Міністерство освіти і науки України
Одеський національний політехнічний університет

Інститут штучного інтелекту і робототехніки
Кафедра «Комп'ютерні системи»

Пшегалінський Олег Вікторович,
студент групи АК-151

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Дослідження методів підвищення ефективності системи розпізнавання типів руху людини

Напрям підготовки: 123 – "Комп'ютерна інженерія"

Спеціалізація: Спеціалізовані комп'ютерні системи

Керівник:
Стрельцов Олег Васильович,
канд. техн. наук, доцент

Одеса – 2020

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ОС – операційна система

СКБД – системи керування базою даних

СУБД – система управління базою даних

ІС – інформаційна система

UI – user interface (інтерфейс користувача)

UX – user expirience (досвід користування)

REST – representational state transfer (передача репрезентативного стану)

API – application programming interface (прикладний програмний інтерфейс)

KNN – k-nearest neighbors algorithm (метод найближчих k-сусідів)

NB – naive Bayes (наївний баєсів класифікатор)

DT – decision tree (дерево ухвалення рішень)

RF – random forest (випадковий ліс)

ВСТУП

Можливість точного визначення типів рухів людини має важливе значення в багатьох сферах діяльності. Моніторинг рухів людини широко використовується у спорті та медицині, віртуальній реальності, дистанційному управлінні, гібридних (людини сумісно із роботом) робочих місцях і т. ін. Для виконання цієї функції людині потрібні різні мобільні прилади, які здатні визначити тип її руху за допомогою датчиків та систем обробки інформації. Ця функція може бути частково реалізована за допомогою, сучасного смартфона, або фітнес-трекера. Помилки в таких системах не мають великого значення. Але існують ситуації, в яких точність визначення типу руху є надважливою. Особливо це стосується систем дистанційного управління роботами за допомогою рухів, або визначення стану людини під час активного руху.

Для того щоб поліпшити робочий процес пристрою, важливо розуміти який рух в даний момент робить людина. Точність визначення рухів безпосередньо відповідає за якість робочого процесу пристрою. Якщо рухи будуть визначатися невірно, пристрій не зможет отримати правильні команди що треба робити. В даному випадку основною проблемою яка постає перед розробниками - це написання рішення для високоточного визначення рухів. На даний момент існує вже ряд готових методик, що дозволяють визначати людські активності, але серед них нема тієї, яка б змогла задовольнити одночасно такими запитами:

- точність на рівні 92%;
- висока швидкість визначення;

Всі існуючі рішення не задовольняють одночасно цим вимогам, що не дозволяє рекомендувати їх для використання в сучасних мобільних додатках та пристроях.

Об'єктом дослідження є процес визначення людських рухів за допомогою мобільного пристрою.

Предметом дослідження є методи визначення людських рухів за допомогою мобільного пристрою.

Мета дослідження – розробка методики для підвищення точності визначення рухів в порівнянні з вже існуючими рішеннями, реалізація алгоритму системи визначення типів руху.

Завданнями дослідження є:

– аналітичний огляд сучасних досліджень з проблеми визначення типів руху людини;

– розробка методики визначення типів руху людини за допомогою мобільного пристрою;

– розробка алгоритму для визначення типів руху людини;

– аналіз результатів моделювання та експериментальної перевірки.

Наукова новизна роботи полягає у удосконаленні методики визначення типів руху людини та подальшому розвитку існуюючих методів визначення типів руху людини.

Практична цінність роботи полягає у можливості використання результатів роботи для створення систем у яких потрібно визначення людських рухів.

Дана работа спрямована на удосконалення поточного процесу розпізнавання рухів. Завдяки створеній методиці можливо поліпшити існуючий процес розпізнавання, оптимізувати використовувані ресурси.

Публікації. За результатами дослідження підготовлена наукова стаття для публікації у фаховому науковому виданні.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Огляд наукових досліджень з визначення рухів людини

Більшість наукових робіт описує проблему розпізнавання руху за допомогою мобільних пристроїв як частково вирішену, що дозволяє використовувати даний підхід в невеликих додатках, де не потрібна висока точність визначення і висока швидкість роботи. Така ж ситуація з процесом, пов'язаним з розпізнаванням людських облич за допомогою камер. Компанії як Apple, Google чи Span Inc. змогли досягти величезного прогресу в цій сфері, але при цьому не змогли зробити цю технологію масовою, доступною для кожного розробника, так як складність розпізнавання, складність використовуваних алгоритмів, а також споживані ресурси не дають можливість широко використовувати її. У розпізнаванні людських рухів так само присутній ряд проблем, для вирішення яких продовжуються дослідження шляхів поліпшення поточних методик використовуваних в цій сфері [1].

Одною із найсуттєвіших є проблема положення пристрою під час захоплення руху [2]. Велике значення має положення пристрою, то, як його тримає людина, де вона його несе в момент захоплення рухів (в сумці, кишені, руці). Практично єдиним пристроєм завдяки якому можна проводити досить точні захоплення рухів є акселерометр (рис 1.1). На поточний 2020 рік близько 99% випущених телефонів містять в собі акселерометр (за винятком бюджетних моделей). Акселерометр дає дуже точні дані, але розуміння про те, в якому становищі знаходиться пристрій повинен складати розробник програми виходячи з аналізу даних. Акселерометр буде давати різні дані навіть при одному і тому ж типі руху (наприклад ходьбі) якщо людина тримає пристрій в руці, і якщо вона несе його в кишені так як змінюється при цьому характер руху пристрою (рис 1.1).

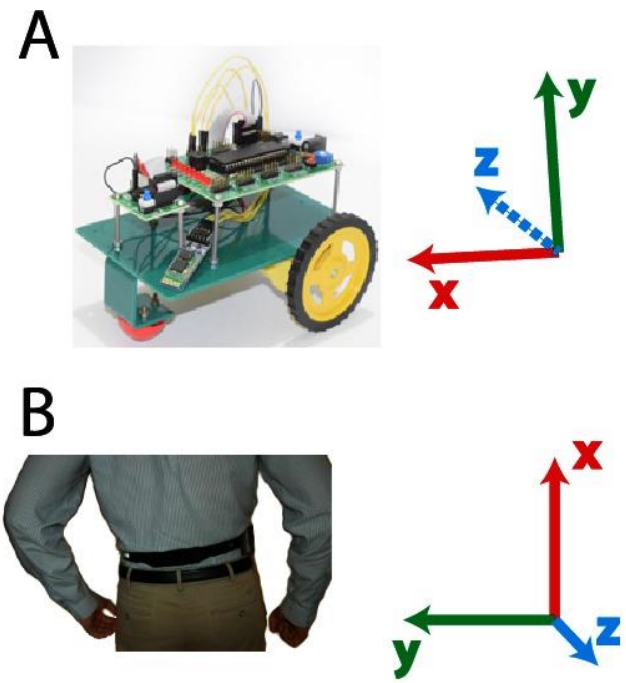

Рисунок 1.1 – Акселерометр в мобільному пристрої

Це має значення при розпізнаванні складних рухів: гра з дитиною або вихованцем, прибирання приміщення. Автор наукової статті робить припущення, що вирішити дану проблему можна тільки використовуючи останні напрацювання в області нейронних мереж і використовуючи величезні обсяги тренувальних даних (понад 1 млн записів). Для того щоб записати і проаналізувати такий обсяг даних потрібно багато ентузіастів з різними типами пристроїв.

Використання додаткових сенсорів також допомагає підвищити якість розпізнавання. Наприклад розбір траєкторії GPS в сукупності з акселерометром може покращувати результати приблизно на 25 відсотків в залежності від типу руху.

Автор також зауважує, що в деяких моделях телефонів (на операційній системі Android) акселерометр працює не зовсім коректно. Дані з даного сенсора проходять неточні і вирішити проблему можна калібрацією.

