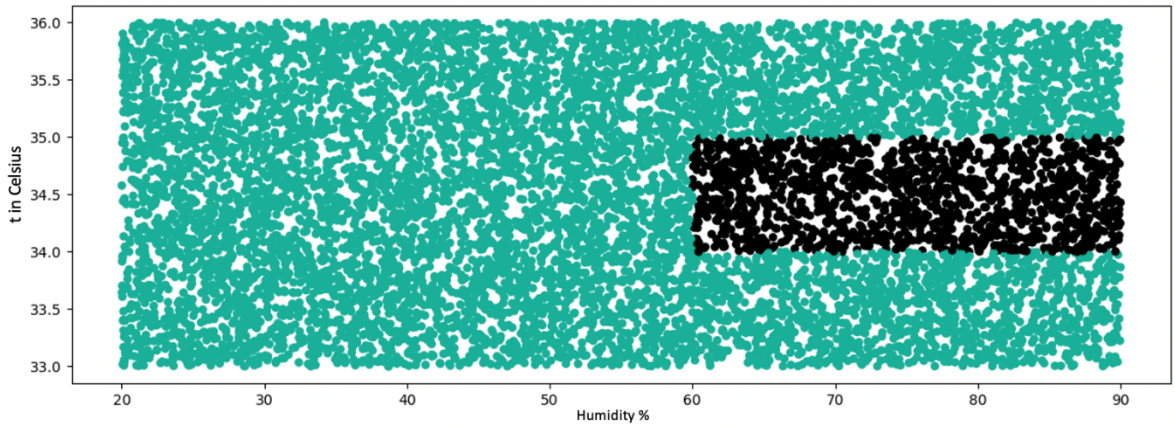
Балансиране на данните

Избраният модел е Two-Class Boosted Decision Tree. С него трябва да бъде съобразен изборът на техника за балансиране на данните. Този модел е чувствителен към пропорциите на различните класове и има склонност за предпочитание към класа с най-голям дял наблюдения (наричани още мнозинство). Това често води до подвеждаща точност на модела.

Проблематично в този тип случаи е когато се интересуваме от правилната класификация на редкия клас (известен още като малцинствен). Този тип модели имат за цел да минимизират общия процент на грешки, а не да обръщат специално внимание на малцинствения клас. За да могат да направят точна прогноза, моделите трябва да разполагат с достатъчна информация за този клас. Затова е необходимо да бъде извършено балансиране на данните. То може да бъде извършено чрез намаляване на клас мнозинство или увеличаване на примерите от клас малцинство.

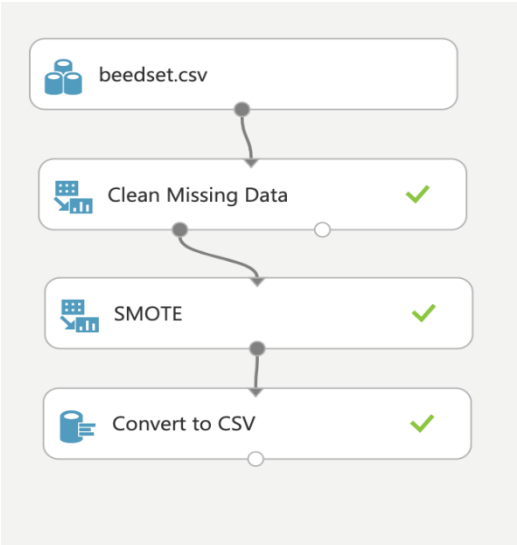
Поради спецификата и трудното добиване на данни за това изследване, намаляването на данни от клас мнозинство не е приемлив вариант. Този вид балансиране може да доведе до прекомерно нагаждане на модела поради малък брой данни с които се извършва изследването. Затова е избрана техника за синтезиране на нови случаи на малцинство – SMOTE (Synthetic Minority Oversampling Technique). Този подход увеличава възможностите пред всеки клас и прави извадките по-общи. SMOTE приема целия набор от данни като вход, като увеличава процента само на малцинствените случаи без да променя броя на мажоритарните.

Входящите данни са силно небалансирани в процентно съотношение на клас малцинство към клас мнозинство - 10,39 към 89,61 (фиг. 4.7).



Фигура 4.7: Разпределение на суровите данни преди прилагане на SMOTE модел

Работния процес (фиг. 4.8) за балансиране на данните започва с почистване на невалидни или липсващи стойности, след което се пристъпва към прилагане на алгоритъма SMOTE.



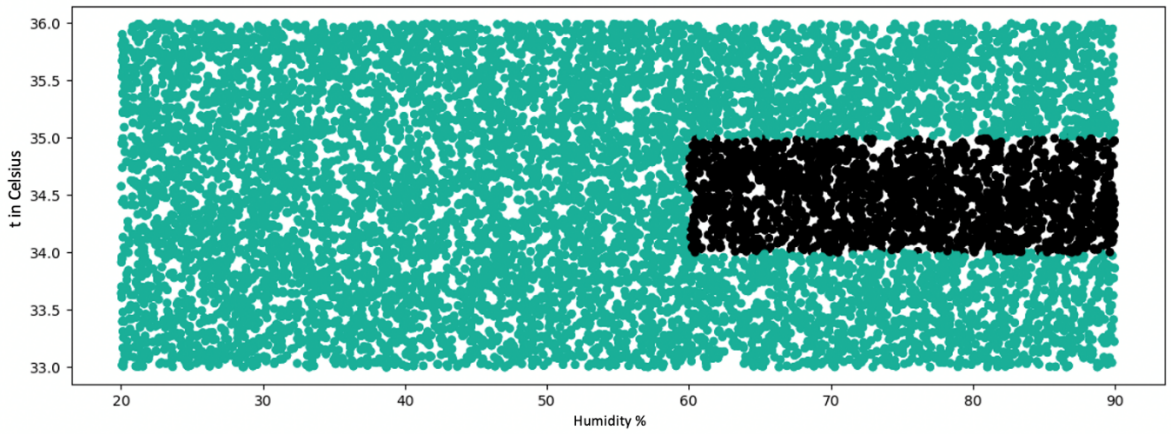
Фигура 4.8: Работен процес за прилагане на SMOTE

По време на работа на SMOTE се изпълнява последователност от пет стъпки:

1. Идентифициране на характеристичните вектори и техните най-близки съседи.
2. Взимане на разликата между двата вектора.
3. Умножение на разликата със случайно число между 0 и 1.
4. Определяне на нова точка в сегмента на линията, като се добави произволното число към характеристичния вектор.
5. Повтаря се процеса за идентифицираните вектори на елементите.

Идентифицирането на характеристичните вектори и техните най-близки съседи е важно правило за успешното функциониране на модела. Определянето на размера на обхвата им служи за правилното определяне на размера на пространството с признаци, които алгоритъмът използва, когато изгражда нови случаи. Разстоянието между всеки два случая се измерва чрез комбиниране на претеглени вектори на всички характеристики.

След изпълнение на пет стъпковата последователност се образува набор от данни, който съдържа оригиналните проби плюс допълнителен брой синтетични проби от малцинството, които са създадени, така че да бъде запазено разпределението на малцинствения клас (фиг. 4.9). Повишен е броят да данните без да бъдат получени нови данни, които да влияят на извличането на информация от малцинствения клас.

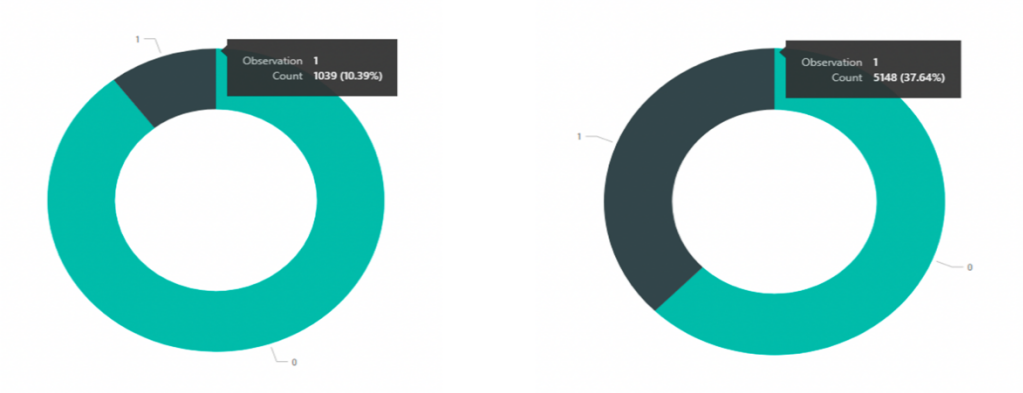


Фигура 4.9: Разпределение на данните след прилагане на SMOTE модел

След прилагането на алгоритъма се наблюдава увеличение на малцинствения клас от 10,39% на 37,64%, което се равнява на 27,25% повече генерирани копия на съществуващите данни. Това от една страна балансира данните, а от друга повишава размера им, което задължава модела да ги генерализира, за да постигне напредък. Това

води до повишена точност на модела, като същевременно намалява шанса за прекомерно нагаждане.

От фиг. 4.10 може да бъде видяна разликата в процентното съотношение на данните преди прилагане на SMOTE модела (в ляво) и след неговата употреба (в дясно).



Фигура 4.10: Процентно съотношение на данните преди (в ляво) след прилагане (в дясно) на SMOTE

В този дисертационен труд чрез прилагане на алгоритъма SMOTE върху небалансираните данни се постигна:

- Генериране на нови екземпляри от съществуващи случаи на малцинства, без да бъде променен броят на мажоритарните случаи.
- Получен балансиран тренировъчен набор от данни, който се използва за обучение на класификатора.

Избор на променливи

Регуларизацията е метод за добавяне на някои допълнителни ограничения към условие, за да предотврати прекомерното нагаждане.

В машинното обучение и статистиката изборът на променливи е процес на избор на подмножество от подходящи, полезни променливи, които да бъдат използвани при изграждането на аналитичен модел. Като вход се използва набор от данни, върху който се прилагат статистически методи по колоните. Това помага да се стесни полето на данните до най-ценните такива. Стесняването на полето на данните помага за намаляване на шума и подобряване на ефективност при обучението.

Изборът на правилни променливи подобрява модела и предотвратява проблеми, като:

- Прекомерно нагаждане на данните.

- Данни, съдържащи излишни или неуместни променливи, които не предоставят повече информация от избраните в момента променливи.
- Данни съдържащи неподходящи променливи, които не предоставят полезна информация в никакъв контекст. Включването на неподходящи променливи не само увеличава времето, необходимо за обучение на данните, но също така може да доведе до лоши резултати.
- Наличие на дублирана информация в обучителните данни води мултиколинеарност. При мултиколинеарността наличието на две силно корелирани променливи причинява неточни изчисления на други променливи.

Поради тези причини за правилния избор на метод, определящ полезността на променливи се прилагат следните алгоритми: Корелация на Пиърсън (Pearson Correlation), Корелация на Кендал (Kendall Correlation) и Чи-квадратична статистика (Chi Squared).

Таблица 4.6: Получени метрики след прилагане на методи за избор на променливи

	Температура - вътре	**Влажност**	**Температура - вън**	**Точност**
Пиърсън	0,013381	0,569841	0,017359	0,792
Кендал	0,009175	0,458461	0,009175	0,800
Чи-квадратична статистика	**0,156944**	**0,12615**	**0,000597**	**0,863**

След прилагане на трита вида алгоритми за избор на променливи върху данните в комбинация с избрания алгоритъм *Boosted Decision Tree* за класификация на данните, може да бъде видяно от таблица 4.6, че прилагането на алгоритъма Чи-квадратична статистика извършва най-добрия избор на променливи спрямо останалите. При използването на Чи-квадратична статистика се получава 86% точност спрямо 79% точност за Пиърсън и 80% за Кендъл.

Кръстосано валидиране

Кръстосаното валидиране е важна техника в машинното обучение за оценка както на променливостта на набора от данни, така и на надеждността на всеки модел, обучен с тези данни.

Като вход приема набор от данни, заедно с необучен модел на класификация – Boosted Decision Tree. Кръстосаното валидиране разделя набора от данни на 10 подмножества (сгъвки) с по 1367 или 1368 елемента и изгражда модел на всяка сгъвка. Генерират се общо 10 модела по време на кръстосаната проверка, като всеки модел е

обучен използвайки 8/10 от данните и е тестван върху 2/10 от тях. Когато процесът на изграждане и оценка е завършен за всички части се генерира набор от показатели за ефективност и отбелязва резултати за всяко сгъване (фиг. 4.11).

Fold Number	Number of examples in fold	Model	Accuracy	Precision	Recall	F-Score	AUC
0	1368	FastTree (Boosted Trees) Classification	0.841374	0.696148	0.99187	0.818106	0.890062
1	1367	FastTree (Boosted Trees) Classification	0.872714	0.761506	0.994536	0.862559	0.908885
2	1367	FastTree (Boosted Trees) Classification	0.858083	0.73617	0.98482	0.842532	0.899925
3	1367	FastTree (Boosted Trees) Classification	0.862473	0.739943	0.98659	0.845649	0.89535
4	1368	FastTree (Boosted Trees) Classification	0.870614	0.747384	0.984252	0.849618	0.918324
5	1368	FastTree (Boosted Trees) Classification	0.859649	0.723373	0.989879	0.835897	0.897282
6	1368	FastTree (Boosted Trees) Classification	0.865497	0.736152	0.994094	0.845896	0.906551
7	1368	FastTree (Boosted Trees) Classification	0.866959	0.744428	0.980431	0.846284	0.903235
8	1368	FastTree (Boosted Trees) Classification	0.869152	0.748571	0.994307	0.854116	0.898259
9	1368	FastTree (Boosted Trees) Classification	0.865497	0.739003	0.988235	0.845638	0.90508
Mean	13677	FastTree (Boosted Trees) Classification	0.863201	0.737268	0.988901	0.844629	0.902295
Standard Deviation	13677	FastTree (Boosted Trees) Classification	0.008952	0.017529	0.004871	0.011676	0.007972

Фигура 4.11: Набор от данни разделен на 10 подмножества при Кръстосано валидиране

На база получените статистически резултати за всяко едно подмножество, кръстосаното валидиране връща прогнозирани резултати в колона „Scored Probabilities“ и колона „Scored Labels“, които могат да бъдат съпоставени с реалните наблюдения в колона „Observation“ (фиг. 4.12), така че да можете да се оцени надеждността на прогнозите.

rows 13677 columns 5

view as

Fold Assignments	observation	Scored Labels	Scored Probabilities
2	0	0	0.000046
2	0	0	0.000042
1	0	0	0.000032
1	0	0	0.000032
2	1	0	0.482458
3	1	1	0.776413
6	0	1	0.750183
3	0	0	0.000037
1	0	1	0.702266
2	0	0	0.000046
7	0	0	0.000047
0	0	0	0.000034
0	0	0	0.000039
6	0	0	0.000041
1	0	1	0.63494
9	0	0	0.000038
7	0	0	0.000047
4	0	1	0.672253
1	0	0	0.000038

Фигура 4.12. Прогнозни резултати след прилагане на кръстосано валидиране

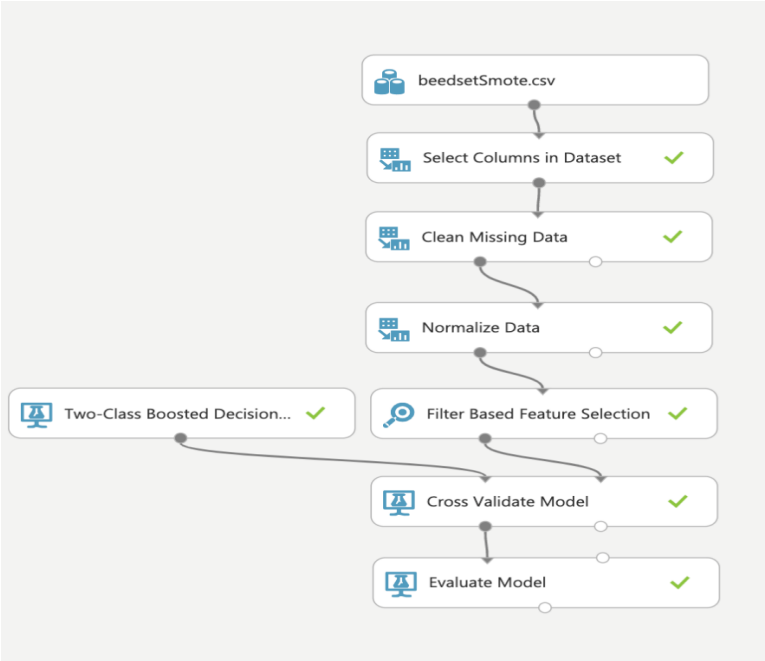
Сравнявайки статистическите данни за точност на всички сгъвания не се забелязват отклонения на някоя от десетте части. Средната точност е 0,86320, а стандартното отклонение на точността е в размер на 0,008952, което е в рамките на допустимите граници. Тези метрики показват, че моделът е стабилен и няма прекомерно нагаждане на данните.

Обучение и тестване на избраният модел

Важно е повторното обучение и тестване на избрания модел Boosted Decision Tree с цел проверка на ефективността на приложените подходи за справяне с прекомерното нагаждане на модела към данните. Отново се следва постъпков процес на работа, който се различава от този, използван по време на избирането на модела.

Целият работен процес (фиг. 4.13) се състои от осем основни последователни стъпки:

1. Зареждане на базата с балансирани данни.
2. Избор на работни колони.
3. Почистване на данните.
4. Нормализация на данните.
5. Избор на променливи.
6. Конфигуриране и прилагане на избрания алгоритъм – Boosted Decision Tree.
7. Кръстосано валидиране.
8. Оценка на модела.