Але користувачі не будуть вручну калібрувати свої акселерометри, і це призводить до того що на частини пристроїв рух розпізнається некоректно. Іноді пристрої починають посилати невірні дані з координатами і вирішити дану

проблему можна тільки перезавантаженням пристрою, тому важливо постійно перевіряти дані на їх якість.

В наступній науковій роботі підіймається Автор розкрив цю проблему і дійшов до висновку, що на відміну від поширеної методики коли користувацькі рухи розпізнаються на класифікаторі, заздалегідь навченому за допомогою спеціальних тренувальних даних, що збираються волонтерами, – повинно бути підстроювання під конкретну людину кожен раз в процесі розпізнавання (рис 1.2).

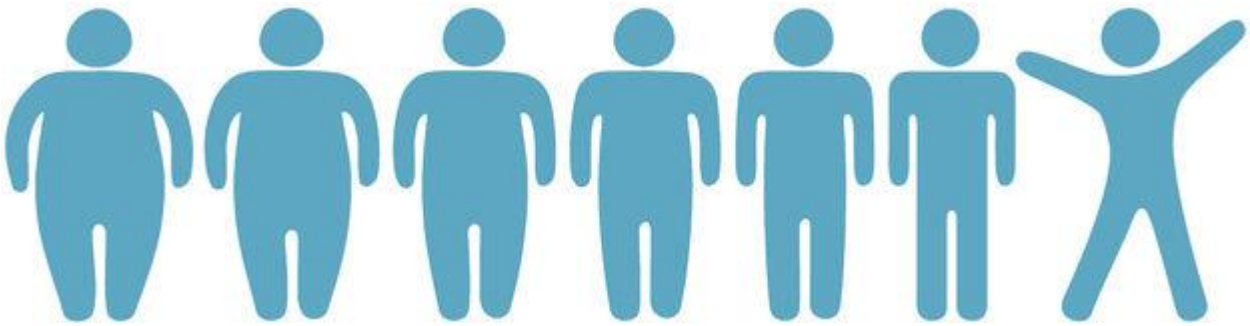

Рисунок 1.2 – Фізичні типи людей

Тобто повинен бути каркас, класифікатор, який розпізнає рухи для всіх людей однаково орієнтуючись на середню вибірку. Але в процесі користування цей класифікатор продовжує тренуватися, наприклад: обчислюється середня швидкість людини, на підставі даної швидкості можна визначити такий тип руху як ходьба. Значні відхилення в більшу сторону можна буде визначати як біг. У той же час для людей похилого віку їх біг буде вважатися за середню швидкість, і для них класифікатор знижує свої швидкісні параметри.

Подібний підхід активно використовується у мобільних голосових помічників, наприклад Сірі. Чим більше користувач спілкується з Сірі, тим краще вона починає передбачати його бажання та інтереси. Розробники стверджують, що кожен користувач має свою унікальну Сірі.

Але автори роботи хоч і відзначають, що такий метод зміг би значно підвищити якість розпізнання людських рухів, він так само вимагає і значних ресурсів від мобільного телефону, особливо енергоспоживання. Щоб обійтися без такої методики необхідно обробити величезну кількість даних, просити

користувача вводити свій зріст, вагу і вік на початку роботи з програмою, щоб виходячи з його особистих даних використовувати класифікатор, який найкраще підходить до нього. Автори вкінці відзначають, що не тільки зріст і вік мають значення, варто так само враховувати при підборі спеціального класифікатора і місце проживання людини, і його рід діяльності (няні або вихователькі незалежно від віку будуть пересуватися дуже багато). Також вони віднесли до недоліків відсутність готової бази даних за типами рухів розбитих на різні категорії. Кожному хто починає займатися питанням розпізнавання людських рухів, доводиться збирати свої власні дані.

У останній розглянутій науковій статті йдеться про огляд поточного прогресу в розпізнаванні людських рухів і можливості їх застосування [4]. Автор зазначає, що з того моменту як смартфони стали обов'язковим атрибутом сучасної людини, розробники всіляко намагаються використовувати їх можливості. Так існує багато ігор, що використовують гіроскоп, акселерометр та GPS для поліпшення ігрового процесу [5]. Такі ігри подобаються дітям і їх батькам, так як дитина граючи в такі ігри не залишається нерухомою, але при цьому ходить, виконує якісь рухи для досягнення успіху в ігровому процесі. У цих іграх дуже важливо, щоб якість ігрового процесу була на високому рівні, щоб у дитини не було бажання змінити гру і вона якомога більше часу проводила в ній. Завдяки таким іграм вийшло досягти гарного прогресу в використанні зв'язки гіроскопа, акселерометра та GPS. Існують посібникі, де описується як правильно працювати з датчиками, які є підводні камені і поширені проблеми. У активних іграх можно активно використовувати акселерометр, як сенсор для визначення типів рухів дитини. На даний момент сенсор не зможе визначити яким, наприклад, видом танцю займається дитина, але зрозуміти – танцює дитина чи ні, він може.

Аналізуючи амплітуди координат x, y, z, їх положення відносно один одного, можна зрозуміти (хоча б приблизно) чим займається дитина. Цю можливість можна використовувати в ігровому процесі та інших додатках. Також визначаючи найчастіший тип руху людини можна зробити припущення про його переваги. Це допоможе показувати більш якісну контекстну рекламу. Автор у своїй статті також

показує, що використання розпізнавання типів руху людини може застосовуватися як в цілях підвищення зручності використання смартфона, з метою розваг (найпоширеніший варіант) і в корисливих цілях (у рекламі). Також автор зазначає, що щоб підвищити якість мобільних розваг можна також використовувати камеру, але в цій області є певні проблеми зі стабілізацією та якістю зображень.

В процесі розробки своєї системи розпізнавання людських рухів автор зауважив, що людина рідко тримає телефон тільки в одному місці. Періодично, навіть під час руху користувач телефону дістає його з кишені (що б наприклад перемкнути пісню в плеєра). У такі моменти відбуваються різкі скачки амплітуд, навіть, якщо тип руху залишився тим же самим. Важлива частина процесу розробки це вміння розуміти коли користувач взаємодіє з пристроєм і при цьому автоматично підлаштуватися під зміни. Фактично, велику частину розробки займає розпізнавання таких нечастих, але критично важливих моментів використання мобільного пристрою, при яких можна сильно помилитися в розпізнаванні поточного типу руху. Автор заявляє, що передбачити їх все просто неможливо, навіть, якщо зібрати величезну кількість тренувальних даних і над процесом розпізнавання людських рухів ще необхідно працювати.

У списках основних завдань на майбутнє є створення програмного продукту, який буде працювати і з іншими розумними пристроями як фітнес-трекери або годинники. Але на даний момент розробка під них обмежена через проблеми з підтримкою браузерів. Виробники таких пристроїв часто не встановлюють в них веб-браузери, так як екрани на більшості годинників займають 2 дюйма, що незручно для веб-серфінгу. Крім того, навіть, якщо є веб-браузер, то залишається проблема з розробкою інтерфейсу під такий маленький екран і необхідно буде з самого початку захоплювати дані, так як годинники носяться не так і ті ж показники ходьби з акселерометра годинників не будуть співпадатимуть з таким пристроєм.

1.2 Аналіз проблеми розпізнавання рухів людини з мобільного пристрою

Огляд наукових робіт показав, що незважаючи на те, що існують вже готові рішення з розпізнавання людських рухів, в той же час жодне з них не можна назвати готовим на 100 відсотків.