Фигура 4.13 Процес на работа с кръстосано валидиране

На първа стъпка от работния процес се зареждат вече балансираните данни получени след прилагане на алгоритъма SMOTE към суровите данни. Новата база съдържа 27,25% (3677 примера/елемента) новогенерирани данни в малцинствения клас, което води до необходимост от генерализирането им от модела.

След селектиране на работните колони, данните в тях биват почистени и нормализирани. Чрез прилагане на алгоритъма Чи-квадратична статистика се определят тежести на променливите според степента на тяхното влияние върху крайния резултат.

Моделът Boosted Decision Tree се обучава и тества с приведените за това вид данни, като се използва кръстосано валидиране.

Получените метрики от кръстосаното валидиране показват успеваемостта на модела да генерализира данните.

Измерване на ефективността на обучения модел

За измерване ефективността на модела след трениране и тестване чрез кръстосано валидиране се генерира колона “Score bin”, която е със стойности в интервала [0;1]. В този интервал се разпределят предвидените от модела позитивни и негативни примери получени в колона „Scored Probabilities“. Благодарение на детайлното разпределение могат да бъдат открити както слабите точки в модела, така и местата с най-висока точност, прецизност и чувствителност.

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall
(0.900,1.000]	128	21	0.010	0.599	0.042	0.859	0.022
(0.800,0.900]	2196	584	0.203	0.711	0.527	0.793	0.395
(0.700,0.800]	2543	727	0.430	0.837	0.806	0.785	0.827
(0.600,0.700]	669	289	0.497	0.863	0.849	0.774	0.941
(0.500,0.600]	235	148	0.523	0.869	0.860	0.765	0.981
(0.400,0.500]	89	83	0.535	0.870	0.862	0.760	0.996
(0.300,0.400]	15	20	0.538	0.869	0.862	0.758	0.998
(0.200,0.300]	3	5	0.538	0.869	0.862	0.758	0.999
(0.100,0.200]	6	34	0.541	0.867	0.860	0.755	1.000
(0.000,0.100]	0	6618	1.000	0.408	0.580	0.408	1.000

Фигура 4.14: Интервално разпределение на получените метрики

Обобщено разпределение на получените резултати за истинските и фалшивите позитивни и негативни такива са представени в „матрицата на объркването“ (таблица 4.7), която отразява точността и грешките на модела. Матрицата описва работата на

класификационния модел върху набор от тестови данни, за които са известни истинските стойности.

Таблица 4.7. Матрица на объркване на модела

	Predicted 1	Predicted 0
True 1	TP = 5091	FN = 57
True 0	FP = 1814	TN = 6715

Броят на верните предположения е размер на 6715 – за отрицателните и 5091 за положителните, а броят на грешните предположения е 1814 – за положителните и 57 за отрицателните.

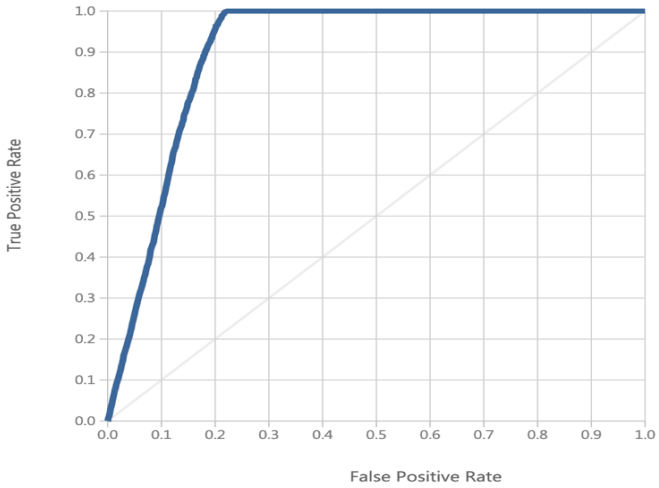
На база получените резултати разпределени в матрицата се изчисляват важни параметри като точност, прецизност, чувствителност и хармонична средна стойност (подробно описани в глава 2). Таблица 4.8 отразява обобщените резултати на тези параметри.

Таблица 4.8: Получени метрики след обучение и тестване на Boosted Decision Tree

	Boosted Decision Tree
Точност	0.863
Прецизност	0.737
Чувствителност	0.989
Хармонична стойност (F1 Score)	0.845

Визуално разпределението на резултатите от „матрицата на объркването“ се визуализират в три типа криви:

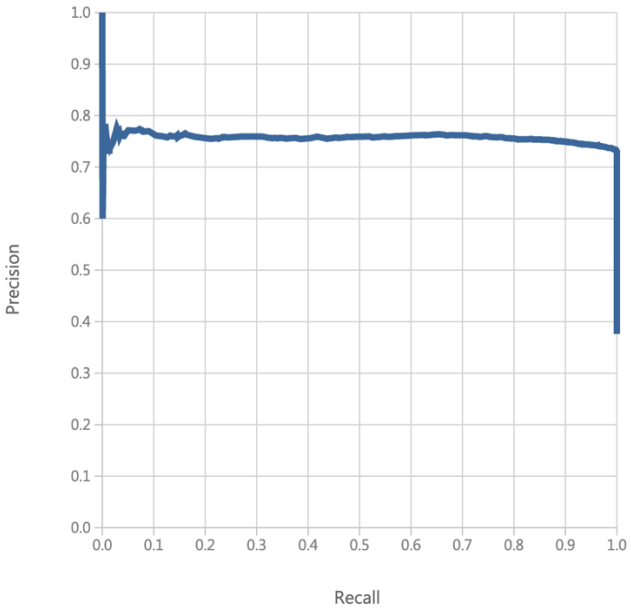
ROC curve (Receiver Operating Characteristic Curve) – графика, показваща ефективността на модела за класификация. Тази крива очертава два параметъра: истински положителни резултати и фалшиви положителни резултати. Тя обобщава производителността на класификатора над всички възможни прагове. Той се генерира чрез изчертаване на ос-у (истински положителен резултати) спрямо ос-х (фалшив положителни резултати), докато се променя прага за присвояване на наблюденията на даден клас (фиг. 4.15). Идеалната крива е максимално близо до горния ляв ъгъл.



Фигура 4.15: ROC curve

AUC (accuracy) – Осигурява обобщена мярка за сигурност във всички възможни прагове за класификация. Варира в стойност от 0 до 1. Генерира се чрез изчертаване по оста *y* на показателите за прецизност спрямо ос x с показатели за чувствителност. Модел, чиито прогнози са 100% грешни, има AUC 0,0, а този, чийто прогнози са 100% верни, има AUC 1,0. Това може да бъде изчислено като получения коефициент за прецизност бъде разделен на коефициента за чувствителност.

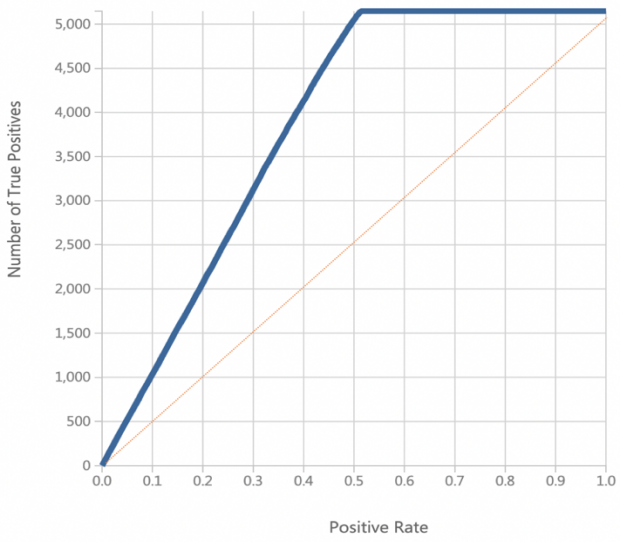
Получените метрики след обучение и тестване на Boosted Decision Tree показват, че прецизността на модела е в размер на 0.737, а чувствителността е в размер на 0.989. След тяхното изчисление получения коефициент е 0.902, което показва, че избраният алгоритъм е балансиран и има висок процент на успеваемост (фиг. 4.16).



Фигура 4.16: AUC

Lift curve – мярка за ефективността на класификационен модел, изчислен като съотношението между резултатите, получени с и без прогнозния модел. Графиката визуализира ефективността на модела. Колкото по-голяма е площта между кривата на повдигане и основната линия, толкова по-добър е модела. Генерира се чрез изчертаване по оста *y* на сбора на истинските позитивни прогнозирани резултати спрямо ос x с позитивна стойност.

Червената линия в фиг. 4.17 представлява базова линия спрямо която се измерва успеваемостта на използвания модел. Кривата на повдигане (в синьо) показва колко по-добър е избраният модел спрямо употребата на случаен подход (линия в червено).



Фигура 4.17: Lift curve

Благодарение на визуалното представяне на резултатите чрез различни по тип диаграми може лесно да бъде установена ефективността на използвания модел преди да бъде интегриран в работна среда.

4.2.2 Регресионен анализ

Регресионният анализ е съвкупност от статистически методи за оценка на характера на взаимоотношенията между променливите. Той включва много техники за моделиране и анализ на променливите, когато фокусът е върху връзката между зависима променлива и независими променливи (предиктори). Чрез регресионния анализ се разбира как стойността на зависимата променлива се променя, когато някоя от независимите променливи се променя, докато другите независими променливи са фиксирани.

Регресията и корелацията са двата анализа, основани на мултивариантно разпределение. То се описва, като разпределение на множество променливи. Корелацията се описва като анализ, който ни позволява да познаем връзката или липсата на връзка между две променливи „x“ и „y“. От друга страна, регресионният анализ, прогнозира стойността на зависимата променлива на базата на известната стойност на независимата променлива, като се приеме, че е средната математическа връзка между две или повече променливи.

4.2.2.1 Прилагане на методи за избор на променливи

Опознаването на данните е една от първите стъпки, които трябва да бъдат извършени преди да бъде направен избор за променливите, които ще бъдат включени в процеса на работа. Предвид това, на етап „Селекция на променливи, базирана на филтри“ познаването на данните играе съществена роля за избор на метод. С най-висок коефициент на влияние е важно да бъде променливата, която реално има най-голямо въздействие върху крайния резултат. А коефициентите с резултат 0 и ≈0 да бъдат променливите, влияещи най-малко. Чрез тези коефициенти обучението е по-пълноценно и се постига по-висока точност, защото се взима предвид коефициентът на всяка една променлива от избрания метод.

С цел постигане на висока точност, намаляване на шума и времето за обучение прилагаме 5 различни метода. След прилагане на различните методи (таблица 4.9) за избор на променливи се установява, че най-подходящият метод за целите на изследването е корелация на Пиърсън, която описва най-добре степента на влияние на променливите. Тя показва, че променливите величини са количествени, имат нормално разпределение и зависимостта между тях е линейна.

Таблица 4.9: Сравнение на получените резултати при прилагане на различни методи за избор на променливи..

	Температура	**Влажност**	**Атмосферно налягане**	**Вятър - скорост**	**Часови диапазон**
Пиърсън	**0,58**	**0,28**	**0,44**	**0,09**	**0,01**
Кендал	0,33	0,18	0,45	0,029	0,001
Чи-квадратична статистика	146,86	97,07	235,20	40,84	2,29
Фишър	2,45	1,84	1,06	0,92	0,91
Спирмън	0,47	0,26	0,61	0,038	0,002

Получените резултати (фиг. 4.18) показват, че променливите температура, атмосферно налягане, влажност, скорост на вятъра и час са най-определящите фактори,

с различен коефициент на влияние върху промяната на количеството мед в пчелен кошер.

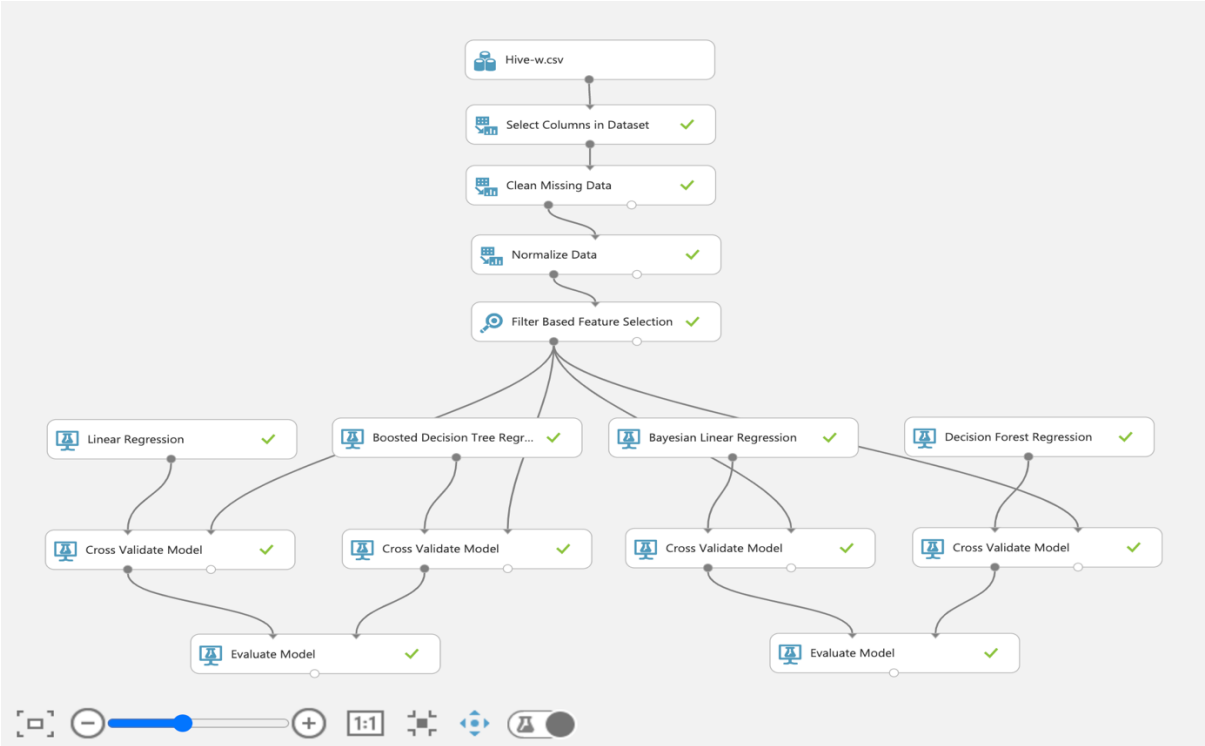
Фигура 4.18 Получените резултати при прилагане на корелация на Пиърсън

4.2.2.2 Работен процес по изграждане на регресионен модел

След извършен избор на колони за работа от базата, нормализиране на данните, изчистване на липсващите данни и направен избор на променливи за работа се извършва обучение на четири модела - Linear Regression, Boosted Decision Tree Regression, Bayesian Linear Regression и Decision Forest Regression (фиг. 4.19).

За трениране и тестване на избраните модели се прилага кръстосано валидиране (cross-validation). С него създаваме 10 на брой разделения на данните и съответно 10 на брой гънки. Към кръстосаното валидиране се прилага стратификация, чрез която взимаме данни от различни места от набора от данни, чрез което се подсигуряваме, че всяка гънка е добър представител на цялата съвкупност от данни.

Оценката на модела (Evaluate model) е финална стъпка в целия работен процес при която се изчислява набор от стандартни метрики за оценка, които показват размера на грешката.



Фигура 4.19: Работен процес по изграждане на регресионен модел

4.2.2.3 Резултати от извършен работен процес по изграждане на регресионен модел

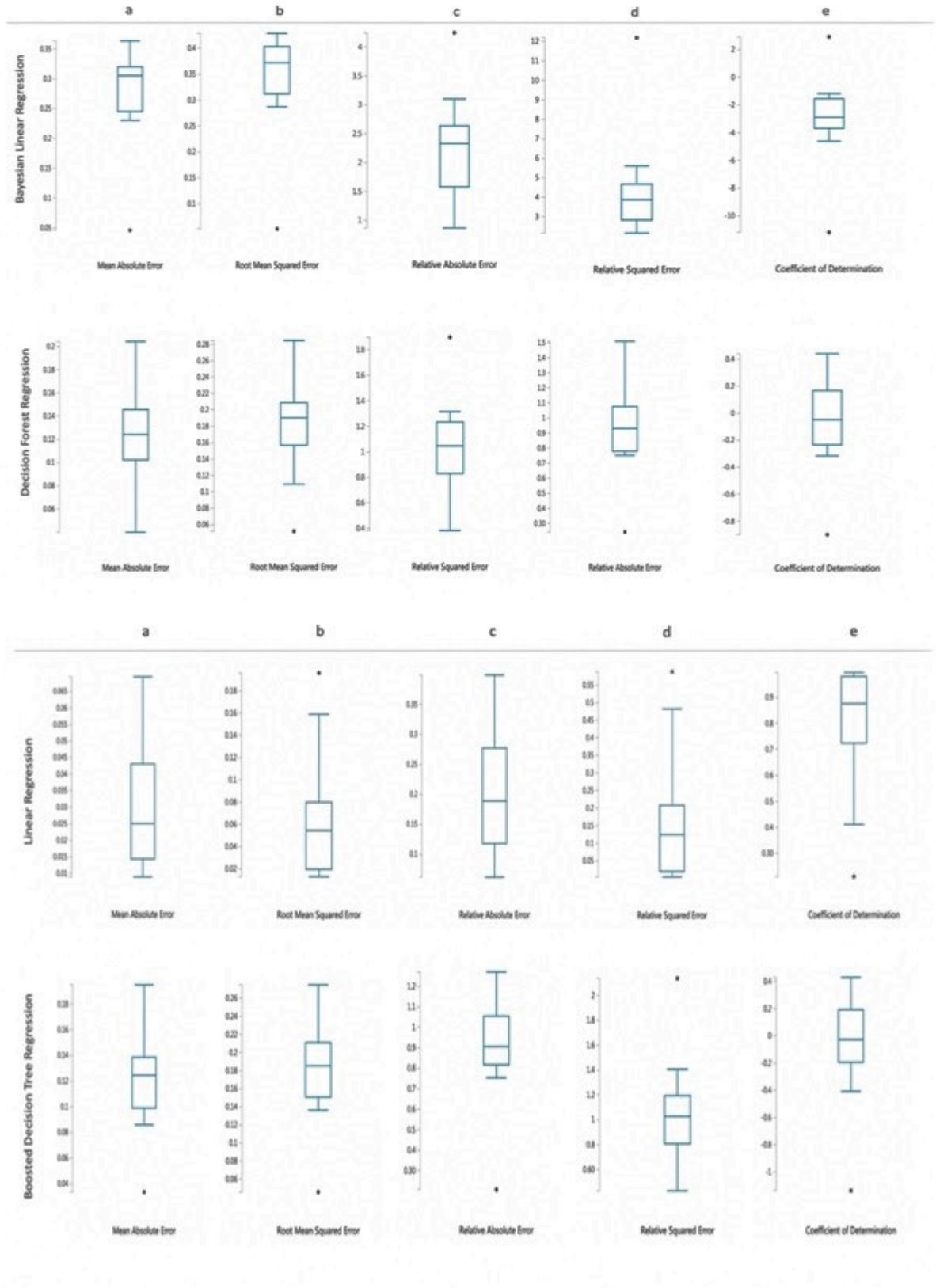
Получените резултати след извършено обучение и тестване на избраните модели са определящият фактор за правилен избор на модел. В таблица 4.10 са представени получените метрики след трениране и тестване на четири регресионни модела.