Першою проблемою є накопичення даних і їх обробка. Від цього безпосередньо залежить якість розпізнавання рухів. Вибір методики та алгоритму розпізнавання також впливає на процес, але на неякісних даних навіть з найкращою методикою не вийде створити якісно працююче рішення. Ця проблема виникає через різний характер рухів у людей. Важливо враховувати вік, так як діти або люди похилого віку пересуваються набагато повільніше дорослих, і якщо тренувати на таких даних класифікатор, то він середню швидкість дорослих людей буде сприймати як біг. Часто люди протягом дня змінюють характер свого руху і необхідно правильно розпізнати перехід між, наприклад, стрибками і підйомами на сходи. Тут потрібно вручну аналізувати такі переходи, виявляти особливості, риси, і по ним спеціально тренувати класифікатор.

Варто відзначити складність в побудові навчальної характеристики, досить важко знайти баланс між швидкістю і якістю майбутніх распознаваний. Якщо зробити ознаку, що складається з 3–4 атрибутів, розпізнаватися вона буде швидко, але в той же час ймовірність помилок може досягати 15-20 відсотків [6]. У той же час, якщо зробити ознаку яка складається з 15 полів, можна знизити ймовірність помилок до 3–5 відсотків (це хороший результат в машинному навчанні), але процес розпізнавання буде займати більше часу.

Також варто відзначити, що на поточному етапі неможливо розпізнати за допомогою даних з акселерометра, гіроскопа і GPS складні рухи. Під складними рухами можна розуміти наприклад прибирання квартири, миття посуду, різні види танців. Розпізнавання таких рухів могло б допомогти розширити можливості смартфонів, дозволило б створювати нові види додатків, але з поточним наборів сенсорів це неможливо. Вищеперелічені сенсори не дають можливість створити

необхідне уявлення про характер складного руху але, можливо, з використанням додаткових пристроїв вийде досягти цього прогресу.

Однією з проблем є створення універсального програмного рішення. В світі існує величезна кількість мов програмування, існують різні версії операційних систем для мобільних телефонів. Дану різноманітність серед пристроїв важко програмно підтримувати. Рішення, написане для однієї операційної системи не буде автоматично запускатися на інший. Необхідно буде переписати з однієї платформи на іншу, враховуючи відповідні особливості платформи. В даному випадку рішенням проблеми є використання веб-екосистеми. Браузери є на всіх сучасних пристроях, вони надають мінімальний доступ до сенсорам смартфонів, як: доступ до GPS, акселерометру, гіроскопу, камері і мікрофону. Веб платформа єдина для мобільних пристроїв і для десктопних платформ. Єдине де треба здійснити свій вибір – це вибір бази даних, в якій будуть зберігатися дані для навчання і то, як буде виглядати серверна частина програми. Веб платформа найкраще підходить для того, що б написати одне рішення, яке буде працювати на всіх пристроях в яких є веб браузер.

Багато дослідників, які займаються проблемою розпізнавання рухів не використовують сучасні алгоритми класифікації в машинному навчанні. Машинне навчання (англ. Machine Learning, ML) – клас методів штучного інтелекту, характерною рисою яких є не пряме рішення задачі, а навчання в процесі застосування рішень множини подібних завдань. Для побудови таких методів використовуються засоби математичної статистики, чисельних методів, методів оптимізації, теорії ймовірностей, теорії графів, різні техніки роботи з даними в цифровій формі. Машинне навчання як галузь оновлюється щороку, вже існуючі алгоритми поліпшуються, створюються нові [7]. В машинному навчанні існує величезна кількість алгоритмів для вирішення різного ряду проблем. Алгоритми класифікації є одними з найбільш популярних алгоритмів в машинному навчанні, використовуючихся як в процесі розпізнавання зображень, так і в процесі роботи з гео-даними і т.д. Тому варто відзначити, що використання останніх напрацювань може значно підвищити якість розпізнавання людських рухів. Проблема якості та

швидкості визначення руху стоїть дуже гостро на поточному етапі розпізнавання людських рухів. Незважаючи на те, що зростає потужність мобільних пристроїв, існує цілий пласт бюджетних пристроїв, які не дозволяють широко розпоряджатися ресурсами. Для таких пристроїв на перший план виходить оптимізація існуючих алгоритмів розпізнавання під невисокі потужності. Такі бюджетні пристрої найчастіше є найпопулярнішими, тому, якщо на них визначення руху буде здійснюватися з значними пригальмовуванням, це знизить якість користування додатками в яких використовується процес визначення людських рухів – як, наприклад, фітнес-додатки, або деякі ігри.

Підвищити якість розпізнавання враховуючи будову тіла і вік людини можна використовуючи величезну кількість даних для навчання і іншу область роботи з даними – Data mining (інтелектуальний аналіз даних). Data Mining – це процес виявлення в "сирих" даних раніше невідомих нетривіальних, практично корисних і доступних інтерпретації знань, необхідних для прийняття рішень в різних сферах людської діяльності. Data Mining є одним з кроків Knowledge Discovery in Databases [8].

Інформація, знайдена в процесі застосування методів Data Mining, повинна бути нетривіальною і раніше невідомою. Знання повинні описувати нові зв'язки між властивостями, передбачати значення одних ознак на основі інших і т.д. Знайдені знання повинні бути застосовні і на нових даних з деякою мірою вірогідності. Корисність полягає в тому, що ці знання можуть приносити певну вигоду при їх застосуванні. Знання повинні бути в зрозумілій для користувача не-математика вигляді. Наприклад, найпростіше сприймаються людиною логічні конструкції "якщо ... то ...". Більш того, такі правила можуть бути використані в різних СУБД в якості SQL-запитів. У разі, коли витягнуті знання непрозорі для користувача, повинні існувати методи постобробки, що дозволяють привести їх до інтерпретованого виду. Завдяки використанню Data Mining можна буде перевіряти наскільки чітко людина виконує ті чи інші рухи просто аналізуючи графік його руху на основі даних з акселерометра.

1.3 Огляд існуючих мобільних додатків з розпізнавання рухів

Додатків, які використовують розпізнавання рухів, на даний момент не багато і в основному це фітнес додатки:

Google Fit – це додаток який збирає статистику про переміщення користувача, активності, пройдених кроках і потім всі ці дані відображаються у вигляді простих графіків. Ця програма для смартфона, який може взаємодіяти з Android Wear, але додаток здатен працювати і без сторонніх пристроїв. Дана програма автоматично вміє визначати яку активністю користувач робив у в різні моменти часу, і організовує статистику. Додаток написано на мові Java тільки для платформи Android, тому з мінусів варто відзначити швидкість роботи і високе енергоспоживання. Google Fit вміє визначати тільки людські руху, і тому не може визначити переміщення на велосипеді або роликах.

Moves – це програма яка розроблена під платформи iOS і Android. На сьогоднішній день це одне з кращих додатків для трекінгу активності з високою точністю визначення (близько 85%). Завдяки використанню GPS, програма знає, де знаходився користувач в кожен момент часу і будує карту переміщень. Визначає Moves і різницю між бігом, поїздкою на велосипеді і інший активністю. Дуже важливою особливістю програми є можливість передавати дані в інші програми. Таким чином можна використовувати Moves для класифікації рухів, а вихідну інформацію використовувати в іншому додатку. До недоліків варто віднести високе енергоспоживання, так як для додаткового підвищення точності програма так також задіює GPS, який споживає багато ресурсів.

ActivityRecognition – це технологія, яка написана для розпізнавання чотирьох типів рухів (стрибки, біг, ходьба, присідання) в режимі реального часу [9]. Розробник використав ряд сучасних технологій (Scala, Cassandra, Spark) разом з спеціальною бібліотекою для машинного навчання за рахунок чого досягнув високої якості розпізнавання укупі з досить високою швидкістю роботи (близько 89% правильних визначень). Додаток написано для системи Android, але частину для розпізнавання рухів може використовувати будь-який розробник на своєму

локальному комп'ютера і посилати до неї дані з будь-якого пристрою з акселерометром. Таким чином, даний додаток є основним на який слід орієнтуватися при розробці методики і наступного додатку, так як він містить в собі швидкість, високу точність розпізнавання і майже повну платформонезалежність.