Таблица 4.10: Сравнение на получени метрики от приложените четири алгоритъма

Метрики	Linear Regression	Boosted Decision Tree Regression	Bayesian Linear Regression	Decision Forest Regression
Средна абсолютна грешка	**0.031631**	0.100482	0.066806	0.087543
Средна квадратична грешка	**0.092386**	0.145703	0.098254	0.138833
Относителна абсолютна грешка	**0.229573**	0.729287	0.484872	0.635377
Относителна квадратична грешка	**0.209455**	0.520975	0.236910	0.473003
Коефициент на детерминация	**0.790545**	0.479025	0.763090	0.526997

Таблично представеното сравнението на получените метрики за всеки един от приложените регресионни модели показва, че линейната регресия е моделът, който най-добре описва статистическата връзка между разглежданите променливи. Средната абсолютна грешка (Mean Absolute Error (MAE)) при прогнозиране с линейна регресия е със стойност 0,031631, която представлява средната стойност на абсолютната разлика между реалните точки на данните и прогнозният резултат (фиг. 4.20-a). Средната квадратична грешка (Root Mean Squared Error (RMSE)) при линейна регресия е със стойност 0,092386, която показва стандартното отклонение на грешките при прогнозиране. Стойността отразява отдалечеността на данните от регресионната линия (фиг. 4.20-b). Относителна абсолютна грешка (Relative Absolute Error (RAE)) при линейна регресия е със стойност 0,229573, която показва несъответствието между точната стойност и приближението (фиг. 4.20-c). Относителна квадратична грешка (Relative Squared Error (RSE)) при линейна регресия е със стойност 0,209455, което показва средната стойност на квадратичната грешка на действителните стойности (фиг. 4.20-d). Коефициент на детерминация (Coefficient of Determination (R^2)) при линейна регресия е със стойност 0,790545, която отразява степента, до която зависимата променлива е предсказуема (фиг. 4.20-e). Също така тя може да бъде интерпретирана,

като пропорция на вариацията в зависимата променлива, която е предсказуема от независимата променлива.

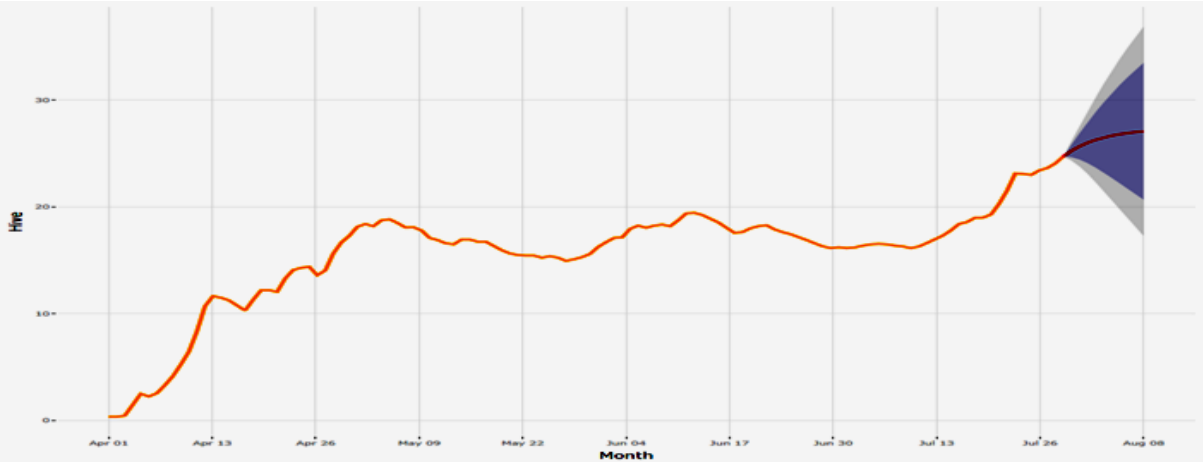


Фигура 4.20: BoxPlot на метрики на линейна регресия

4.2.2.4 Предсказване на резултати в краткосрочен план

Поради спецификата на разглежданата задача е важно да бъдат получени максимално точни данни в краткосрочен план. Прогнози отразяващи дълъг период от време биха били неуместни и нереалистични.

Фигура 4.21 отразява прогнозни количества за период от 10 дни, които играят определяща ключова роля за краткосрочното планиране на дейностите.



Фигура 4.21: Краткосрочна прогноза на количествата

Благодарение на получените прогнозни резултати пчеларите имат възможност да планират своята логистика за посещение на своите пчелини, което ще спести тяхното време и средства. От друга страна добрата организация ще доведе до по-малко количество на инспекции на пчелните кошери, които влияят неблагоприятно върху пчелните семейства поради настъпваща промяна на техния изграден благоприятен микроклимат, който ще се измени в следствие на отварянето на пчелния кошер.

4.3 Изводи

В тази глава се прилага и валидира предложената в Глава 2 методология за обработка, моделиране и интеграция на хетерогенни данни от разпределени IoT устройства. Експериментът е извършен в облачната изчислителна среда на Microsoft - Azure Machine Learning Studio. Изградени са работни схеми, съдържащи всички необходими стъпки от представената методология, които са необходими за решаването на дефинираните проблеми.

Разгледани и са решени задачи за класификация и регресия. При решението на класификационния проблем са разгледани два варианта с цел да бъдат обхванати и успешно валидирани всички стъпки от етапите „Обработка на данни“ и „Моделиране“ от методологията.

В резултат на извършените експерименти са направени следните заключения:

1. Методологията е гъвкав и систематичен процес, който е необходимо да бъде следван, за постигане на правилен резултат. Пропускането на стъпка в процеса на работа води до неверен резултат (overfit или underfit).
2. Изборът на един измежду няколко алгоритъма, решаващи една и съща задача, зависи от неговите качествени показатели. Като едни от най-важни качествени показатели на моделите се приема времето, необходимо за тяхното изпълнение, точността на получените резултати, прецизността и чувствителността на данните към промени.
3. Правилното конфигуриране на всеки алгоритъм значително увеличава точността на получения краен резултат.
4. Балансирането на данни при решаване на класификационни задачи играе ключова роля за успешното генерализиране на данните от модела.
5. Определянето на най-пряко зависещи променливи към предмета на разглежданата задача повишават точността и прецизността на модела, което води до най-добри крайни резултати.

С цел независимост на IoT системата от Azure облачни услуги за обработка и моделиране на данни, е разработено решение за валидиране на предложената методология с използване на програмен език Python. Решението следва всички стъпки представени в методологията. Моделите за машинно обучение (за класификация и регресия) и тяхната правилна конфигурация се съхраняват във файлов формат Pickle. Програмният код може да бъде намерен в **Приложение 4А и 4Б**.

Получените резултати са отразени в публикациите:

[3] Dineva, K., Atanasova, T. Regression Analysis on Data Received from Modular IoT System. ESM'2019, EUROSIS-ETI, 2019, 114-120, **Scopus.**

[5] Dineva, K., Atanasova, T. ICT-based Beekeeping using IoT and Machine Learning, DCCN 2018, Revised Selected Papers, Springer, 2018, ISBN:978-3-319-99446-8, ISSN:1865-0929, 132-143. **SJR:0.17**

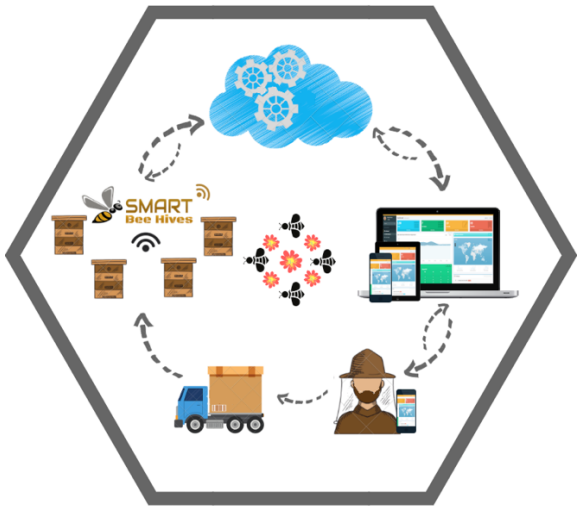
[7] Dineva, K., Atanasova, T. Applying Machine Learning against Beehives Dataset. SGEM 2018, 18, 6.2, SGEM 2018, ISBN:978-619-7408-51-5, ISSN:1314-2704, 35-42. **SJR:0.209.**

Глава 5. Практическо приложение на разработената IoT система

В тази глава на дисертацията се отразява едно от многото практически приложения на предложената модулна IoT система. Разгледана е значимостта от интегриране на мониторингова система за сектора, ползите и положителните ефекти, които тя оказва върху отрасъла. Изготвено е кратко описание на съществуващите аналогични решения на пазара на база, на което е направен сравнителен анализ на тези решения спрямо разработената в този дисертационен труд система.

С цел по-лесна разпознаваемост на представената IoT система, тя е наречена *SmartBeeHives***.**

IoT системата SmartBeeHives съчетава основните седем компонента, които ICT обхваща – софтуер, хардуер, транзакции, комуникация, данни, интернет достъп и облачни изчисления. Чрез съчетанието на тези компоненти системата извършва цялостен процес (фиг. 5.1) - събиране, предаване, обработка, анализ, моделиране, визуализиране на данни и предсказване на събития.



Фигура 5.1. SmartBeeHives работен процес

Процесът започва от имплементиране на хардуерни устройства в пчелни кошери, които събират хетерогенни данни за микроклимата в кошера и атмосферните условия извън него. Тези данни се предават към облачна среда посредством интернет. В облачната среда се извършва трансформация на данните с цел извършване на точни изчисления и получаване на полезна информация. Получената информация се визуализира в потребителски интерфейс, където потребителят може да следи

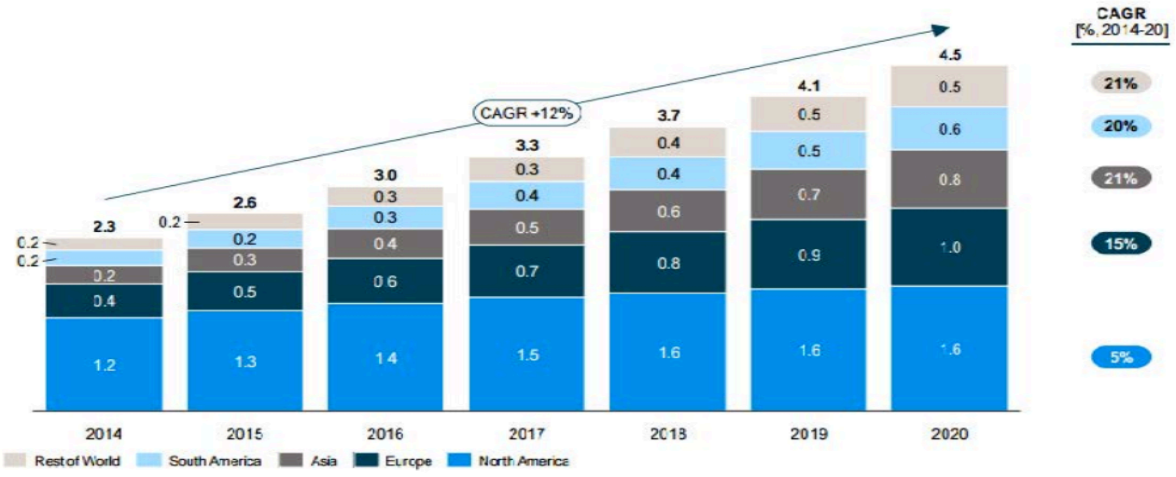
развитието на пчелните семейства в кошерите и да бъде уведомен за бъдещо настъпване или моментно наличие на заплаха. Въз основа на типа на тези сигнали пчеларят може да предприеме своевременни мерки – да посети пчелина или да изпрати обратна команда през потребителския интерфейс към системата.

5.1 Цифрово земеделие

Селското стопанство заема важно място в световната икономика. Натискът върху отрасъла се увеличава с тенденциозното нарастване на човешкото население. Според оценки на ООН, населението на света ще се увеличи до 9,6 милиарда до 2050 г., като се очаква световното производство на храни да се увеличи със 70%, за да отговори на новото търсене (Bruinsma J., 2017). Същевременно обаче селското стопанство е изправено пред сериозни предизвикателства като ограничената наличност на обработваема земя, нарастващото търсене на прясна вода, различни последици от изменението на климата и значителното намаляване на пчелните популации, които се грижат за опрашването на 80% от земеделските култури.

Едно от решенията на тези проблеми е използването на интелигентни земеделски подходи, които се обединяват в термина „цифрово земеделие“.

Цифрово земеделие, възниква като нова научна област, която използва данни и интензивни подходи за стимулиране и подобряване на селскостопанската производителност, като същевременно се минимизира нейното въздействие върху околната среда.



Фигура 5.2 Обобщена пазарна оценка на цифровото земеделие 2014-2020

[Източник: Strategy Consultants GmbH]

Фигура 5.2 показва обобщена пазарна оценка на цифровото земеделие за периода от 2014 до 2020 за Северна Америка, Южна Америка, Европа и Азия (Berger,

2015). Данните от диаграмата показват, че европейският селскостопански сектор ще трябва да се дигитализира по-бързо, ако иска да остане конкурентоспособен в световен мащаб. Според изчисления въз основа на данните, събрани от Евростат и Roland Berger (Eurostat, 2018) (Berger, 2019), цифровото земеделие възлиза на по-малко от 0,5% от европейския селскостопански сектор. В сравнение с това, средния растеж на цифровото земеделие в световен мащаб се оценява на 12% за периода 2014—2020 г.

Доказването на стойността на цифровото земеделие като двигател на рентабилността е от решаващо значение за земеделските производители да започнат да инвестират в перспективни цифрови технологии.

5.2 Технологиите и пчелите

Днес пчеларството е в челните редици на селскостопанските дейности, които се извършват широко в света. Поради развитието на технологиите и намаляването на дейностите по пчеларството поради екологичните и климатични характеристики навсякъде по света, тя се превърна във важна област на научноизследователска дейност (Braga A., 2020) (Beecham, 2017).

5.2.1 Прецизно пчеларство

Благодарение на развитието на технологиите на "Интернет на нещата" (IoT) е настъпило време за цифровизация на пчеларството (Peter C., 2016) и въвеждане на термина "прецизно пчеларство".

Прецизното пчеларство има за цел да помогне на пчеларите да наблюдават от разстояние пчелните колонии и да идентифицират различни състояния, включително девиантно поведение, тъй като това е един от факторите, които могат значително да увеличат рентабилността и да дадат пълен поглед над състоянието и развитието на всяко отделно пчелно семейство (Jiang J., 2016).

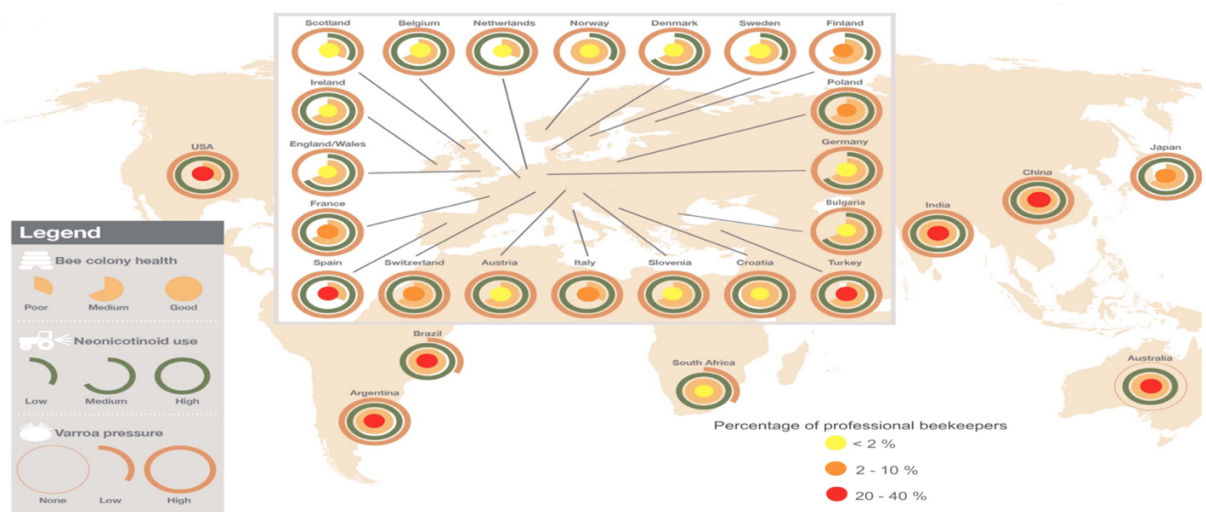
С развитието на сензори и комуникационни устройства могат да се реализират по-точни измервания в пчеларството и тези данни да бъдат доставени и обработвани по-бързо, по-точно и на по-ниска цена.

Прецизното пчеларство използва съвременни информационни технологии, като предпазни мерки срещу заплахите за безопасността на пчелите от човешки фактор, глобално затопляне и болести (Zacepins A., 2015).

5.2.2 Проблеми в пчеларството

Пчеларството е една от най-старите известни селскостопански дейности. Според специалисти пчелите са барометър за влиянието на човешката дейност върху природата и за равновесието в околната среда (Gallai N., 2009).

Към днешна дата статистиката показва тревожни факти за съществуването на пчелните колонии – те са изправени пред голямо предизвикателство. Според годишно изследване в световен мащаб (Nosowitz D., 2016), близо 44% от колониите на пчелните семейства са загубени през последните 10 години (при приемлива загуба от 16,9%). След извършени редица проучвания в областта са откроени три основни причини (диаграма 5.3) за проблеми с нарастващия процент смъртност на пчелните колонии (Delaplane K., 2000).



Фигура 5.3: Състояние и проблеми в отрасъла.
Източник: Syngenta [адаптирано]

На първо място е поставен проблемът с нерегламентираното пръскане с пестициди върху земеделски култури по време на опрашващия период намиращи се в близост до пчелни кошери (Syngenta, 2017). Друг основен проблем, пряко допринасящ за смъртността на пчелите, е Varroa mite – паразит атакуващ пчелите. Проблемът с този паразит е силно разпространен в Европа и Северна Америка и води до катастрофални загуби от 2006 година насам (Conte Y., 2010).

Освен тези два фактора съществува и трети, който също силно се отразява на здравето и продуктивността на пчелните семейства – компетентността и професионализмът на пчеларя, както и времето и средствата, влагани в отглеждането на пчелни колонии. От особена важност е извършването на редовни инспекции на

всеки един кошер. Често пчеларските инспекции са една от най-сложните задачи дори и за професионалните пчелари.

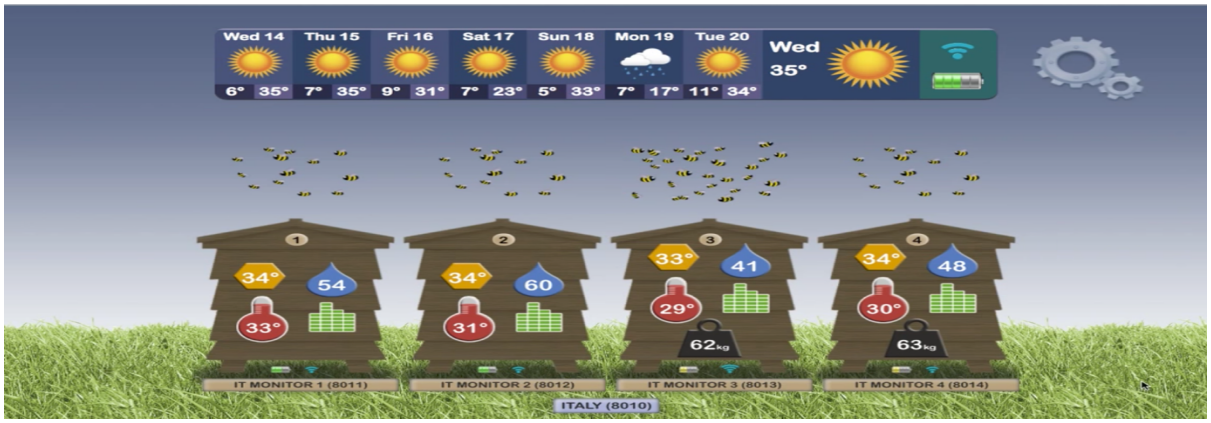
Проучвания показват, че голяма част от пчелните кошери по света са разположени извън населени места. Извършването на необходимия брой инспекции на всеки кошер е затруднено поради различни причини, като възникнали влошени метеорологични условия, лоша или липсваща инфраструктура или поради нарастващия размер на транспортните разходи, което от своя страна води до намалена рентабилност. От друга страна извършването на инспекции има неблагоприятен ефект върху цялостното състояние на пчелното семейство. Изследвания показват, че пчелите се нуждаят от 3 до 4 дни, за да се стабилизират и да възстановят идеалните параметри на температура и влажност в кошера (Verboven H., 2014).

Всичко това води до нуждата от вграждане на нови технологии в пчеларството, чрез които да бъде извършван отдалечен мониторинг на пчелните кошери и извличане на знания и ползи от получените данни от модулна IoT мониторинговата система.

5.2.3 Преглед на съществуващи системи и разработки

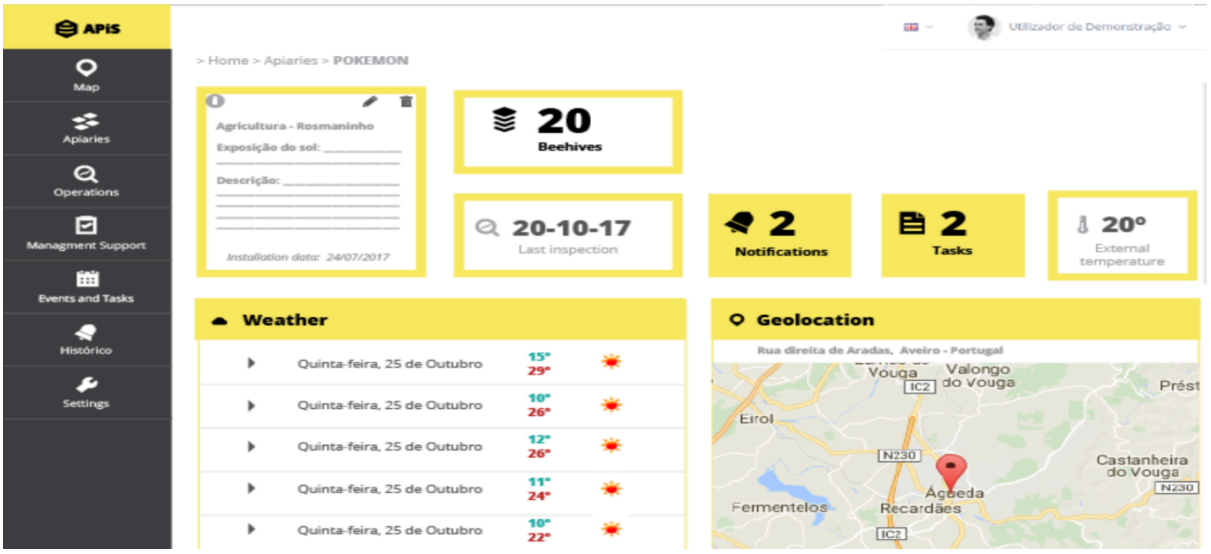
Все по-често последните технологични изложения обръщат внимание на "Интернет на нещата" (IoT) като технология, която спомага за устойчивото развитие и опазване на околната среда. Мнозина проекти свързани с IoT и опазване на околната среда получават финансиране и започват своята развойна дейност. Част от тези проекти са свързани със земеделието и опазването на пчелните колонии.

Arnia - калифорнийската система *Arnia* е пример за такъв проект с фонд от 1,4 милиона евро, усвоен за три годишен период (Arnia, 2016). *Arnia* представлява IoT система за наблюдение на кошери, която събира данни за температура, влажност, акустика, активност и теглото на всеки кошер. Събраните данни от системата се визуализират в специално изградена за целта софтуерна платформа (фиг. 1.2), която също така предоставя информация за прогноза за времето и възможност на всички регистрирани потребители да споделят собствените си данни в глобален форум. Платформата дава възможност за графично представяне на извлечените данни през различни периоди и проследяване развитието на всеки кошер в определена времева рамка. *Arnia* предлага тази сензорна система за $380 на кошер и вариращ месечен абонамент в зависимост от броя на кошерите под наблюдение.



Фигура 5.4: Потребителски интерфейс на Arnia

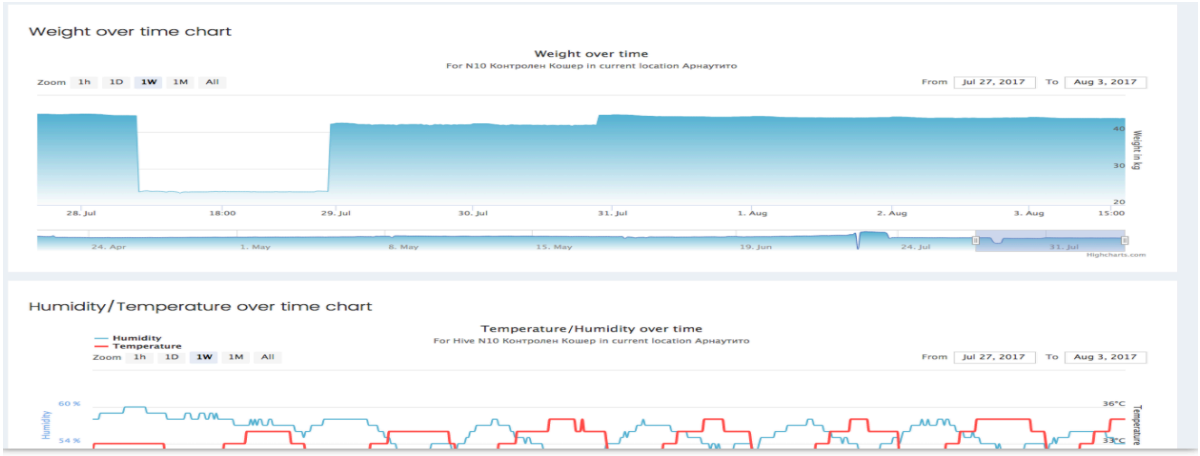
ApiS - Системата *ApiS*, разработвана от португалски екип с дарителски фонд от 2 милиона евро, представлява IoT система със сензори, събиращи хетерогенни данни от пчелни кошери (Apis, 2015). Събираните данни от системата са за температура, звук, влажност и тегло. Също се събират и данни за точното местоположение на кошера (GPS координати). Събраните данни се изпращат и визуализират в *ApiS* платформа (фиг. 5.4). Визуализирането им се извършва чрез диаграми, от които може да се проследи изменението на параметрите във всеки кошер. Системата APIS има изграден за целите на пчеларството софтуер за планиране на ресурсите (Enterprise Resource Planning - ERP). Цената на системата е разделена на пакети, като най-предпочитан е професионалният пакет на стойност $1899, който включва един модул за наблюдение на 10 кошера.



Фигура 5.5: Потребителски интерфейс на APIS

Bee Smart Technology - българска напълно интегрирана мониторингова система за пчелни кошери, спечелила награден фонд от 300 000 долара (Technologies, 2016). Системата се състои от IoT сензорна станция, която позволява получаване на критични данни за дадена пчелна колония. Станцията за диагностика на кошери следи за изменение в температурата,

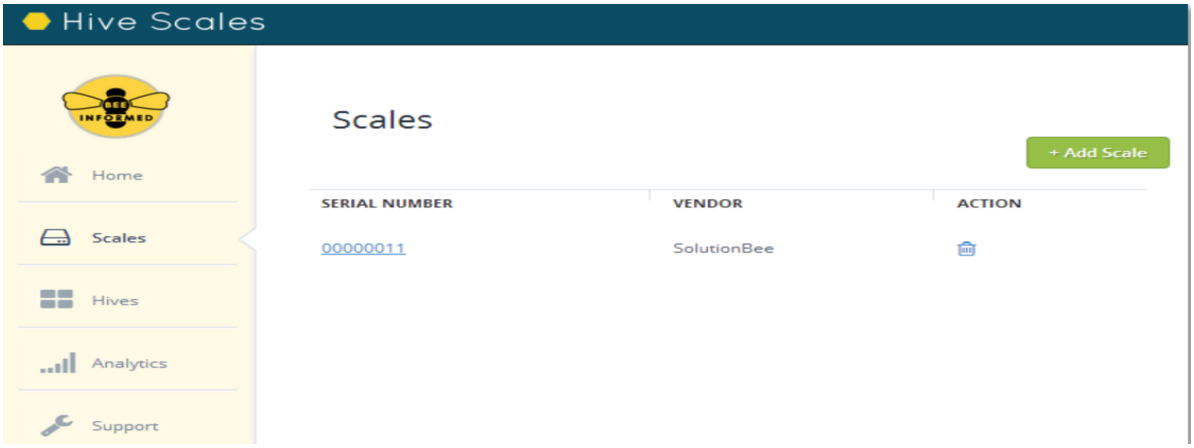
влажността, теглото и акустиката на контролните кошери. Всички данни, събрани от поставени в и извън пчелните кошерите сензори, се визуализират чрез графики в изградена за целта софтуерна платформа (фиг. 5.6). Платформата позволява, освен визуализация на събраните данни и възможност за водене на бележки, дигитална проверка на кошерите и бърза справка за метеорологичната обстановка на пчелина. Цената на вътрешно кошерната сензорна станция е в размер на 189 лв. за кошер, а цената на кошерният кантар е 499 лв.



Фигура 5.6: Потребителски интерфейс на Bee Smart Technologies

BeeHive Scales - BeeHive Scales е американска система за мониторинг на пчелни кошери. Тя е насочена както за любители пчелари, така и за професионални ферми (Scales, 2012). Системата събира данни за температура в и извън кошера, ниво на влажност, количество валежи, скорост на вятъра и тегло на кошера. Събраните данни се визуализират чрез диаграми в софтуерната платформа “Hive Scales” (фиг. 5.7). Потребителският интерфейс разполага и с опция за незабавно известяване на пчеларя в случай на отклонения в предварително зададените от потребителя параметри.

Цената на вътрешно кошерната сензорна станция е в размер на 89 € за кошер, а цената на кошерната везна е 189 €. Месечният абонамент за употреба на софтуерната платформа е в размер на 49 €.

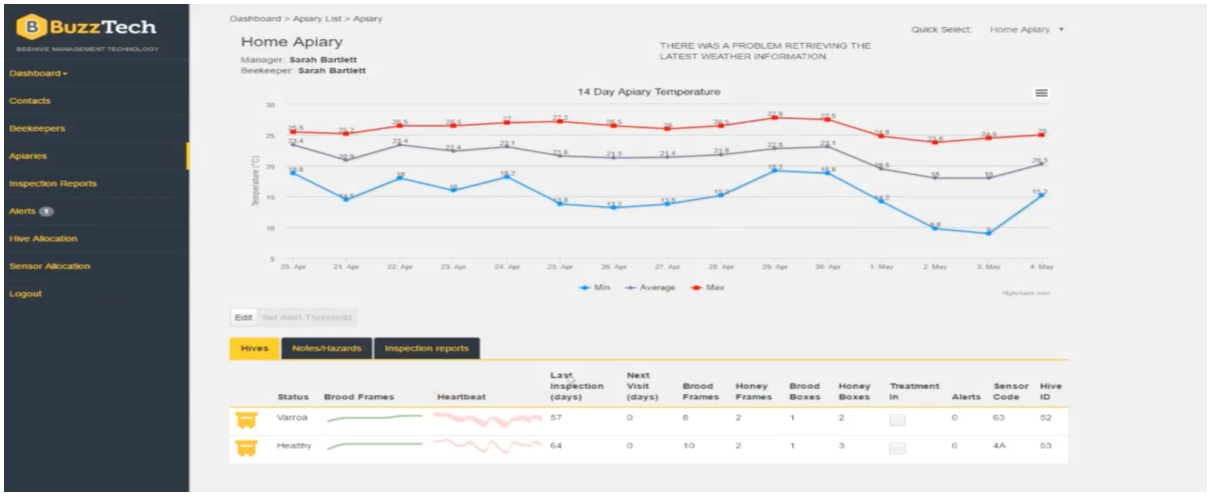


Фигура 5.7: Интерфейс на BeeHive Scales

BuzzTech - мониторингова система за пчелни кошери разработена в Нова Зеландия, която се намира в заключителен етап на инвестиционния процес по програма "Sprout AgriTech Accelerator". BuzzTech предоставя на своите потребители софтуер за управление на пчеларските процеси и IoT система за извличане на данни от пчелни кошери и опция за предоставяне на решения за правилно хранене на пчелната колония (BuzzTech, 2017).

IoT системата на BuzzTech извлича данни за температурата по пет пъти дневно и прави запис от две секунди на всеки час от горната част на разплодната камера в пчелния кошер. Акцентът на софтуерът за управление на пчеларските процеси „ApiBuzz Apiar Management System" (фиг. 5.8) е поставен върху преподаването и практикуването на холистични методи чрез инициатива за градско пчеларство.

Системата е насочена към големи пчелни ферми с повече от няколко стотин кошера. Цената на IoT системата е $ 625 за кошер с месечен абонамент от $ 30 за употреба на софтуерната платформа.



Фигура 5.8 Интерфейс на BuzzTech

5.2.4 Сравнителен анализ на разработената модулна IoT система със съществуващи системи

В таблица 5.1 е представен сравнителен анализ между съществуващите системи и ново изградената система SmartBeeHives.