1.4 Огляд веб-технологій для розробки додатків

Існує велика кількість різних фреймворків і бібліотек для розробки веб-додатків. Серед них необхідно виділити технології для розробки односторінкових веб-додатків. Такий тип програми відрізняється високою швидкістю роботи і головною метою забезпечити користувачу досвід близький до користування настільною програмою. Одним з основних фреймворків для розробки таких програм є Angular написаний на мові TypeScript [10]. Для роботи фреймворка також необхідна бібліотека RxJS для реактивного програмування [11]. Реактивне програмування — це парадигма програмування, побудована на потоках даних і розповсюдженні змін. Це означає, що у мовах програмування має бути можливість легко виразити статичні чи динамічні потоки даних, а реалізована модель виконання буде автоматично розсилати зміни через потік даних. Використання зв'язки з таких технологій дозволить створити додаток чуйний на дії користувачів.

1.5 Висновки до першого розділу

В даному розділі був проведений огляд наукових робіт з питання розпізнавання людських рухів. Був проведений аналіз існуючих методик і додатків, виділені їхні переваги і недоліки. Сформовані підходи щодо поліпшення точності і швидкості розпізнавання. Зроблений огляд поточних програмних рішень і технологій в галузі захоплення даних з акселерометра, створення високопродуктивних веб-додатків, серед яких виділені ті, що будуть використовуватися в процесі розробки мобільного веб-додатку. Сформовано список алгоритмів класифікації для використання в методиці розпізнавання типів людських рухів.

2 ДОСЛІДЖЕННЯ МЕТОДІВ І АЛГОРИТМІВ ВИЗНАЧЕННЯ ТИПІВ РУХУ ЛЮДИНИ

2.1 Аналіз існуючих алгоритмів класифікації даних руху

Аналіз наукових робіт в попередньому розділі дозволив сформувати список вимог до розроблюваної методики і вибрати найбільш оптимальні алгоритми для вирішення задачі розпізнавання рухів.

Так як буде визначаться до якого типу руху буде відноситися здійсненена людиною дія, то дана задача є завданням класифікації. Для побудови методики необхідне дослідження яке дозволить вибрати кращий алгоритм класифікації по метриці кількості правильних распознаваний. Також в процесі дослідження необхідно визначити з якими налаштуваннями потрібно його використовувати.

Класифікація це впорядкована по деякому принципу множина об'єктів, які мають подібні класифікаційні ознаки (одну або декілька властивостей), обраних для визначення подібності або відмінності між цими об'єктами. Під класифікацією слідує розуміти віднесення об'єктів (спостережень, подій) до одного з заздалегідь відомих класів. Класифікація – це закономірність, що дозволяє робити висновок щодо визначення характеристик конкретної групи. Таким чином, для проведення класифікації повинні бути присутні ознаки, що характеризують групу, до якої належить та чи інша подія або об'єкт (зазвичай при цьому на підставі аналізу вже класифікованих подій формулюються якісь правила). Класифікація відноситься до стратегії навчання з учителем (supervised learning), яке також називають контрольованим або керованим навчанням.

Класифікація може бути одновимірною (за однією ознакою) і багатовимірною (за двома і більше ознаками) [12].

Мета процесу класифікації полягає в тому, щоб побудувати модель, яка використовує прогнозуючи атрибути в якості вхідних параметрів і отримує

значення залежного атрибута. Процес полягає в розбитті множини об'єктів на класи за певним критерієм.

Для того щоб навчити модель попередньо будуть записані дані по кожному типу руху, на яких буде проводиться тренування та перевірка якості. Аналіз наукових робіт допоміг встановити найбільш підходящі для методики алгоритми класифікації, які дозволяють отримати високу точність визначення рухів укупі з невисоким споживанням ресурсів:

– K-Nearest Neighbors (метод найближчих k-сусідів);

– Naive Bayes Classifier (наївний баєсів класифікатор);

– Decision Tree (дерево ухвалення рішень);

– Random Forest (випадковий ліс).

Головне завдання навчальних алгоритмів – їх здатність узагальнюватися, тобто добре працювати на нових даних. Оскільки на нових даних відразу не можна перевірити якість побудованої моделі (треба для них зробити прогноз, тобто істинних значень цільового ознаку ми для них не знаємо), то треба пожертвувати невеликою порцією даних, щоб на ній перевірити якість моделі. Найчастіше це робиться одним з 2 способів:

– відкладена вибірка (held-out/hold-out set). При такому підході залишається якась частка навчальної вибірки (як правило від 20% до 40%), навчається модель на інших даних (60-80% вихідної вибірки) і рахується деяка метрика якості моделі (найпростіше – частка правильних відповідей в задачі класифікації) на відкладеної вибірці;

– крос-валідація (cross-validation, ще перекладають як ковзний або перехресний контроль).

У процесі розробки методики використовуватиметься відкладена вибірка, так як цей метод більш поширений, і повністю підходить під вимоги. Буде використовуватися 70% вибірки (x_train, y_train) під навчання і 30% будуть відкладеною вибіркою (x_holdout, y_holdout). Відкладена вибірка ніяк не братиме участі в налаштуванні параметрів моделей, на ній в кінці, після настройки, оцінюватиметься якість отриманої моделі.

Дані з акселерометра надходять в вигляді списку значень:

– x – значення по осі X;

– y – значення по осі X;

– z – значення по осі X;

– timestamp – відмітка часу коли було отримано значення.

2.2 Аналіз K-Nearest Neighbors алгоритму

Людина, зустрічаючись з новою задачею, використовує свій життєвий досвід, згадує аналогічні ситуації, які колись з ним відбувалися. Про властивості нового об'єкта ми судимо, покладаючись на схожі знайомі спостереження. Наприклад, зустрівши іноземця на вулиці, можна здогадатися про його походження з мови, жестів і зовнішності. Для цього необхідно згадати найбільш схожу на нього людину, походження якої вже відомо.

Так, подібно до наведеного вище прикладу, схожість об'єктів лежить в основі алгоритму k-найближчих сусідів (k-nearest neighbor algorithm, KNN). Алгоритм здатний виділити серед всіх спостережень k-відомих об'єктів (k-найближчих сусідів), схожих на новий невідомий раніше об'єкт. На основі класів найближчих сусідів виноситься рішення щодо нового об'єкта [13]. Сусіди беруться, виходячи з множини об'єктів, класи яких уже відомі, і, виходячи з ключового для даного методу значення k, вираховується, який клас є найчисленнішим серед них. Кожен об'єкт має кінцеву кількість атрибутів (розмірностей). Передбачається, що існує певний набір об'єктів з уже наявною класифікацією. Важливим завданням даного алгоритму є підбір коефіцієнта k – кількість записів, які будуть вважатися близькими.

Переваги:

– алгоритм стійкий до аномальних відхилень, тому що ймовірність попадання такого запису в число k-найближчих сусідів мала. Якщо ж це сталося, то вплив на голосування (особливо зважене) (при k> 2) також, швидше за все, буде незначним, і, отже, малим буде і вплив на підсумок класифікації;

– програмна реалізація алгоритму відносно проста;

– результат роботи алгоритму легко піддається інтерпретації. Експертам в різних областях цілком зрозуміла логіка роботи алгоритму, заснована на знаходженні схожих об'єктів;

– можливість модифікації алгоритму, шляхом використання найбільш придатних функцій поєднання і метрик можна підлаштувати алгоритм під конкретну задачу.

Алгоритм KNN має і ряд недоліків. По-перше, набір даних, який використовується для алгоритму, повинен бути репрезентативним. По-друге, модель не можна "відокремити" від даних: для класифікації нового прикладу потрібно використовувати всі приклади. Ця особливість сильно обмежує використання алгоритму [13].