Таблица 5.1: Сравнителен анализ на съществуващите системи

	Smart BeeHives	***BTS***	***Arnia***	***Apis***	***BeeHive Scales***	***BuzzTech***
Сензори - вътре						
Температура	√	√	√	√	√	√
Влажност	√	√	√	√	√	x
Звук	√	√	√	√	x	√
Тегло	√	√	√	√	√	x
Брояч на пчели	x	x	√	√	x	x
GPS	√	√	√	√	x	x
Wireless	√	√	x	x	x	√
Сензори – вън						
Температура	√	√	√	√	√	x
Влажност	√	√	√	x	√	x
Атмосферно налягане	√	x	x	x	x	x
Газ	√	x	x	√	x	x
Скорост на вятъра	√	x	x	x	√	x
Детектор за движение	√	x	x	√	x	x
Потребителски интерфейс						
Диаграми	√	√	√	√	√	x
Прогноза за времето	√	√	√	√	x	x
Календар	√	x	x	√	x	x
Дневник	√	√	√	√	x	x
Органайзер	√	x	x	x	x	x
Анализ на данните	√	√	x	√	x	x
Статистика на данните	√	x	x	x	x	x
ERP	x	x	x	√	x	x
Дистанционно наблюдение	√	√	√	√	√	√
Наблюдение в реално време	√	x	x	√	x	x
Предсказване на бъдещи събития	√	x	x	x	x	x
Характеристики						

Модулна архитектура	√	x	x	x	x	x
Двустранна комуникация	√	x	√	√	x	x
Самоанализ	√	x	x	x	x	x
Отворен код	√	x	x	x	x	x
Филтриран данни	√	x	√	x	√	x
Скалируемост	√	√	√	√	x	x
Поддръжка						
Живот на батерията	***6 м.***	***6 – 9 м.***	***3 – 6 м.***	***Слънчев панел***	***-***	***Слънчев панел***
Цена на кошер за устройство	***≈ €100***	***€ 372***	***€ 380***	***€ 1899***	***€ 278***	***$ 625***
Месечен абонамент за платформата	***0***	***€ 2***	***€ 11***	***NAN***	***€49***	***$ 30***

От сравнителния анализ може да бъде изведено, че разработената в дисертацията система SmartBeeHives се отличава от съществуващите системи с:

- Хардуерни компоненти – намалява се зависимостта от хардуерни компоненти от конкретен производител или тип.
- IoT устройства за измерване на атмосферните условия извън кошера.
- Лесно премахване и добавяне на модули без да се налага промяна във функционалността на системата – полезно при увеличаване или намаляване на брой кошери в пчелин.
- Двупосочен обмен на данни - устройствата поддържат безжична комуникация.
- Самоанализ на системата. По този начин данни събирани от дефектирали устройства не се вземат предвид при извършването на анализ. Потребителят бива уведомен за конкретните дефектирали устройства и нуждата да бъде извършен контрол над тях.
- Сървис за обработка на събраните данни.
- Сървис за машинно обучение, чрез който се предсказват бъдещи събития настъпващи в пчелните кошери.
- Софтуерна платформа с множество функционалности, позволяваща на потребителите да:
  - Проверяват данни за своите кошери в реално време.
  - Следят състоянието на своите кошери – чрез разбираеми диаграми;

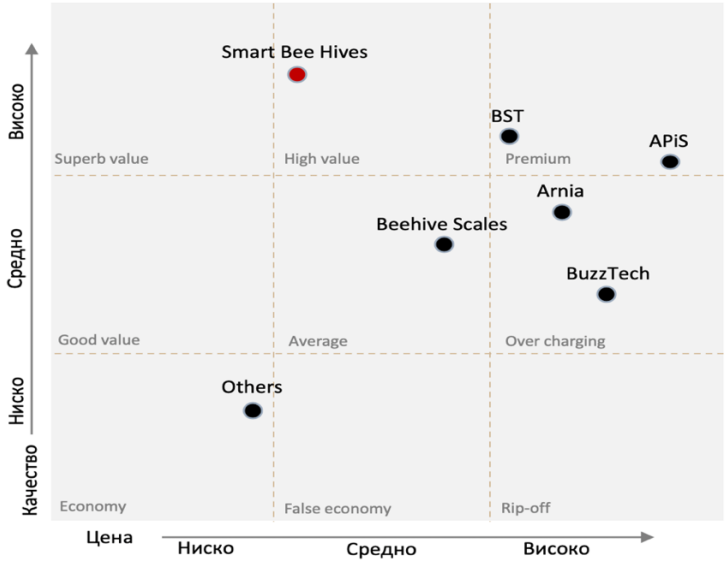
- Проверяват статистически данни за своите кошери – дава се възможност да се следи развитието на пчелните семейства за определени периоди от време.
- Получават известия при настъпване на анормални дейности с пчелните семейства – нерегламентирано отваряне на кошера, нерегламентирано преместване на кошера, рязко увеличаване нивото на замърсителите във въздуха (при пръскане с пестициди в околността);
- Попълват календар за минали и бъдещи дейности по пчелните семейства. Опция за напомняне, че наближава датата за извършване на предварително зададена дейност.
- Попълват дневник за развитието на всяко пчелно семейство – подхранвания, смяна на майка, роене, количество изваден мед, третирания, заболявания, състояние на пчелното семейство (силно, слабо или нестабилно) и др.

5.2.5 Матрица цена-качество за сравнение на разработената модулна IoT система със съществуващи системи

Матрицата цена – качество описва най-често срещаните стратегии чрез картографиране на цената спрямо качеството. Въз основа на модела на Котлър (Kotler, 2011) разглеждаме възможностите, които произтичат в зависимост от разположението на продукта в матрицата.

Позиционирането на продуктите в един от деветте квадранта показва съотношението цена-качество и как то се възприема от потребителите.

Възприеманата стойност от потребителите е съществен фактор при изграждането на мнение за продукта.



Фигура 5.9 Матрица цена-качество

Деветте квадранта описват различните видове съотношение между цена и качество:

1. Висока цена и високо качество (Premium)
2. Висока цена и средно качество (Over charging)
3. Висока цена и ниско качество (Rip off)
4. Средна цена и високо качество (High value)
5. Средна цена и средно качество (Average)
6. Средна цена и ниско качество (False economy)
7. Ниска цена и високо качество (Superb value)
8. Ниска цена и средно качество (Good value)
9. Ниска цена и ниско качество (Economy)

Златната среда за въвеждане на нов продукт е позиционирането му в квадрант “high value” – средна цена и високо качество спрямо предлаганите на общия пазар подобни продукти.

SmartBeeHives попада именно в тази златна среда, където качеството надхвърля цената. Това дава възможност на потребителите достъпно да се запознаят с продукта и да направят сравнение с продуктите позициониращи се в квадрант “Premium”, където високата цена отговаря на високо качество и от потребителска гледна точка това е полезна инвестиция.

Както може да бъде видяно на матрицата цена-качество наличните системи за мониторинг на пчелни кошери се разполагат в пет от общо деветте квадранта на матрицата. Това показва, че предлаганите системи са разнообразни по цена и брой предлагани функционалности.

От съществуващите на пазара системи „Bee Smart Technology“ и „Apis“ попадат в квадрант “Premium”. Този квадрант се характеризира с продукт поддържащ висока цена и високо качество - от потребителска гледна точка това е полезна инвестиция.

Системата „Arnia“ попада в квадрант „Over charging“, които се характеризира с висока цена и средно качество на продукта. „BuzzTech“ се позиционира на границата между два квадранта „Over charging“ и „Rip off“. Причина за заеманата позиция е намаленото внимание към хардуерната част на продукта за сметка на софтуерната. „BeeHive Scales“, „Hive Tracks“ and „Bee Tight“ се намират в съседни квадранти характеризиращи се с ниско качество и средна цена.

Съществуват редица фактори влияещи върху позиционирането на системите в отделните квадранти. Едни от факторите са желаната целева група и желаната пазарна

позиция. На база целевата група и достигането на желаната пазарена позиция се определя необходимия баланс между цена и качество.

5.3 Изводи

В тази глава е представено практическо приложение на разработената система за моделиране и интегриране на хетерогенни данни придобити от IoT устройства.

Като сфера на приложение е разгледан сектор пчеларство. Анализирана е нуждата от интегриране на нови технологии в този отрасъл и ползите, до които те ще доведат. Изготвен е сравнителен анализ между съществуващите системи на пазара, на база на който е направена икономическа обосновка на практическото приложение на системата в този сектор.

В резултат на това могат да бъдат направени следните изводи:

1. Новите технологии дават възможност на пчеларите да направят дейностите си по-ефективни и оптимизирани, като предоставят възможност за отдалечен мониторинг и ранно известяване при настъпване на отклонения в нормалните параметри в пчелния кошер.
2. Интегрираните IoT устройства позволяват събиране на данни за всички важни вътрешните параметри на пчелните семейства.
3. Съществуващите системи са насочени към големи пчелни ферми и не покриват нуждите на малки производители и любители на пчеларството. Предложената модулна система в този дисертационен труд позволява еднакво добре да бъдат задоволени функционалните изисквания както на големи производители, така и на любители.
4. Системата SmartBeeHives единствена съчетава основните седем компонента на ICT.
5. Системата позволява интегриране на вече съществуващи устройства, които да изпращат данни към софтуерната платформа с цел анализ и визуализация на данните.
6. Системата е много гъвкава поради разнообразието си от хардуерни компоненти от които тя може да бъде изградена.

Съдържанието на тази глава е отразено в:

[2] Dineva, K., Atanasova, T. Integrated Systems with Embedded Sensors for Digital Agriculture. 19th International Multidisciplinary Scientific GeoConference SGEM 2019, DOI:10.5593/sgem2019/6.1/S25.098, pp. 61-768. **SJR:0.209**

[11] Dineva, K., Analytical review of existing computer systems for monitoring of beehives, Proc. International Scientific Conference UNITECH'2017, ISSN: 1313-230X, pp. 148-152.

Заключение - резюме на получените резултати

В дисертационния труд е подробно анализирана значимостта на данните, придобити от разпределени IoT устройства, и тяхната роля за непрекъснато подобряване на бизнес процеси свързани с тях. Посочени са проблемни области и са дефинирани конкретни проблеми.

Предложена е нова систематизирана методология, която се състои от последователни етапи за обработка, моделиране и интегриране на хетерогенни данни. Методологията е валидирана в Облачна изчислителна среда, където са обучени и потвърдени регресионни и класификационни модели за машинно обучение, които се използват за получаване на полезни знания от събраните данни и прогнозиране на бъдещи събития.

Разработена и внедрена е обща хардуерна и софтуерна скалируема система от модулен тип за събиране и обработка на данни от разпределени IoT устройства. Тя поддържа ефективна работа с голям брой различни по вид интегрирани хардуерни компоненти и устройства, които общуват чрез използване на различни комуникационни протоколи.

Практическите резултати от работата на системата се визуализират в специално изграден потребителски интерфейс, който представлява точка на взаимодействие между потребителя и системата.

Разгледана е сфера на практическа приложимост на разработената система и е анализирано пазарното й позициониране спрямо други съществуващи системи.

С оглед на работата извършена в този дисертационен труд и резултатите, получени в хода на изследванията и изложени по-горе, могат да бъдат формулирани следните **научно-приложни приноси**:

1. Разработена е методология за интегриране на хетерогенни данни придобити от разпределени IoT устройства, която позволява обработка, моделиране и интеграция на тези данни, като са селектирани и избрани:
   - методи за работа с хетерогенни данни;
   - класификационни и регресионни алгоритми за машинно обучение;
   - метрики за оценка и валидиране на получени прогнозни резултати.
2. Разработена е архитектура на модулна хардуерна IoT система, състояща се от сензорни, управляващи и комуникационни модули. Разработен е нов метод за

комуникация между IoT устройства, основан на йерархична IP адресация с предложената логическа схема на групиране „Снежинка“.

3. Проектирана и реализирана е софтуерна MSA архитектура за съхранение, обработка и анализ на хетерогенни данни. Разработен е нов подход за организация на услуги за интелигентна обработка и обмен на данни в IoT системата, който повишава надеждността на функционирането на системата.

4. Създадени са модели за машинно обучение, които експериментално потвърждават разработената методология и са интегрирани в реализираната MSA софтуерна архитектура.

5. Показано е възможно приложение на разработената IoT система за интегриране на хетерогенни данни в интелигентно земеделие. Направен е сравнителен анализ на функционалните характеристики и пазарното позициониране на съществуващи аналогични системи, чрез който е доказана икономическата ефективност и целесъобразност на разработената IoT системата.

Насоки за бъдещи изследвания

Основните насоки за бъдещи изследвания върху тематиката на дисертацията включват:

- Интегриране на нови платформи за работа с големи данни в реално време като Hadoop и Spark.
- Изследване на процеси за подобряване на UX при използване на потребителския интерфейс през мобилни устройства.
- Изследване на сигурността на системата и анализ на потенциални нови атакуващи вектори. Интегриране на автоматизирани системи за сигурност.
- Изследване на нови хардуерни компоненти за съвместимост към съществуващите такива. Анализ и подобряване на жизнения цикъл на внедрените хардуерни компоненти.

Публикации по темата на дисертационния труд

1. **Dineva K.,** Atanasova T., Methodology for Data Processing in Modular IoT System. Distributed Computer and Communication Networks, 22-st International Conference, DCCN 2019, Springer Nature Switzerland AG 2019. V. M. Vishnevskiy et al. (Eds.): DCCN 2019, LNCS 11965, **2019**, https://doi.org/10.1007/978-3-030-36614-8_35, pp. 457 – 468, **Q2**, **SJR:0.283**
2. **Dineva K.,** Atanasova T., Integrated Systems With Embedded Sensors for Digital Agriculture. 19th International Multidisciplinary Scientific GeoConference SGEM 2019, 19, 6.1, SGEM, **2019**, ISBN:978-619-7408-88-1, ISSN:1314-2704, DOI:10.5593/sgem2019/6.1/S25.098, pp. 761-768, **Q4**, **SJR:0.232.**
3. **Dineva K.**, Atanasova T., Regression Analysis on Data Received from Modular IoT System. ESM'2019, EUROSIS-ETI, **2019**, pp. 114-120, **Scopus.**
4. **Dineva K.**, Atanasova T., Security in IoT Systems, *Proceedings 19th International Multidisciplinary Scientific Geoconference SGEM,* **2019***,* Vol. 19, Informatics, Geoinformatics and Remote Sensing, Issue 2.1, ISBN 978-619-7408-79-9, ISSN 1314-2704, DOI:10.5593/sgem2019/2.1, pp. 576-577, **Q4**, **SJR:0.232.**
5. **Dineva K.**, Atanasova T., ICT-based Beekeeping using IoT and Machine Learning, *Distributed Computer and Communication Networks,* 21-st International Conference, DCCN 2018, Revised Selected Papers, Vladimir Vishnevskiy, Dmitry Kozyrev (Eds.), Springer, Communications in Computer and Information Science (CCIS). 919, Springer, **2018**, ISBN:978-3-319-99446-8, ISSN:1865-0929, DOI: https://doi.org/10.1007/978-3-319-99447-5, pp. 132-143, **Q3**, **SJR:0.188**
6. **Динева К., Атанасова Т.,** Подходи и методи за анализ и обработка на данни в мониторингова система за пчелни кошери, Годишник на департамент „Телекомуникации", НБУ, **2018**, ISSN: 2534-854 X (online) No: 5, pp. 37-46.
7. **Dineva K.,** Atanasova T., Applying Machine learning against beehives Dataset. *18-th International Multidisciplinary Scientific Geoconference - SGEM 2018,* 18, 6.2, **2018**, ISBN:978-619-7408-51-5, ISSN:1314-2704, DOI:10.5593/sgem2018/6.2, pp. 35-42, **Q4**, **SJR:0.232.**
8. **Dineva K.,** Atanasova T., OSEMN process for working over data acquired by IoT devices mounted in beehives. Current Trends in Natural Sciences, 7, 13, University of Pitesti, **2018**, ISSN:2284-953X, pp. 47-53.
9. **Dineva K.,** Atanasova T., Computer System Using Internet of Things for Monitoring of Bee Hives, SGEM GeoConference, 27 - 29 November 2017, ISSN:1314-2704, DOI:10.5593/SGEM_GeoConference, **2017**, Vol. 17, Issue 63, pp. 169-176, **Q4**, **SJR:0.232.**
10. **Dineva K.,** Atanasova T., Model of Modular IoT-based Bee-Keeping System. European Simulation and Modelling Conference ESM'2017, EUROSIS-ETI, **2017**, ISBN:978-492859-00-6, 404-406, **Scopus.**
11. **Динева К.,** Аналитичен обзор на съществуващите компютърни системи за мониторинг на пчелни кошери, Международна научна конференция UNITECH'2017, 17-18 Ноември **2017**, Габрово, България, ISSN: 1313-230X, стр. II-148-II-152.
12. **Динева К.,** Интернет на нещата и устойчиво развитие на земеделието, Международна научна конференция AUTOMATICS AND INFORMATICS'2017, 4-6 Октомври **2017**, София, България, Sofia, Bulgaria, 2017, ISSN:1313-1850, стр. 309-312.

Забелязани цитирания

I. **Dineva, K.,** Atanasova, T.: Model of Modular IoT-based Bee-Keeping System, ESM'2017, Lisbon, EUROSIS-ETI, 404-406 (**2017**).

Цитиращи трудове:

1. Todor Balabanov, Ivan I. Blagoev, Zornitsa Atanassova, Greedy Genetic Algorithm Hybrid Solution of 1D Stock Cutting Problem, International Scientific Conference UNITECH 2018, November **2018**, Gabrovo, Bulgaria, ISSN 1313-230X, pp.307-312.
2. S. Šakanović, N. Dogru, D. Kečo and J. Kevrić, "Short-Term Prediction of Honey Production in Bosnia and Herzegovina using IoT," *2019 8th Mediterranean Conf. on Embedded Computing (MECO)*, Budva, Montenegro, **2019**, pp. 1-4. doi: 10.1109/MECO.2019.8760012.
3. Johannes Pirhonen. "A data transmitter using the GSM network", Bachelor's thesis, University of Tampere, Finland, Bachelor's Degree Program in Information and Electrical Engineering, Electrical Engineering, December **2019.**
4. Riste Poposki, Dejan Gjorgjevikj, Precision Apiculture - IoT System for remote monitoring of honeybee colonies, 17th International Conference on Informatics and Information Technology - CIIT2020, Ss. Cyril and Methodius University in Skopje, **2020.**

II. **Dineva, K.,** Atanasova, T.: Computer System Using Internet of Things for Monitoring of Beehives, Vienna, SGEM GeoConference, vol. 17, Issue 63, 169-176 (**2017)**.