2.2.1 Приклад класифікації з використанням KNN

Нехай є m спостережень, кожному з яких відповідає запис в таблиці. Всі записи належать якомусь класу. Необхідно визначити клас для нового запису.

На першому кроці алгоритму слід задати число k – кількість найближчих сусідів. Якщо прийняти $k = 1$, то алгоритм втратить узагальнюючу здатність (тобто здатність видавати правильний результат для даних, що не зустрічалися раніше в алгоритмі) так як нового запису буде присвоєно клас близькому до неї. Якщо встановити занадто велике значення, то буде виявлено багато локальних особливостей.

На другому кроці знаходяться k записів з мінімальною відстанню до вектора ознак нового об'єкта (пошук сусідів). Функція для розрахунку відстані повинна відповідати наступним правилам:

– $d(x, y) \geq 0$, $d(x, y) = 0$ тоді і тільки тоді, коли $x = y$;

– $d(x, y) = d(y, x)$;

– $d(x, z) \leq d(x, y) + d(y, z)$, за умови, що точки x, y, z чи не лежать на одній прямій.

де *x, y, z* – вектори ознак порівнюваних об'єктів.

Для впорядкованих значень атрибутів знаходиться Евклідова відстань:

$$D_E = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}, \tag{2.1}$$

де *n* – кількість атрибутів.

Для строкових змінних, які не можуть бути впорядковані, може бути застосована функція відмінності, яка задається наступним чином:

$$dd(x, y) = \begin{cases} 0, & x = y \\ 1, & x \neq y \end{cases} \tag{2.2}$$

При знаходженні відстані іноді враховують значимість атрибутів. Вона визначається експертом або аналітиком суб'єктивно, покладаючись на власний досвід. У такому випадку при знаходженні відстані кожен і-ий квадрат різниці в сумі множиться на коефіцієнт *Zi*. Наприклад, якщо атрибут *A* в три рази важливіше атрибута *B (ZA = 3, ZB = 1)*, то відстань буде знаходитись наступним чином:

$$D_E = \sqrt{3(x_A - y_{A)}{}^2 + (x_B - y_B)^2} \tag{2.3}$$

Подібний прийом називають розтягуванням осей (stretching the axes), що дозволяє знизити помилку класифікації.

Слід зазначити, що якщо для запису *B* найближчим сусідом є *A*, то це не означає, що *B* – найближчий сусід *A* (рис 2.1). При *k = 1* найближчій для точки *B* буде точка *A*, а для *A – X*. при збільшенні коефіцієнта до *k = 7*, точка *B* так само не буде входити в число сусідів.

На наступному кроці, коли знайдені записи, найбільш схожі на нову, необхідно вирішити, як вони впливають на клас нового запису. Для цього

використовується функція поєднання (combination function). Одним з основних варіантів такої функції є просте незважене голосування (simple unweighted voting).

Спочатку, задавши число *k*, визначається, скільки записів буде мати право голосу при визначенні класу. Потім виявляються записи, відстань від яких до нової виявилося мінімальним. Відтепер можна приступити до простого незваженого голосування.

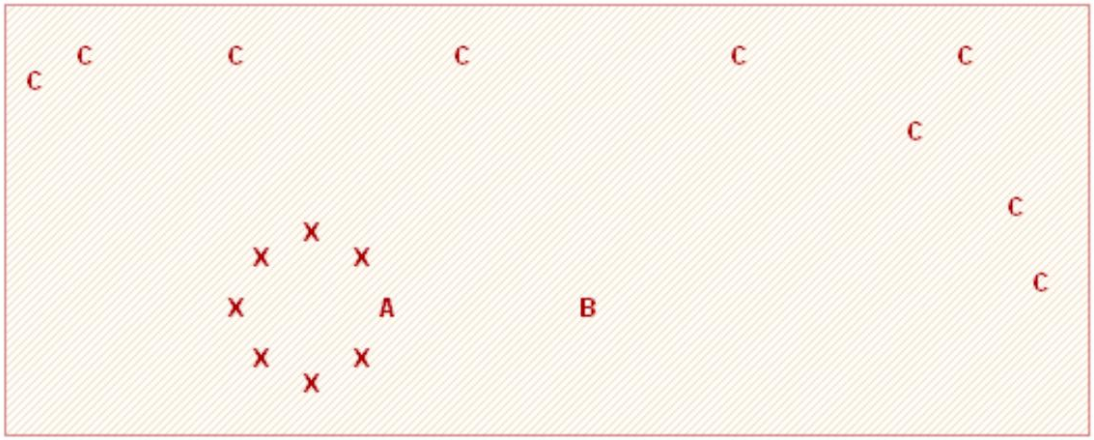

Рисунок 2.1 – Найближчі сусіди A і B

Відстань від кожного запису при голосуванні тут більше не грає ролі. Всі мають рівні права у визначенні класу. Кожна вільна позиція голосує за клас, до якого належить. Новому запису присвоюється клас, який набрав найбільшу кількість голосів.

Проблему, якщо декілька класів набрали рівну кількість голосів, знімає зважене голосування (weighted voting).

У такій ситуації враховується також і відстань до нового запису. Чим менше відстань, тим більш значущий внесок вносить голос. Голоси за клас знаходяться за наступною формулою:

$$votes(class) = \sum_{i=i}^{n} \frac{1}{d^2(X,\ Y_i)}, \tag{2.4}$$

де d^2 *(X, Yi)* – квадрат відстані від відомої записи *Yi* до нової *X*;

n – кількість відомих записів класу, для якого розраховуються голосу;

class – найменування класу.

Новий запис співвідноситься з класом, який набрав найбільшу кількість голосів. При цьому ймовірність того, що кілька класів наберуть однакові голоси, набагато нижче [13].

2.2.2 Практичне використання KNN

Основні параметри класифікатора:

– weights: "uniform" (всі ваги рівні), "distance" (вага обернено-пропорційна відстані до тестового прикладу) або інша визначена користувачем функція;

– algorithm (опціонально): "brute", "ball_tree", "KD_tree", або "auto". У першому випадку найближчі сусіди для кожного тестового прикладу рахуються перебором навчальної вибірки. У другому і третьому – відстань між прикладами зберігаються в дереві, що прискорює знаходження найближчих сусідів. У разі зазначення параметра "auto" відповідний спосіб знаходження сусідів буде обраний автматіческі на основі навчальної вибірки;

– leaf_size (опціонально): поріг перемикання на повний перебір в разі вибору BallTree або KDTree для знаходження сусідів;

– metric: "minkowski", "manhattan", "euclidean", "chebyshev" та інші.

Для тестуемої моделі будуть використовуватися такі параметри: weights – "uniform", algorithm – "auto", metric – "euclidean". Число сусідів задається автоматично і воно дорівнює числу класів + 1 [6].

Результати роботи алгоритму KNN приведені у таблиці 2.1

Таблиця 2.1 – Результати роботи алгоритму KNN.

Назва руху	Точність визначення
Стояння	80,1%
Сидіння	82,2%
Ходьба	78,0%
Стрибки	81,0%
Середня точність	80,3%

Можна зробити висновок що KNN алгоритм показав гарну точність, схожу з значеннями в інших наукових работах.

2.3 Аналіз Naive-Bayes алгоритму

Наївний байесовский алгоритм (NBA) – це алгоритм класифікації, заснований на теоремі Байєса з припущенням про незалежність ознак [14]. Іншими словами, NBA передбачає, що наявність якої-небудь ознаки в класі не пов'язано з наявністю будь-якої іншої ознаки. Наприклад, фрукт може вважатися яблуком, якщо він червоний, круглий і його діаметр становить близько 8 сантиметрів. Навіть, якщо ці ознаки залежать один від одного або від інших ознак, в будь-якому випадку вони вносять незалежний внесок у ймовірність того, що цей фрукт є яблуком. У зв'язку з таким припущенням алгоритм називається «наївним».

Моделі на основі NBA досить прості і корисні при роботі з дуже великими наборами даних. При своїй простоті NBA здатний перевершити навіть деякі складні алгоритми класифікації.

Теорема Байєса дозволяє розрахувати апостеріорну ймовірність P(c|x) на основі *P(c), P(x) і P(x|c)*:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}, \tag{2.5}$$

де $P(c|x)$ – апостериорная ймовірність даного класу с (тобто даного значення цільової змінної) при даному значенні ознаки х;

$P(c)$ – апріорна ймовірність даного класу;

$P(x|c)$ – правдоподібність, тобто ймовірність даного значення ознаки при даному класі;

$P(x)$ – апріорна ймовірність даного значення ознаки.

Залежно від точної природи ймовірнісної моделі, наївні байєсівські класифікатори можуть навчатися дуже ефективно. У багатьох практичних додатках для оцінки параметрів для наївних байєсівських моделів використовують метод максимальної правдоподібності; іншими словами, можна працювати з наївною байєсівської моделлю, не вірячи в Байєсову ймовірність і не використовуючи байєсівськи методи.

Незважаючи на наївний вигляд і, безсумнівно, дуже спрощені умови, наївні байєсівськи класифікатори часто працюють набагато краще в багатьох складних життєвих ситуаціях. Алгоритм naive bayes широко застосовується в Data Mining.

Переваги:

– класифікація, в тому числі багатокласова, виконується легко і швидко;

– коли допущення про незалежність виконується, NBA перевершує інші алгоритми, такі як логістична регресія (logistic regression), і при цьому вимагає менший обсяг навчальних даних;

– NBA краще працює з категорійними ознаками, ніж з безперервними. Для безперервних ознак передбачається нормальний розподіл, що є досить сильним допущенням.

Недоліки:

– якщо в тестовому наборі даних є певне значення категорійної ознаки, яке не зустрічалося в навчальному наборі даних, тоді модель присвоїть нульову ймовірність цього значення і не зможе зробити прогноз. Це явище відоме під назвою «нульова частота» (zero frequency). Дану проблему можна вирішити за допомогою згладжування. Одним з найпростіших методів є згладжування по Лапласу (Laplace smoothing);

– хоча NBA є хорошим класифікатором, значення прогнозованих ймовірностей не завжди є достатньо точними;

– ще одним обмеженням NBA є припущення про незалежність ознак. В реальності набори повністю незалежних ознак зустрічаються вкрай рідко.

NBA широко застосовується у:

– класифікації в режимі реального часу. NBA дуже швидко навчається, тому його можна використовувати для обробки даних в режимі реального часу;

– багатокласової класифікації. NBA забезпечує можливість багатокласової класифікації. Це дозволяє прогнозувати ймовірність для множини значень цільової змінної;

– рекомендаційній системи. Наївний байєсівський класифікатор в поєднанні з коллаборативной фільтрацією (collaborative filtering) дозволяє реалізувати рекомендаційну систему. В рамках такої системи за допомогою методів машинного навчання та інтелектуального аналізу даних нова для користувача інформація фільтрується на підставі прогнозованої думки цього користувача про неї.

2.3.1 Практичне використання Naive Bayes

Розподілення для наївного байєсівського алгоритму:

– Gaussian (нормальний розподіл). Алгоритм даного типу використовується в разі безперервних ознак і передбачає, що значення ознак мають нормальний розподіл;

– Multinomial (поліноміальний розподіл). Використовується в разі дискретних ознак. Наприклад, в задачі класифікації текстів ознаки можуть показувати, скільки разів кожне слово зустрічається в даному тексті;

– Bernoulli (розподіл Бернуллі). Використовується в разі двійкових дискретних ознак (можуть приймати тільки два значення: 0 і 1). Наприклад, в задачі класифікації текстів із застосуванням підходу «мішок слів» (bag of words) бінарна ознака визначає присутність (1) або відсутність (0) даного слова в тексті.

Для поточної задачі підходіть поліноміальний розподіл.

Також наївний баєсів класифікатор має набір параметрів, доступних для налаштування:

– alpha – активує згладжування по Лапласу
– fit_prior – визначає, навчати чи модель апріорним можливостям класів.

Будуть використовуватися такі настройки: alpha – 0, fit_prior – false [6];

Результати роботи Naive Bayes classifier приведені у таблиці 2.2

Таблиця 2.2 – Результати роботи Naive Bayes classifier.

Назва руху	Точність визначення
Стояння	88,8%
Сидіння	86,4%
Назва руху	Точність визначення
Ходьба	86,0%
Стрибки	82,1%
Середня точність	85,8%

2.4 Аналіз Decision Tree класифікатора

Дерева регресії і класифікації, відомі також під загальною назвою як дерева рішень (Decision Tree – DT), які являють собою структуру даних, яка дозволяє інтерпретувати шаблони даних з метою їх розпізнання. Дерева рішень організовані у вигляді ієрархічної структури, яка складається із вузлів прийняття рішень з оцінки значень певних змінних для прогнозування результуючого значення. [15].

Величезна перевага дерев рішень в тому, що вони легко інтерпретуються, і є зрозумілими для людині. Багато інших, хоч і більш точних, моделей не володіють цією властивістю і можуть розглядатися скоріше як "чорний ящик", в який завантажили дані і отримали відповідь. У зв'язку з цією "зрозумілістю" дерев рішень і їх схожістю з моделлю прийняття рішень людиною дерева рішень отримали величезну популярність, а один з представників цієї групи методів

класифікації, C4.5, розглядається першим в списку 10 кращих алгоритмів інтелектуального аналізу даних [10].

Будь-яке дерево рішень виводить значення яке прогнозується, отримане в результаті оцінки деяких вхідних атрибутів. Дерева рішень поділяються на два різних типа: дерева класифікації і дерева регресії. Ця різниця не залежить від типів вхідних даних, оскільки дерева того і іншого типу можуть приймати або неперервні, або символічні значення. Визначним фактором, від якого залежить тип дерева, є вихідне значення. Дерево рішень з неперервними вихідними значеннями іменується деревом регресії, а дерева класифікації замість цього виводять конкретні значення.

Будь-яке дерево рішень по-суті являє собою деревовидний граф. Ця структура даних складається з вузлів, з'єднаних один з одним ребрами (рис 2.2). При цьому не допускається, щоб ребра утворювали цикл, так як в противному випадку дерево перетворюється в граф, відмінний від деревовидного (а при використанні такого графа для прийняття рішень виникають ускладнення).

Дерево має один особливий вузол, відомий як кореневий. По суті, цей вузол є основою дерева, оскільки від кореня можна перейти по дереву до будь-якого іншого вузла. Ще до одного особливого різновиду вузлів відносяться вузли, що знаходяться в кінці будь-якої ланцюжка поспіль ребер – листові вузли.