Цитиращи трудове:

5. Balabanov T., I. I. Blagoev, Z. Atanassova, Greedy Genetic Algorithm Hybrid Solution of 1D Stock Cutting Problem, International Scientific Conference UNITECH 2018, November **2018,** Gabrovo, Bulgaria, ISSN 1313-230X, pp.307-312.
6. Braga, A.R., Rabelo, J.C., Callado, A.C., da Rocha, A.R., Freitas, B.M., Gomes, D.G. Beenotified! a notification system of physical quantities for beehives remote monitoring | [Beenotified! um sistema de notificac¸˜oes de grandezas f´ısicas para monitoramento remoto de Colmeias de abelhas], *Revista de Informatica Teorica e Aplicada*, 27(3), pp. 50-61, **2020**

III. **Dineva, K.,** Internet of Things in Help of Sustainable Agricultural Development, International Conference AUTOMATICS AND INFORMATICS'2017, 4-6 October 2017, Sofia, Bulgaria, John Atanasoff Society Of Automatics And Informatics, ISSN:1313-1850, pp.309-312, (**2017**).

Цитиращ труд:

7. Atanasova T., N. Bakanova, I. Blagoev, Analysis of Data from OIS To Discover and Model Process-Oriented Information, NVU "Vasil Levski", 14-15 June **2018**, Tom 9, pp. 106-111.

IV. **Dineva, K.,** Atanasova, T., OSEMN Process For Working Over Data Acquired By IoT Devices Mounted In Beehives, Current Trends in Natural Sciences, 7, 13, University of Pitesti, **2018**, ISSN:2284-953X, 47-53, http://www.natsci.upit.ro.

Цитиращи трудове:

8. Jae Deok Son, Sooho Lim, Dong-In Kim, Giyoun Han, Rustem Ilyasov, Ural Yunusbaev and Hyung Wook Kwon, Automatic Bee-Counting System with Dual Infrared Sensor based on ICT, *Journal of Apiculture* 34(1): 47-55 (**2019**) DOI: 10.17519/apiculture.2019.04.34.1.47
9. Blagoev I., Application of Time Series Techniques for Random Number Generator Analysis, Proceedings of XXII Int. Conference DCCN 2019, September 23-27, **2019**, Moscow, Russia, pp. 437-446. ISBN 978-5-209-09683-2, 2019.
10. Yadhunath R. , S. Srikanth, A. Sudheer and S. Palaniswamy, "Identification of Criminal Activity Hotspots using Machine Learning to Aid in Effective Utilization of Police Patrolling in Cities with High Crime Rates, " 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), IEEE, Bengaluru, India, **2019**, pp. 1-6.
11. Braga, A. R., Gomes, D. G., Rogers, R., Hassler, E. E., Freitas, B. M., Cazier, J. A. "A method for mining combined data from in-hive sensors, weather and apiary inspections to forecast the

health status of honeybee colonies", *Computers and Electronics in Agriculture*, Volume 169, February **2020.**

12. Braga A.R., Gomesa D. G., Freitas B.M., Cazier J.A. "A cluster-classification method for accurate mining of seasonal honey bee patterns", *Ecological Informatics*, Elsevier, **2020**

13. Braga, A. R., Modelos de Classificação Para Predição do bem Estar de Colônias da Abelha Apis Mellifera, Doutorado em Engenharia de Teleinformática, Fortaleza **2020**

14. Da Silva, Daniel; Rodrigues, Ícaro; Braga, Antonio; Nobre, Juvêncio; Freitas, Breno; Gomes, Danielo. An Autonomic, Adaptive and High-Precision Statistical Model to Determine Bee Colonies Well-Being Scenarios. In: WORKSHOP DE COMPUTAÇÃO APLICADA À GESTÃO DO MEIO AMBIENTE E RECURSOS NATURAIS (WCAMA), 11, 2020, Evento Online. Anais do XI Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais. Porto Alegre: Sociedade Brasileira de Computação, june **2020**, p. 31-40. ISSN 2595-6124. DOI: https://doi.org/10.5753/wcama.2020.11017.

V. Dineva, K., Atanasova, T. Applying machine learning against beehives dataset. SGEM 2018, 18, 6.2, SGEM 2018, (**2018)**, ISBN:978-619-7408-51-5, ISSN:1314-2704, DOI:10.5593/sgem2018/6.2, 35-42.

Цитиращи трудове:

15. Braga, A. R., Gomes, D. G., Rogers, R., Hassler, E. E., Freitas, B. M., Cazier, J. A. "A method for mining combined data from in-hive sensors, weather and apiary inspections to forecast the health status of honeybee colonies", *Computers and Electronics in Agriculture*, Volume 169, February **2020.**

16. Braga, Antonio Rafael, Modelos de Classificação Para Predição do bem Estar de Colônias da Abelha Apis Mellifera, Doutorado em Engenharia de Teleinformática, Fortaleza **2020**

VI. Dineva K., Atanasova T., Security in IoT Systems, SGEM 2019, Vol. 19, Informatics, Geoinformatics and Remote Sensing, Issue 2.1, ISSN 1314-2704, pp. 576-577, (**2019)**

Цитиращ труд:

17. P. Panev, S. Dimitrov, Innovative Technology for Increasing the Efficiency in Tubular Furniture Production Machine, 8th International Conference on Advanced Technologies (ICAT'19), August 26-30, 2019, Sarajevo, Bosnia and Herzegovina, E-ISBN: 978-605-68537-4-6 pp. 338-341**, 2019.**

VII. Dineva, K., Atanasova, T. Regression analysis on data received from modular IOT system. ESM'2019, EUROSIS-ETI, (**2019)**, ISBN: 978-9492859-09-9, pp. 114-120.

Цитиращ труд:

18. Tomov, P., Zankinski, I., Balabanov, T. "Training of Artificial Neural Networks for Financial Time Series Forecasting in Android Service and Widgets", Scientific journal "*Problems of Engineering Cybernetics and Robotics*", vol. 71, pp. 50-56**, 2019**.

VIII. Dineva, K., Atanasova, T. ICT-based Beekeeping using IoT and Machine Learning. Distributed Computer and Communication Networks, 21-st International Conference, DCCN 2018, 919, Springer, (**2018)**, ISBN:978-3-319-99446-8, ISSN:1865-0929, https://doi.org/10.1007/978-3-319-99447-5, pp. 132-143.

Цитиращ труд:

19. Pešović, U., D. Marković, S.Ď. Đurašević. "Remote monitoring of beehive activity*"*. *Acta Agriculturae Serbica*, Vol. XXIV, 48 (**2019**); 157-165

Награди

1. Награда на ИИКТ-БАН за отлични научни постижения през 2019 г. в категория „Докторанти".

Декларация за оригиналност на резултатите

Декларирам, че дисертацията съдържа оригинални резултати, получени, при проведени от мен, научни изследвания с подкрепата и съдействието на научния ми ръководител.

Резултатите, които са получени, описани и/или публикувани от други учени, са коректно и подробно цитирани в библиографията.

Настоящият дисертационен труд не е прилаган за придобиване на научна степен в друго висше училище, университет или научен институт.

Подпис: …….

/Кристина Динева/

Благодарности

Бих желала да изразя своите най-искрени благодарности към своя научен ръководител доц. д-р Татяна Атанасова, за многобройните дискусии проведени между нас, нейните ценни съвети от научно и практическо естество, постоянната подкрепа и мотивация през годините, които спомогнаха за моето научно развитие и реализацията на настоящия дисертационен труд.

Благодаря!

Библиография

1. **Abraham T., Parasher Sh.** Hands-on Machine Learning with Azure [Book]: Packt, 2018.

2. **Agarwal K., Uniyal P., Virendrasingh S., Duff V.** Spam Mail Classification using Ensemble and Non- ensemble Machine Learning Algorithms [Book Section]: Springer SIST, Springer, 2020.

3. **Alekseev I., Nikitinskiy M.** Eventbus module for distributed openflow controllers [Conference]:17th Conference of Open Innovations Association (FRUCT) - IEEE, 2015.

4. **Ali A., Inglis A., Prado E., Wundervald B.** Bayesian Linear Regression [Report]: National University of Ireland, Maynooth, 2019.

5. **Allsopp M., Tirado R., Johnston P.** Plan bee living without pesticides [Report]: Greenpeace, 2014.

6. **Anoshin D., Shirokov D., Strok D.** Getting Started with Cloud Analytics, Jumpstart Snowflake [Book]: Apress, 2020.

7. **Apis** [Online] // www.apistech.eu. - 2015.

8. **Arnia** [Online] // www.arnia.com. - 2016.

9. **Atanasov J., Atanasova T.** Optimizing the management and control of apparel enterprise by information technologies [Conference]: TELFOR'2011 - IEEE, 2011.

10. **Atanasova T.** Methods for Processing of Heterogeneous Data in IoT Based Systems [Conference]: DCCN - Moscow, Springer, 2019, pp. 524-535.

11. **Atanasova T., Poryazov S., Saranova E.** Problems With Quality Enabling Of Information Functions Composition In Smart Buildings [Conference]: IEEE 24th Telecommunications Forum TELFOR 2016 - Belgrade, IEEE, 2016.

12. **Atwal H.** DataOps Technology. In: Practical DataOps [Book]: Apress, 2020.

13. **Azure.** Microsoft Azure Machine Learning Studio [Online]: 2020 - https://studio.azureml.net/.

14. **Bakanova N., Atanasova T.** Use Of Information Resources Of Organizational Systems In The Algorithms To Support Managerial Decisions 1 [Conference]: BdKCSE'2018 - Sofia, IEEE, 2018.

15. **Balabanov T., Atanasova T., Blagoev I.** Activation Function Permutation for Multilayer Perceptron Training [Conference]: International Conference on Big Data, Knowledge and Control Systems Engineering - Sofia, IEEE, 2018.

16. **Balabanov T., Ivanov S., Ketipov R.** Solving Combinatorial Puzzles with Parallel Evolutionary Algorithms [Conference]: Large-Scale Scientific Computing: 12th International Conference, LSSC 2019, Springer, 2020.

17. **Banafa A.** Three Major Challenges Facing IoT [Journal]: IEEE, 2017 - pp. 1-4.

18. **Barker S., Rothmuller M.** The Internet Of Things: Consumer, Industrial & Public Service 2020-2024 [Report]: Juniper Research, 2020.

19. **Barnkob M., Krukow J.** Event Sourcing and Command Query Responsibility Segregation Reliability Properties [Report]: ES and CQRS, 2018.

20. **Beecham Research.** Smart Farming: The sustainable way to food [Report]: Beecham Research, 2017.

21. **BeeTight** [Online] // www.beetight.com. - 2010.

22. **Belani G.** Programming Languages You Should Learn in 2020 [Journal]: IEEE Computer Society Magazines, 2020.

23. **Bell J.** Machine Learning Streaming with Kafka [Book Section]: Machine Learning: Hands-On for Developers and Technical Professionals, Wiley, 2020. - Vol. 12.

24. **Berger R.** Building longevity into an organizatioon's dna [Report]: Ronald Berger, 2019.

25. **Berger R.** Business opportunities in precision farming: Will big data feed the world in the future? [Report]: Roland Berger – Strategy Consultants GmbH, 2015.

26. **Bertelle C., Duhart C.** Toward Ogranic Computing Approach for Cybernetic Responsive Environment [Journal]: International Journal of Ambient Systems and Applications (IJASA) Vol.3, No.4, 2016. - 4 : Vol. 3.

27. **Blagus R., Lusa L.** SMOTE for high-dimensional class-imbalanced data [Conference]: BMC Bioinformatics'14 - Bioinformatics, 2013.

28. **Boiroju N., Yerukala R., Rao M., Malkareddy K.** A bootstrap test for equality of mean absolute errors [Journal]: ARPN Journal of Engineering and Applied Sciences, 2011 - 5 : Vol. 6.

29. **Bonaccorso G.** Machine Learning Algorithms [Book]: Packt, 2017.

30. **Borcard D., Gillet F., Legendre P.** Numerical Ecology with R [Book]: Springer, 2011.

31. **Braga A., Gomes D., Rogers R., Hassler E., Freits B., Cazier J.** A method for mining combined data from in-hive sensors, weather and apiary inspections to forecast the health status of honey bee colonies [Journal]: Elsevier, 2020 - Vol. 169.

32. **Bruinsma J.** World Agriculture: Towards 2015/2030 [Book]: London : Earthscan, 2017.

33. **BuzzTech** [Online] // www.buzztech.com. - 2017.

34. **Byrne C.** Development Workflows for Data Scientists [Book]: O'Reilly, 2017.

35. **Carter J., Grommon E., Harris Ph.** From conceptual to operational: Over-the-air-programming of land mobile radios [Book Section]: Physical Communication. Elsevier, 2016 - Vol. 19.

36. **Chawla V., Bowyer W., Hall O., Kegelmeyer Ph.** SMOTE: Synthetic Minority Over-sampling Technique [Journal]: Journal of Artificial Intelligence Research 16, 2002.

37. **Cohn J.** Scrum Mastery + Agile Leadership: The Essential and Definitive Guide to Scrum and Agile Project Management [Book]: Jeff Cohn, 2019.

38. **Consonni V., Todeschini R., Ballabio D., Grisoni F.** On the Misleading Use of for QSAR Model Comparison [Journal]: Wiley, 2019. - 102 : Vol. 38.

39. **Conte Y., Ellis M., Ritter W.** Varroa mites and honey bee health: can Varroa explain part of the colony losses? [Journal]: Journal of general virology Volume 41, 2010, pp. 353-363.

40. **Cortes C., Mohri M., Rostamiizadeh A.** L2 Regularization for Learning Kernels [Journal]: Cornell University, 2012.

41. **Criminisi A., Shotton J., Konukoglu E.** Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning [Journal]: Now Publishers, 2012. - Vol. 7.

42. **Croux Ch., Dehon C.** Influence functions of the Spearman and Kendall correlation measures [Journal]: Springer, 2010.

43. **Delaplane K., Mayer D., Mayer F.** Crop Pollination by Bees [Book]: CABI, 2000.

44. **Deol Sh.** Cross-layer design in the Internet of Things (IoT): issues and possible solutions [Report]: Canada : Carleton University, 2016.

45. **Dharmendra S.** Unsupervised, supervised and reinforced learning via spiking computation [Book]: DARPA, 2016.

46. **Dineva K., Atanasova T.** Methodology for Data Processing in Modular IoT System [Conference]: 22-st International Conference, DCCN 2019, Springer, 2019 pp. 457–468.

47. **Dineva K., Atanasova T.** Integrated Systems With Embedded Sensors for Digital Agriculture [Conference]: 19th International Multidisciplinary Scientific GeoConference SGEM, 2019

48. **Dineva K.**, **Atanasova T.** Regression Analysis on Data Received from Modular IoT System [Conference]: ESM'19, 2019, pp. 114-120

49. **Dineva K.**, **Atanasova T.,** Security in IoT Systems [Conference]: SGEM 2019, Vol. 19, Informatics, Geoinformatics and Remote Sensing, Issue 2.1, 2019, pp. 576-577

50. **Dineva K.**, **Atanasova T.,** ICT-based Beekeeping using IoT and Machine Learning [Conference]: 21-st International Conference, DCCN'18, Springer, 2018

51. **Dineva K.**, **Atanasova T.,** Applying Machine learning against beehives Dataset [Conference]: 18-th International Multidisciplinary Scientific Geoconference SGEM 2018, pp. 35-42.

52. **Dineva K.**, **Atanasova T.,** OSEMN process for working over data acquired by IoT devices mounted in beehives [Conference]: CTNS'18, vol. 7, Issue 13, 2018, pp. 47-53.

53. **Dineva K.**, **Atanasova T.,** Computer System Using Internet of Things for Monitoring of Bee Hives [Conference]: SGEM GeoConference, Vol. 17, Issue 63, 2017, pp. 169-176.

54. **Dineva K.**, **Atanasova T.,** Model of Modular IoT-based Bee-Keeping System [Conference]: European Simulation and Modelling Conference ESM'17, 2017, pp. 404-406.

55. **Dineva, K.,** Analytical review of existing computer systems for monitoring of beehives [Conference]: UNITECH'17, 2017, pp. 148-152.

56. **Dineva, K**., Internet of Things in Help of Sustainable Agricultural Development [Conference]:International Conference Automatics And Informatics'2017, 2017, pp.309-312.

57. **Dominguez S.** The Target Breach – Case Study, Lessons Learned and the Lockheed Martin Intrusion Kill Chain Model [Case]: IBM Lab Services, 2016.

58. **Eurostat.** Food Safety - European Commission [Report]: Eurostat, 2018.

59. **Filip I., Postoaca V., Stochitoiu R., Neatu D., Negru C., Pop F.** Data Capsule: Representation of Heterogeneous Data in Cloud-Edge Computing [Journal]: IEEE, 2019 - Vol. 7.

60. **Firouzi F., Farahani B., Weinberger M., DePace G., Aliee F.S.** IoT Fundamentals: Definitions, Architectures, Challenges, and Promises [Journal]: Springer, 2020.

61. **Fushiki T.** Estimation of prediction error by using K-fold cross-validation [Journal]: Springer, 2011.

62. **Gadge S., Kotwani V.** Microservice Architecture: API Gateway Considerations [Report]: GlobalLogic, 2017.