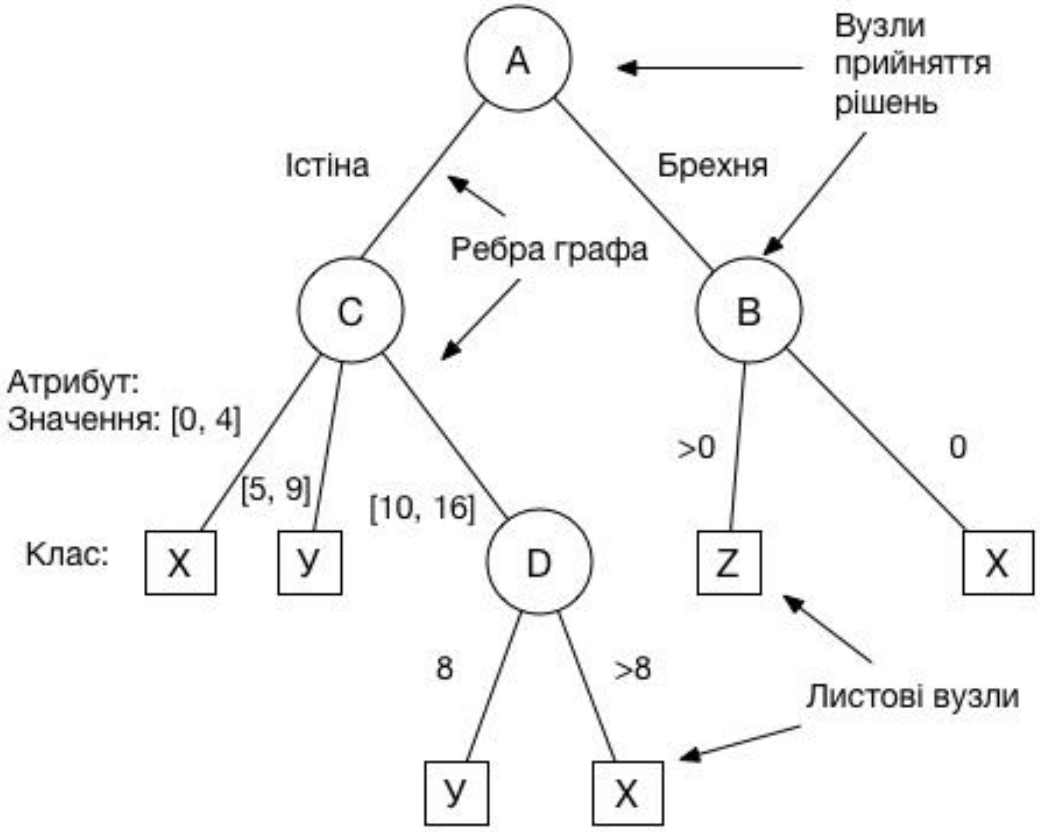

Рисунок 2.2 – Дерево рішень

Кількість можливих способів подання рішень повинно бути дуже велика, тому вибір способу, який використовується, залежить від типу атрибута, що перевіряється, а також від операції, яка використовується при перевірці умов. Оскільки атрибути можуть мати вираз у вигляді символів або неперервних значень, самі перевірки можуть бути організовані у вигляді булевих умов або неперервних відносин. Від кількості можливих результатів перевірки залежить те, скільки ребер повинно виходити з вузлаприйняття рішень. Нижче перераховані перевірки умов, які найбільш часто розглядаються:

– перевірка булевих значень. При проведенні такої перевіркивизначається те, чи призводить застосування якогось конкретного оператора до отримання істинного або хибного значення. Очевидно, що можливими результатами перевірки становляться істина або брехня;

– визначення знаку. При виконанні такої перевірки визначається знак виразу. Результатом може бути або позитивне або негативне значення. Вказана перевірка може розглядатися як окремий випадок перевірки булевих значень.

– перевірка приналежності до класу. При виконанні такої перевірки визначається, до якого класу належить даний символ. Результатом перевірки стає визначення одного із можливих класів(кількість яких може бути довільною).

– перевірка приналежності до області значень. При проведенні такої перевірки повинно бути вияснено, до якої області значень відноситься дане значення. Кожен з можливих результатів вказує, до якої області значень, на які поділяється вся область значень змінної, відноситься дане значення. Така перевірка може розглядатися як перевірка належності до класу [15].

Як правило, в кожному вузлі прийняття рішень виконується єдина перевірка умови, що охоплює тільки один атрибут (як приклад можна вказати операцію перевірки $B === true$). Такий підхід часто стає цілком прийнятним, оскільки дерево рішень дозволяє створювати ієрархічні комбінації перевірок для формування більш складних структур прийняття рішень. Однак деякі більш складні різновиди дерев рішень допускають проведення перевірок умов, в яких розглядається більшу кількість атрибутів (наприклад, одночасно перевіряються умови $A === 1$ і $B < 0$), що дозволяє підвищити точність перевірок, за рахунок ускладнення. Але в цьому випадку зростає кількість можливих комбінацій, тому збільшується кількість вихідних ребер.

Нижче перерахована кілька основних методів, які використовують дерева прийняття рішень.

CART (англ. Classification and regression trees – Класифікаційні і регресивні дерева) був першим з методів, придуманий в 1983 четвіркою відомих вчених в галузі аналізу даних: Leo Breiman, Jerome Friedman, Richard Olshen and Stone [11].

Суть цього алгоритму полягає в звичайній побудові дерева прийняття рішень.

На першій ітерації будуються всі можливі (в дискретному сенсі) гиперплоскості, які розбивають простір на два. Для кожного такого розбиття

простору рахується кількість спостережень в кожному з підпросторів різних класів. В результаті вибирається таке розбиття, яке максимально виділило в одному з підпросторів спостереження одного з класів. Відповідно, це розбиття буде коренем дерева прийняття рішень, а листами на даній ітерації буде два розбиття.

На наступних ітераціях береться один гірший (в сенсі відносини кількості спостережень різних класів) лист і проводиться та ж операція по розподіленню його. В результаті цей лист стає вузлом з розбиттям, і двома листами.

Так продовжується, поки не буде досягнуте обмеження за кількістю вузлів, або від однієї ітерації до іншої перестане зменшуватися загальна помилка (кількість неправильно класифікованих спостережень всім деревом). Однак, отримане дерево буде "перенавчанне" (буде підігнано під навчальну вибірку) і, відповідно, не буде давати нормальні результати на інших даних. Для того, що б уникнути "перенавчання", використовують тестові вибірки (або крос-валідацію) і, відповідно, проводиться зворотний аналіз (так званий pruning), коли дерево зменшують в залежності від результату на тестовій вибірці.

Це простий алгоритм, в результаті якого виходить одне дерево прийняття рішень. За рахунок цього, він зручний для первинного аналізу даних, наприклад, щоб перевірити наявність зв'язків між змінними.

Stochastic Gradient Boosting (стохастичне градієнтне додавання) – метод аналізу даних, представлений в 1999 році, і який представляє собою рішення задачі регресії (до якої можна звести класифікацію) методом побудови комітету (ансамблю) "слабких" дерев прийняття рішень [17].

На першій ітерації будується обмежене за кількістю вузлів дерево прийняття рішень. Після чого вважається різниця між тим, що передбачило отримане дерево помножене на learnrate (коефіцієнт "слабкості" кожного дерева) і шуканої змінної на цьому кроці.

$$Yi + 1 = Yi – Yi \cdot learnrate$$

І вже з цієї різниці будується наступна ітерація. Так триває, поки результат не перестане поліпшуватися. Тобто на кожному кроці йде спроба виправити помилки попереднього дерева. Однак тут краще використовувати перевірочні дані

(які не брали участі в моделюванні), так як на навчальних даних можливо перенавчання.

Переваги:

– породження чітких правил класифікації, зрозумілих людині, наприклад, "якщо вік <25 та інтерес до мотоциклів, то відмовити в кредиті". Це властивість називають інтерпретованістю моделі;

– дерева рішень можуть легко візуалізувати себе, тобто може "інтерпретуватися" як сама модель (дерево), так і прогноз для окремого взятого тестового об'єкта (шлях в дереві);

– швидкі процеси навчання і прогнозування;

– мале число параметрів моделі;

– підтримка і числових, і категоріальних ознак.

Недоліки:

– у породженні чітких правил класифікації є й інша сторона: дерева дуже чутливі до шумів у вхідних даних, вся модель може кардинально змінитися, якщо трохи зміниться навчальна вибірка (наприклад, якщо прибрати один з ознак або додати кілька об'єктів), тому і правила класифікації можуть сильно змінюватися, що погіршує интерпретованість моделі;

– розділяйма межа, побудована деревом рішень, має свої обмеження (складається з гіперплоскостей, перпендикулярних якийсь із координатної осі), і на практиці дерево рішень за якістю класифікації поступається деяким іншим методам;

– необхідність відсікати гілки дерева (pruning) або встановлювати мінімальне число елементів в листі дерева або максимальну глибину дерева для боротьби з перенавчанням. Втім, перенавчання – проблема всіх методів машинного навчання;

– нестабільність. Невеликі зміни в даних можуть суттєво змінювати побудоване дерево рішень. З цією проблемою борються за допомогою ансамблів дерев рішень.