63. **Gallai N., Salles J.** Econimic valuation of the vulnerability of world agriculture confronted with pollinator decline [Journal]: Elsevier, 2009 - 3 : Vol. 68 - pp. 810-821.

64. **Georgieva P., Popchev I.** Fuzzy Logic Q-measure Model for Managing Financial Investments [Journal]: Proceeding of the Bulgarian Academy of Sciences, 2013 - Vol. 66.

65. **Gilchrist A.** Industry 4.0. The Industrial Internet of Things [Book]: Apress, 2016.

66. **Glushkva T., Stoyanov S., Popchev I.** Internet of Things Platform Supporting Mobility of Disabled Learners [Journal]: International Journal Bioautomation, 2019 - Vol. 23.

67. **Gökalp E., Sener U., Eren P.** Development of an Assessment Model for Industry 4.0: Industry 4.0-MM [Conference]: International Conference on Software Process Improvement and Capability Determination - 2017 - pp. 128 - 142.

68. **Guhr S., Martenson J., Laser H.** Data Science as a Service - Prototyping for an Enterprise Self-Service Platform for Reproducible Research [Conference]: IARIA - The Fifth International Conference on Fundamentals and Advances in Software Systems Integration - FASSI, 2019.

69. **Gulowaty B., Ksieniewicz P.** SMOTE Algorithm Variations in Balancing Data Streams [Conference]: Intelligent Data Engineering and Automated Learning – IDEAL 2019 - Springer, 2019 - Vol. 11872.

70. **Hackeling G.** Mastering Machine Learning with scikit-learn [Book]: PACKT, 2014.

71. **Hameed S., Khan F., Hameed B.** Understanding Security Requirements and Challenges in Internet of Things (IoT): A Review. [Journal]: Journal of Computer Networks and Communications, 2019.

72. **Hans C.** Bayesian lasso regression [Journal]: Biometrika, 2009. - 4 : Vol. 96.

73. **Harper R.** Inside the smart home [Book]: Springer, 2006.

74. **Henry A., Ridene Y.** Migrating to Microservices [Book Section]: Microservices - Science and Engineering - Springer, 2020.

75. **Hesse G., Matthies Ch., Rabl T., Uflacker M.** How Fast Can We Insert? A Performance Study of Apache Kafka [Journal]: Cornell University, 2020.

76. **HiveTracks** [Online] // www.hivetracks.com. - 2015.

77. **Howell K.** An introduction to the philosophy of methodology [Book]: SAGE, 2013.

78. **Janssen M., Luthra S., Mangla S., Rana N. and Dwivedi Y.** Challenges for adopting and implementing IoT in Smart cities: An integrated MICMAC-ISM approach [Journal]: Internet Research, 2019 - Vol. 29.

79. **Janssens J.** Data Science at the Command Line [Book]: O'Reilly Media, 2014.

80. **Japikse P., Grossnicklaus K., Dewey B.** Building the Spy Store Web Application with Angular [Book]: Berkeley, CA : Apress, 2020.

81. **Jazdi N.** Cyber physical systems in the context of Industry 4.0 [Journal]. - Romania : IEEE International Conference on Automation, Quality and Testing, Robotics, 2014.

82. **Jena S., Bhalja B.** A New Numeric Busbar Protection Scheme using Bayes Point Machine [Conference]: IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC) - IEEE, 2017.

83. **Jiang J.** Computers and Electronics in Agriculture [Journal]: Elsevier - Elsevier, 2016.

84. **Joshi V.** Azure Machine Learning. In: Machine Learning and Artificial Intelligence [Book Section]: Machine Learning and Artificial Intelligence - Springer, 2020.

85. **Kantardzic M.** Data Mining: Concepts, Models, Methods, and Algorithms, 2nd Edition [Book]: Wiley-IEEE Press, 2011.

86. **Kaspi O., Yosipof A., Senderowitz H.** Random Sample Consensus (RANSAC) algorithm for material informatics: application to photovoltaic solar cells [Journal]: Springer, 2017.

87. **Kolchakov K., Monov V.** Query Conflicts In An Algoorithm For NoN Conflict Scheduling For Crossbar Commutator [Conference]: International Conference Automatics and Informatics`2018 - Sofia : SAI, 2018.

88. **Kost S., Rheinbach O., Schaeben H.** Logistic Regression for Prospectivity Modeling [Journal]: TU Bergakademie Freiberg, 2020.

89. **Kotler Ph.** Philip Kotler's Contributions to Marketing Theory and Price [Journal]: Emerald Group Publishing limited - 2011 - pp. 87-120.

90. **Kotlyar M., Fuhrman S., Ableson A.** Spearman Correlation Identifies Statistically Significant Gene Expression Clusters in Spinal Cord Development and Injury [Journal]: Neurochem, 2002.

91. **Kovaleva M.** Best of JetBrains Toolbox 2019.3 Release - JetBrains Blog, 2019.

92. **Kranenburg R., Bassi A.** IoT Challenges. Communications in Mobile Computing [Journal]: Springer open Journal, 2012.

93. **Krupke Christian H., Hunt, Greg J., Eitzer, Brian D.** Multiple Routes of pesticide exposure for honey bees living near agricultural fields [Journal]: Plos one - 2012.

94. **Li T., Somers J., Xiagiong J. Hu, McCandless L.** Bayesian Sensitivity Analysis for Non-ignorable Missing Data in Longitudinal Studies [Journal]: Springer, 2019.

95. **Liakos G., Busato P., Moshou D., Pearson S., Bochtis D.** Machine Learning in Agriculture: A Review. [Journal]: Sensors, Volume 18, Issue 8 - 2018.

96. **Ling C., Huang J.** Using AUC and Accuracy in evaluating learning algorithms [Journal]: IEEE, 2005 - 3 : Vol. 17.

97. **Liu Y., Liao Sh., Jiang Sh., Lin H., Wang W.** Fast Cross-Validation for Kernel-Based Algorithms [Journal]: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020 - 5 : Vol. 42.

98. **Locht A.** From Operational Data to Trusted Big Data [Report]: IBM Solutions Connect, 2013.

99. **Lunga D., Gerrand J., Yang L., Layton C. and Stewart R.** Apache Spark Accelerated Deep Learning Inference for Large Scale Satellite Image Analytics [Journal]: EEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2020 - Vol. 13.

100. **Luo Q., Wang H., Shang G.** College Students Learning Behavior Analysis Based on SVM and Fisher-Score Feature Selection [Conference]: CSPS 2019: Communications, Signal Processing, and Systems - Springer, 2020 - Vol. 571.

101. **Manel D., Fraser S.** Agile Processes in Software Engineering and Extreme Programming – Workshops [Conference]: 20th International Conference on Agile Software Development. - Montréal, Springer, 2019.

102. **McKinney W.** Python for data analysis: Data wrangling with Pandas, NumPy, and IPython [Book]: O'REILLY, 2012.

103. **Mease D., Wyner A. J. and Buja A.** Boosted Classification Trees and Class Probability Estimation [Journal]: JMLR, 2007.

104. **Meghanadh B., Aravalath, L., Joshi B. et al.** Imputation of Missing Values in the Fundamental Data: Using MICE Framework [Journal]: Springer, 2019.

105. **Meinke K., Nycander P.** Learning-Based Testing of Distributed Microservice Architectures: Correctness and Fault Injection [Conference]: SEFM 2015: Software Engineering and Formal Methods. - Berlin : Springer, 2015 - Vol. 9509.

106. **Mengyu X., Zhang D.** Pearson's chi-squared statistics: approximation theory and beyond [Journal]: Biometrika, 2019 - 3 : Vol. 106.

107. **Mezghani E., Expósito E., Drira K.** A Model-Driven Methodology for the Design of Autonomic and Cognitive IoT-Based Systems: Application to Healthcare.[Conference]: IEEE Transactions on Emerging Topics in Computational Intelligence, 2017 - Vol. 1.

108. **Microsoft Documentation** [Online] - 2016 - https://docs.microsoft.com/en-us/ef/core/.

109. **Microsoft.** Understanding TCP/IP addressing and subnetting basics [Online]: Microsoft Supprot - 20 Dec 2019 - https://support.microsoft.com/en-us/help/164015/understanding-tcp-ip-addressing-and-subnetting-basics.

110. **Minka T.** Expectation Propagation for approximate Bayesian inference [Journal]: Uncertainty in Artificial intelligence, 2001.

111. **Miu A., Ferreira F., Yoshida N., Zhou F.** Generating Interactive Web Socket Applications in TypeScript [Conference]: PLACES 2020 - 2020.

112. **Mojgan G.** Learning from imbalanced and heterogeneous data [Report]: UNSW Australia, 2017.

113. **Molnar Ch.** A Guide for Making Black Box Models Explainable. Chapter 4 [Book Section]: Interpretable Machine Learning - ChristophMolnar, 2020.

114. **Moon J., Shine Y.** A Study of Distributed SDN Controller Based on Apache Kafka [Conference] : IEEE International Conference on Big Data and Smart Computing (BigComp) - Busan, Korea : IEEE, 2020.

115. **Morville P., Rosenfeld L.** Information Architecture for World Wide Web [Book]: O'Reilly, 2006.

116. **Mostafavi S., Dawlatnazer M., Paydar F.** Edge Computing for IoT: Challenges and Solutions [Journal]: Journal of Communications Technology, Electronics and Computer Science, 2019, pp. 26.

117. **Naik N.** Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP [Conference]: IEEE International Systems Engineering Symposium (ISSE'17) - IEEE, 2017.

118. **Naseem I., Togneri R., Bennamoun M.** Linear Regression for Face Recognition [Journal]: IEEE, 2010 - 11 : Vol. 32.

119. **Neumann P., Carreck N.** Honey bee colony losses [Journal]: Journal of Apicultural Research, 2015.

120. **Newton P., Bristoll H.** SWOT Analysis – Strategy Skills [Journal]: Team FME, 2013.

121. **Nosowitz D.** Honeybee Deaths Getting Worse: We Lost 44 percents of Colonies Last Year [Journal]: Journal of Modern Farmer, 2016.

122. **Oleszak M.** Regularization: Ridge, Lasso and Elastic Net [Journal]: Data Camp Tutorials, 2019.

123. **Ozawa M.** Soundness and completeness of quantum root-mean-square errors [Journal]: NPJ Quantum Information, 2019.

124. **Peter C.** Saving Bees With The Internet Of Things [Report]: Forbes, 2016.

125. **Pilloni V.** How data will transform industrial processes: crowdensing, crowdsourcing and big data as pillars of industry 4.0 [Journal]: Future Internet, 2018 - 24 : Vol. 10.

126. **Popchev I., Orozova D., Stoyanov S.,** IoT and Big Data Analytics in E-Learning [Conference]: Big Data, Knowledge and Control Systems Engineering - BdKCSE'2019 - Sofia : IEEE, 2019.

127. **Poyarkov A., Drutsa A., Khalyavin A., Gusev G., Serdykov P.** Boosted Decision Tree Regression Adjustment for Variance Reduction in Online Controlled Experiments [Conference]: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - ACD, 2016.

128. **Pozza M., Nicholson P., Lugones D.** On Reconfiguring 5G Network Slices [Journal]: IEEE Journal on Selected Areas in Communications, 2020.

129. **R. Hertzog, R. Mas** The Debian administrator's handbook: Debian Jessie from Discovery to Mastery [Book]: Freexian, 2015.

130. **Ramamuthy A., Jain P.** The Internet of Things in the Power Sector Opportunities in Asia and Pacific [Book Section]: Sustainable Development Working Paper Series - Asian Development Bank, 2017 - Vol. 48.

131. **Richard W., ArnoldThomas P., BishopOlushola O., EshoJordan R.** Utilization of a concept to obtain data of specific interest to a user from one or more data storage locations [Patent] - US10445310B2, 2020.

132. **Rogozhnikov A.** Reweighting with Boosted Decision Trees [Journal]: Journal of Physics Conference Series 762(1), 2016.

133. **Rustam Z., Rampisela T.** Support vector machines and twin support vector machines for classification of schizophrenia data [Journal]: Journal of Engineering & Technology, 2018.

134. **Saif S., Garba A., Awwalu J., Arshad H., Zakaria L.** Performance Comparison of Min-Max Normalisation on Frontal Face Detection Using Haar Classifiers [Journal]: PERTANIKA, 2017.

135. **Samariya D., Aryal S., Ting K.** A new effective and efficient measure for outlying aspect mining [Journal]: Cornell University, 2020.

136. **Samuels P., Gilchrist M.** Pearson Correlation [Report]: Birmingham City University, 2014.

137. **Saraswathi R.** Four Architecture Choices for Application Development in the Digital Age [Journal]: IBM Cloud, 2020.

138. **Scales Beehive** [Online] - 2012. - www.beehive-scales.co.uk.

139. **Schnaubelt M., Fischer T., Krauss Ch.** Separating the signal from the noise – financial machine learning for Twitter [Journal]: Journal of Economic Dynamics and Control, 2020.

140. **Sengupta U., Rasmussen C., Juniper M.** Bayesian Machine Learning For The Prognosis Of Combustioninstabilities From Noise [Conference]: ASME 2020 Turbomachinery Technical Conference Exposition - ASME, 2020.

141. **Sharma A., Sharma R.** A Review of Applicatioons, Approaches, and Challenges in Internet of Things (IoT) [Conference]: ICRIC - Springer, 2020.

142. **Shekaramiz M., Moon T.** A Note on Bayesian Linear Regression [Report]: Electrical and ComputerEngineering Faculty Publications, 2018.

143. **Singh H. Lone Y.** Introduction to Machine Learning [Article]: Deep Neuro-Fuzzy Systems with Python : Springer, 2019.

144. **Singh J., Saravanan B., Prased V.** [Patent] : 16/147, 976. - US, 2019.

145. **Sorrell S.** IOT- The Internet of Things Transformation 2018 [Report]: Juniper Research – Whitepaper, 2018.

146. **Sorrell S.** The Internet Of Things: Consumer, Industrial & Public Service 2018-2023 [Report]: Juniper Research, 2018.

147. **Sotomayor J., Allala S., Alt P., Phillips J., King T., Clarke P.** Comparison of Runtime Testing Toools for Microservices [Conference]: IEEE 43rd COMPSAC - IEEE, 2019.

148. **Sowmya J., Shetty Ch.** IoT and Data Analytics Solution for Smart Agriculture [Book Section]: The Rise Fog Computing in the Digital Erap - IGI Global, 2019. - Vol. 9.

149. **Steinhauer N., Kulhanek, K., Antunez, K.** Drivers of colony losses [Journal]: IEEE, 16 May 2018.

150. **Suchetha K., Guruprasad H.** Integration of IOT, Cloud and Big Data [Journal]: Global Journal of Engineering Science and Researches, 2015 - 7 : Vol. 2.

151. **Syngenta.** Financial Report 2016 [Report]: Syngenta, 2017.

152. **Talari S., Shafie-khah M., Siano P., Loia V., Tommasetti A., Castalao J** A Review of Cities Based on the Internet of Things Concept [Journal]: MDPI - 2017. - p. 24.

153. **Tashev T., Monov V.** Computer simulations of a modified MiMa-algorithm for a crossbar packet switch [Conference]: CompSysTech ' 14 - Ruse : ACM, 2014 - Vol. 883.

154. **Tashev T., Monov V.** Large-Scale Simulation of Uniform Load Traffic for Modeling of Throughput on a Crossbar Switch Node [Conference]: 8th International Conference LSSC 2011 - Sozopol : Springer, 2011 - Vol. 7116.

155. **Technologies Bee Smart** [Online] // www.beesmarttechnologies.com - 2016.

156. **Tewari S.** Ensemble Methods: An intelligent Modeling Approach [Conference] // EAGE Annual 2020 - EAGE, 2020.

157. **Thoo E., Heudecker N., Zaidi E.** Critiical Capabilities for Data Integration Tools [Report]: Gartner Research, 2019.

158. **Tomov P., Zankinski I., Balabanov T.** Genetic Algorithm Selection Operator Based on Recursion and Brute-Force [Conference]: 14th Annual Meeting of the Bulgarian Section of SIAM, BGSIAM'19 - Sofia : SIAM, 2019.

159. **Tseitlin A., Sondow J.** Progressive deployment and termination of canary instances for software analysis [Patent] - United States US9712411B2, 2015.

160. **Ullah M., Pronobis A., Caputo B., Luo J., Jensfelt P.** The COLD Database [Report] - Martigny, Switzerland : IDIAP Research Institute, 2008.

161. **Valdés J.** Extreme learning machines with heterogeneous data type [Journal]: Neurocomputing, 2018. - Vol. 277.

162. **Vanwinkelen G., Blockeel H.** On estimating Model Accuracy with repeated Cross-Validation [Conference]: Belgian-Dutch Conference on Machine Learning - 2012.

163. **Varga E., Draskovic D., Mijic D.** Scalable Architecture for the Internet of Things [Book]: O'Reily, 2018.

164. **Vasanthi R., Kannan V.** Machine Learning Algorithms with ROC Curve for Predicting and Diagnosing the Heart Disease [Book Section]: Soft Computing and Medical Bioinformatics - Springer, 2019.

165. **Verboven H., Uyttenbroeck R., Brys R., Hermy M.** Different responses of bees and hoverflies to land use in an urban-rural gradient show the importance of the nature of the rural land use [Journal]: Landscape and Urban Planning Volume 126 - 2014 - pp. 31-41.

166. **Waller M., Stradling J., Lotz M., Sikkink J.** Microservice architecture workflow management mechanism [Patent] - US Patent US10235105B1, 2019.

167. **Waltenburg E., McLauchlan W.** An Introduction to Exploratory Data Analysis [Book Section]: Exploratory Data Analysis: A primer for undergraduates - Purdue e-Pubs, 2012.

168. **Wang L.** Heterogeneous Data and Big Data Analytics [Journal]: Automatic Control and Information Sciences Volume 3 Issue 1 - 2017 - pp. 8-15.

169. **Wittek K.** Integration Testing with Docker and Testcontainers [Conference]: DevoxxUK - DevoxxUK, 2019.

170. **Yaghini M.** Data Understanding and preparation [Journal]: Springer, 2010.

171. **Yang B., Sailer A., Jain S., Tomala-Reyes A., Singh M., Ramnath A.** Service Discovery based Blue-Green Deployment Technique in Cloud Native Environments [Conference]: IEEE SCC - IEEE, 2018.

172. **Yang J., Rahardja S., Fränti** Outlier detection: how to threshold outlier scores? [Conference]: AIIPCC'19 - ACM, 2019.

173. **Zacepins A., Brusbardis V., Meitalovs J., Stalidzans E.** Challenges in the development of Precision Beekeeping [Journal]:Elsevier - 2015 - pp. 60-71.

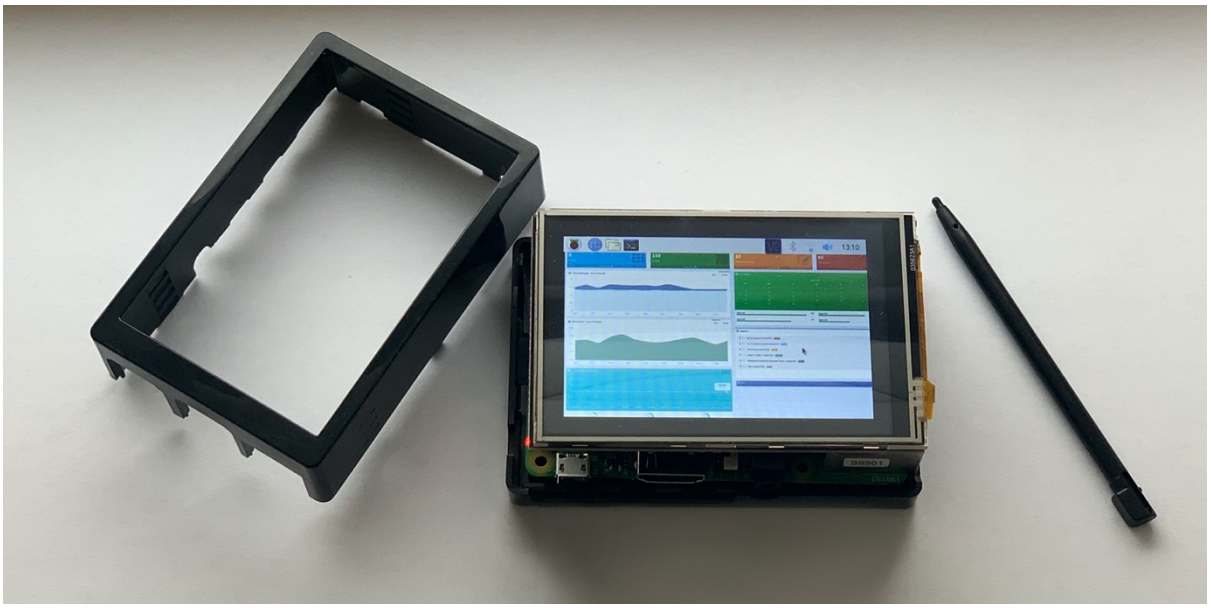
174. **Zhou Z.** Ensemble Methods: Foundation and Algorithms [Book]: Chapman & Hall/CRC, 2012.

175. **Zikria B., Kim S., Hahm O., Afzal M. and Aalsalem M.** Internet of Things (IoT) Operating Systems Management: Opportunities, Challenges, and Solution [Journal]: Sensors, 2019. - Vol. 8.

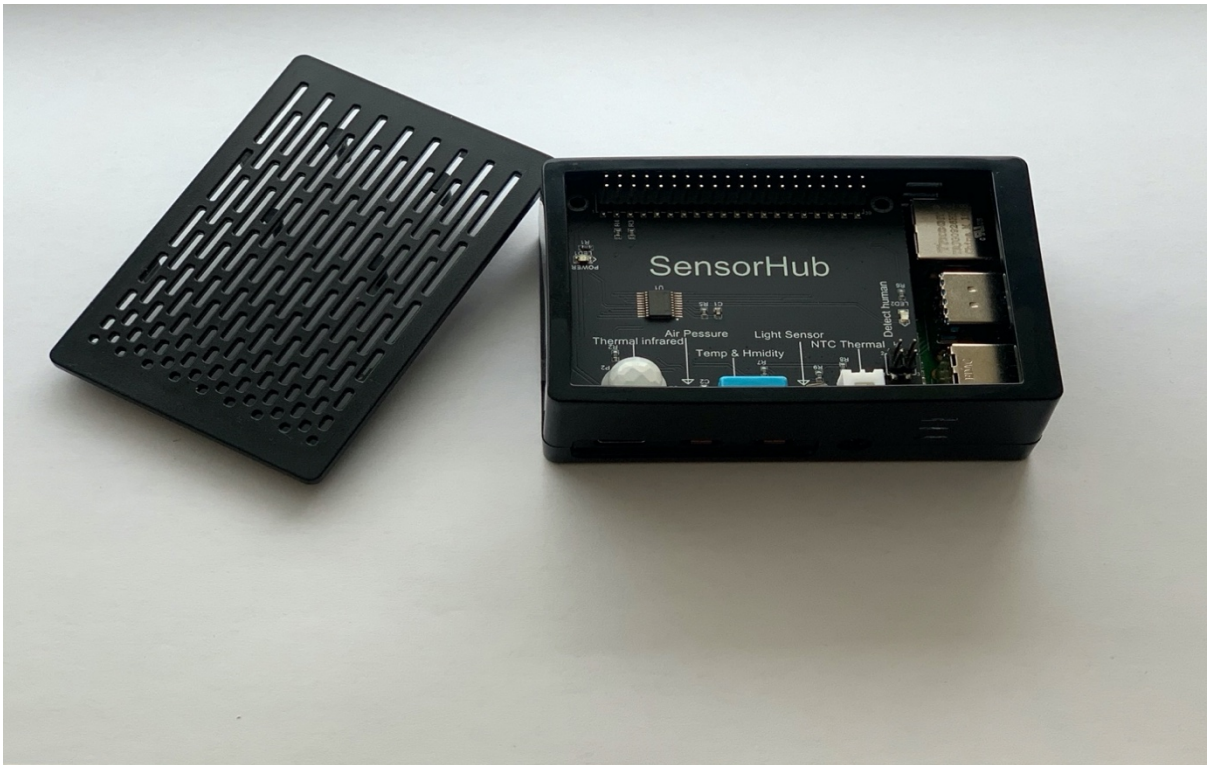
Оформление: GOST стандарт

Приложения

Приложение 1: Прототипи на IoT устройства, разработени според предложената в този дисертационен труд конфигурация.



Снимка 1: IoT Edge

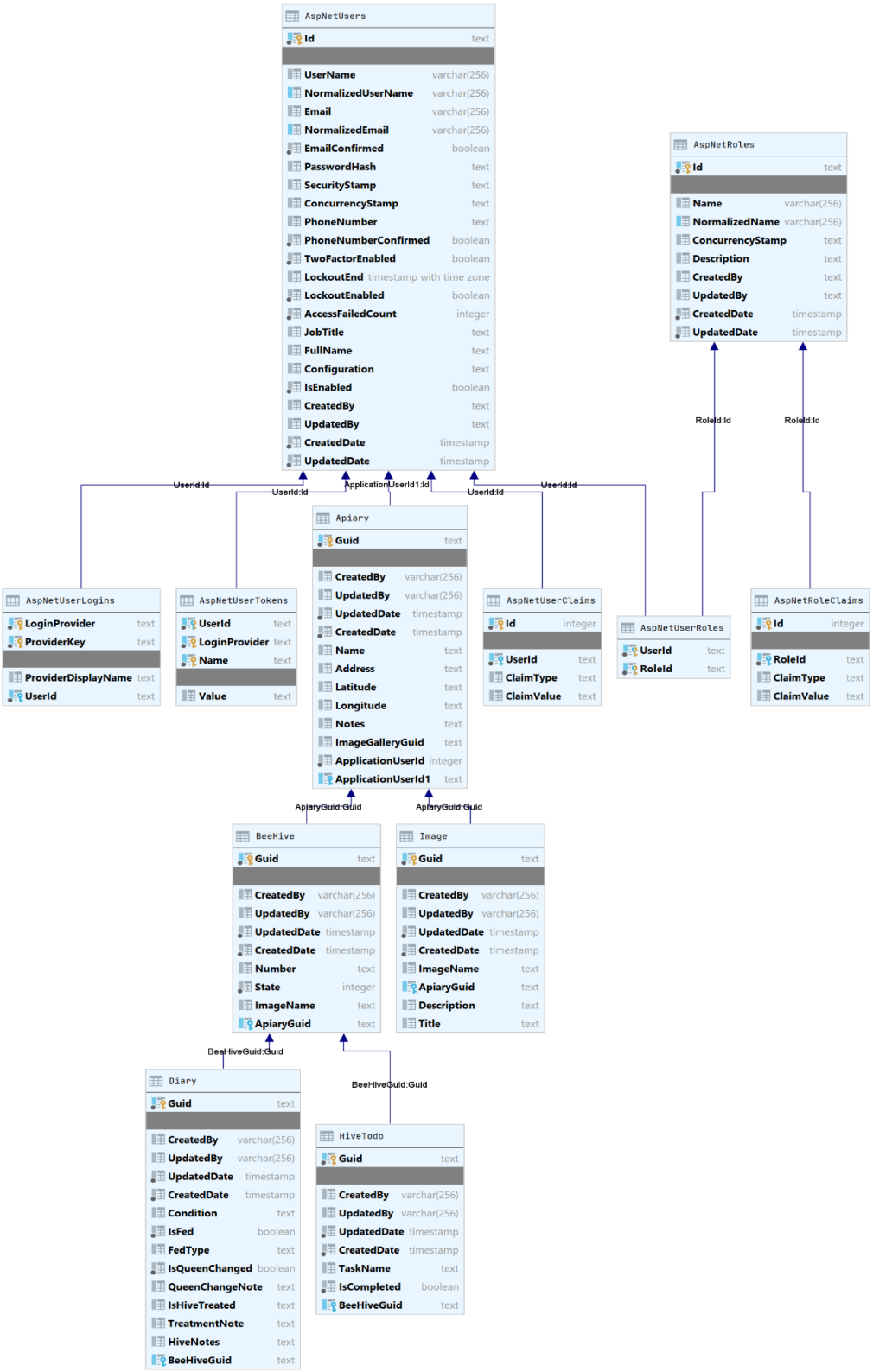


Снимка 2: IoT Unit

Приложение 2: ZigBee комуникация, предложена в дисертацията за връзка между IoT устройствата.

```
#!/usr/bin/env python3
#
# Acts as a receiver of data from a sending XBee module, and controls
#
# **NOTE: REQUIRES PYTHON 36+**

# import libraries
import serial, time
import RPi.GPIO as GPIO
from xbee import XBee

# assign the GLOBAL VARS and XBee device settings
SERIAL_PORT = "/dev/ttyS0"
BAUD_RATE = 9600

# set the pin mode
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# initialize GPIO outputs needed
# do something

# configure the xbee and enable asynchronous mode
ser = serial.Serial(SERIAL_PORT, baudrate=BAUD_RATE)
xbee = XBee(ser, callback=receive_data, escaped=False)


# handler for whenever data is received from transmitters - operates asynchronously
def receive_data(data):
    print("Received data: {}".format(data))
    rx = data['rf_data'].decode('utf-8')
    state, command = rx[:1], rx[1:]

    # parse the received contents and activate the respective GPIO
    # if the data received is actionable
    if state in ("0", "1"):
        if [some logic here]:
            # do work
        else:
            print("ERROR: Received invalid command '{}' - ignoring transmission".format(state))
    else:
        print("ERROR: Received invalid STATE '{}' - ignoring transmission".format(state))

    print("Packet: {}".format(data))
    print("Data: {}".format(data['rf_data']))


if __name__ == '__main__':

    # main loop/functionality
    while True:
        try:
            # operate in async mode where all messages will go to handler
            time.sleep(0.001)
        except KeyboardInterrupt:
            break

    # clean up
    GPIO.cleanup()
    xbee.halt()
    ser.close()
```

Приложение 3: Схема на релационна база данни за Identity Service (разширена) в разработената в дисертацията MSA архитектурна рамка на сървърната архитектура.



Приложение 4 А: Програмен код за конфигуриране на модела за класификационен анализ (Python).

```
"""Classification"""

import multiprocessing
import pickle

import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import cross_validate, RepeatedStratifiedKFold
from xgboost.sklearn import XGBClassifier

import xgboost as xgb

if __name__ == '__main__':
    # 1. read and load data
    df = pd.read_csv('beedset.csv', index_col=0)

    # 2. clean missing data
    df.dropna()

    # 3 SMOTE
    # helper visuals
    cmap = mpl.colors.ListedColormap(['#0fb099', 'black'])
    plt.scatter(df.humidity, df.t_in, s=25, c=df.observation, cmap=cmap).figure.show()
    df.observation.value_counts()

    # smote
    smt = SMOTE(sampling_strategy={0: 8529, 1: 5148}, random_state=None, k_neighbors=1, n_jobs=None)
    df_smote = smt.fit_sample(df, df.observation)
    df = df_smote[0].copy()

    # helper visuals
    plt.scatter(df.humidity, df.t_in, s=25, c=df.observation, cmap=cmap).figure.show()
    df.observation.value_counts()

    # 4. normalize df min/max
    df = (df - df.min()) / (df.max() - df.min())

    # 5. feature selection
    X = df.iloc[:, :3]
    y = df.iloc[:, 3:4]

    # 6. prepare model for predictions
    model = XGBClassifier(num_boost_round=100, booster='gbtree', learning_rate=0.01, max_depth=25,
                          subsample=0.5, reg_alpha=390, gamma=400, reg_lambda=0.5, min_child_weight=205,
                          scale_pos_weigh=8529/5148)
    metrics = ['accuracy', 'precision', 'recall', 'f1', 'roc_auc']
    kfold_repeated = RepeatedStratifiedKFold(n_splits=10, n_repeats=10, random_state=123)

    # 6. cross-validation and prediction
    jobs_count = multiprocessing.cpu_count() * 2
    predictions = cross_validate(model, X, y, cv=kfold_repeated, n_jobs=jobs_count,
                                 scoring=metrics, return_train_score=True, return_estimator=True)

    # 7. format scores
    scores = {k.split('test_')[1]: v.mean() * 100 for (k, v) in predictions.items()
              if 'test_' in k and k.split('test_')[1] in metrics}

    # 8. serialize model for easy deployment
    model_dump = dict(model=model, metrics=metrics, kfold_repeated=kfold_repeated)
    with open('classification_model_dump.pickle', 'wb') as handle:
        pickle.dump(model_dump, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

Приложение 4 Б: Програмен код за конфигуриране на модела за регресионен анализ (Python).

```
"""Linear regression"""

import pickle
from math import sqrt

import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from sklearn.linear_model import ElasticNetCV
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score, mean_absolute_error, median_absolute_error
from sklearn.model_selection import train_test_split

if __name__ == '__main__':
    # 1. read and load data
    df = pd.read_csv('hive_w.csv', index_col=0, parse_dates=[['Month', 'Hour']])

    # 2. prepare data - deal with missing values, shape, transform
    df.dropna()
    df.drop('No.', axis=1, inplace=True)
    df.rename(columns={'Hive': 'HoneyQuantity', 'hPa': 'AtmospherePressure'}, inplace=True)
    df['Cloudy'] = df['Cloudy'].apply(lambda x: eval(x))

    # control
    df.shape

    # plot weights
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.heatmap(df.corr(method='pearson'), annot=True)
    fig.show()

    # helper visuals
    cmap = mpl.colors.ListedColormap(['#0fb099', 'black'])
    plt.scatter(df.Humidity, df.Temperature, s=25, c=df.HoneyQuantity, cmap=cmap).figure.show()
    plt.scatter(df.Humidity, df.AtmospherePressure, s=25, c=df.HoneyQuantity, cmap=cmap).figure.show()
    plt.scatter(df.AtmospherePressure, df.Temperature, s=25, c=df.HoneyQuantity, cmap=cmap).figure.show()

    # 3. shuffle the df
    df = df.sample(frac=1).reset_index(drop=True)

    # 4 .feature selection
    X = df.iloc[:, :5]
    y = df.iloc[:, 5:6]

    # 5. Split, train, fit
    x_train, x_test, y_train, y_test = train_test_split(X, y, random_state=1, test_size=0.3)
    linear_model_cv = ElasticNetCV(cv=10, normalize=True, eps=0.0001, n_alphas=100, l1_ratio=0.5, max_iter=1000,
                                   alphas=None, fit_intercept=True, precompute=False, tol=0.0000001, copy_X=True,
                                   n_jobs=-1, positive=False, random_state=0, selection='cyclic')
    linear_model_cv.fit(X=x_train, y=y_train)

    # 6. Predict
    y_pred = linear_model_cv.predict(X=x_test)

    # 8. report
    print(f'mean absolute error: {mean_absolute_error(y_true=y_test, y_pred=y_pred)}')
    print(f'root mean squared error: {sqrt(mean_squared_error(y_true=y_test, y_pred=y_pred))}')
    print(f'median absolute error: {median_absolute_error(y_true=y_test, y_pred=y_pred)}')
    print(f'coefficient of determintion: {r2_score(y_true=y_test, y_pred=y_pred)}')

    # 9. serialize model for easy deployment
    model_dump = dict(model=linear_model_cv)
    with open('linear_regression_cv_model_dump.pickle', 'wb') as handle:
        pickle.dump(model_dump, handle, protocol=pickle.HIGHEST_PROTOCOL)
```