– проблема пошуку оптимального дерева рішень (мінімального за розміром і здатного без помилок класифікувати вибірку) NP-повна, тому на практиці використовуються евристики жадібного типу пошуку ознаки з максимальним приростом інформації, які не гарантують знаходження глобально оптимального дерева;

– складно підтримуються пропуски в даних. Автор наукової статті оцінив, що на підтримку пропусків в даних пішло близько 50% коду CART (класичний алгоритм побудови дерев класифікації і регресії – Classification And Regression Trees) [18];

– модель вміє тільки інтерполювати, але не екстраполювати. Тобто дерево рішень робить константний прогноз для об'єктів, що перебувають у просторі ознак поза паралелепіпеду, що охоплює всі об'єкти навчальної вибірки.

2.4.1 Правило розбивки дерев в Decision Tree класифікаторі

Для побудови дерева на кожному внутрішньому вузлі необхідно знайти таку умову (перевірку), яке б розбивало множину, асоційовану з цим вузлом на підмножини. В якості такої перевірки повинен бути обраний один з атрибутів. Загальне правило для вибору атрибута можна сформулювати наступним чином: обраний атрибут повинен розбити множину так, щоб одержувані в результаті підмножини складалися з об'єктів, що належать до одного класу, або були максимально наближені до цього, тобто кількість об'єктів з інших класів ("домішок") в кожному з цих множин було якомога менше [13].

Найчастіше використовується теоретико-інформаційний критерій.

Алгоритм C4.5, вдосконалена версія алгоритму ID3 (Iterative Dichotomizer), використовує теоретико-інформаційний підхід.

Для вибору найбільш підходящого атрибута, пропонується наступний критерій:

$$Gain\,(X) = Info\,(T) - Info_x\,(T), \tag{2.6}$$

де *Info(T)* – ентропія множини *T*:

$$Info_x(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \bullet Info(T_i), \tag{2.7}$$

де *T1, T2, ... Tn* – множини отримані при розбитті початкової множини *T* з атрибуту *X*. Вибирається атрибут, що дає максимальне значення за критерієм.

Вперше ця міра була запропонована Р. Куінленом в розробленому ним алгоритмі ID3. Крім віщезгаданого алгоритму C4.5, є ще цілий клас алгоритмів, які використовують цей критерій вибору атрибута.

На додаток до основного методу побудови дерев рішень були запропоновані наступні правила:

– використання статистичних методів для оцінки доцільності подальшого розбиття, так звана "рання зупинка" (prepruning). В кінцевому рахунку "рання зупинка" процесу побудови приваблива в плані економії часу навчання, але тут доречно зробити одне важливе застереження: цей підхід будує менш точні класифікаційні моделі і тому рання зупинка вкрай небажана;

– обмеження глибини дерева. Зупинити подальшу побудову, якщо розбиття веде до дерева з глибиною, що перевищує задане значення;

– розбиття має бути нетривіальним, тобто отримані в результаті вузли повинні містити не менше заданої кількості прикладів.

Цей список евристичних правил можна продовжити, але на сьогоднішній день не існує такого правила, яке б мало велику практичну цінність. До цього питання слід підходити обережно, так як багато які з них застосовні в якихось окремих випадках [13].

2.4.2 Практичне використання Decision Tree

Основні параметри класифікатору:

– max_depth – максимальна глибина дерева;

– max_features – максимальне число ознак, за якими шукається краще розбиття в дереві (це потрібно тому, що при великій кількості ознак буде "дорого" шукати краще (за критерієм типу приросту інформації) розбиття серед усіх ознак);

– min_samples_leaf – мінімальне число об'єктів в листі. У цього параметра є зрозуміла інтерпретація: скажімо, якщо він дорівнює 5, то дерево буде породжувати тільки ті класифікують правила, які вірні як мінімум для 5 об'єктів.

Для нашої моделі будуть використовуватися такі параметри: max_depth – 20, max_features – 15, min_samples_leaf – 1. Також буде використовуватися метод CART для для побудови дерева [11].

Результати роботи Decision Tree класифікатору приведені у таблиці 2.3.

Таблиця 2.3 – Результати роботи алгоритму CART.

Назва руху	Точність визначення
Стояння	74,0%
Сидіння	76,2%
Ходьба	71,9%
Стрибки	73,4%
Середня точність	73,9%

Можна зробити висновок, що DT алгоритм показав не високу точність.

2.5 Аналіз Random Forest класифікатору

Випадковий ліс – один з небагатьох універсальних алгоритмів [20]. Універсальність полягає, по-перше, в тому, що він добре працює у багатьох завданнях (70% що зустрічаються на практиці, якщо не враховувати завдання з

зображеннями), по-друге, в тому, що випадковий ліс вирішую задачі класифікації, регресії, кластеризації, пошуку аномалій, селекції ознак і т.д.

Метод заснований на побудові великої кількості (ансамблю) дерев рішень (це число є параметром методу), кожне з яких будується за вибіркою, що отримується з вихідної навчальної вибірки за допомогою бутстрепа (т.е. вибірки з поверненням). На відміну від класичних алгоритмів побудови дерев рішень в алгоритмі випадкового лісу при побудові кожного дерева на стадіях розщеплення вершин використовується тільки фіксоване число випадково відбираних ознак навчальної вибірки і будується повне дерево (без усічення), т. е. кожен лист дерева містить спостереження тільки за одним класом. Класифікація здійснюється за допомогою голосування класифікаторів, що визначаються окремими деревами, а оцінка регресії – усередненням оцінок регресії всіх дерев. Відомо, що точність (ймовірність коректної класифікації) ансамблів класифікаторів істотно залежить від різноманітності (diversity) класифікаторів, що становлять ансамбль або, іншими словами, від того, наскільки корельовані їх вирішення. А саме, чим різноманітніші класифікатори ансамблю (менше коррелированность їх рішень), тим вище ймовірність коректної класифікації. У випадковому лісі рішення складових дерев слабо корельовані внаслідок подвійної "ін'єкції випадковості" в алгоритм побудови випадкового лісу – на стадії бутстрепа і на стадії випадкового відбору ознак, використовуваних при розщепленні вершин дерев.

Переваги:

– метод гарантує захист від перепідгонки (overfitting) навіть у разі, коли кількість ознак значно перевищує кількість спостережень. Це властивість виділяє метод "випадковий ліс" серед безлічі інших методів класифікації і є надзвичайно цінною для вирішення багатьох прикладних задач;

– метод Out-Of-Bag (OOB), забезпечує отримання природної оцінки ймовірності помилкової класифікації випадкових лісів на основі спостережень, що не входять в навчальні бутстреп вибірки, використовуваної для побудови дерев (ці спостереження називаються OOB вибірками);

– для побудови потрібно завдання всього двох параметрів;

– випадковий ліс може використовуватися не тільки для задач класифікації і регресії, а й для задач виявлення найбільш інформативних ознак, кластеризації, виділення аномальних спостережень і визначення прототипів класів;

– навчальна вибірка для побудови випадкового лісу може містити ознаки, виміряні в різних шкалах: числової, порядкової і номінальної, що неприпустимо для багатьох інших класифікаторів;

– метод допускає легку параллелизацію (тобто програмну реалізацію, придатну для паралельних обчислень), що має велике значення при великих обсягах навчальної вибірки.

Недоліки:

– великий розмір вихідної моделей. Потрібно O(NK) пам'яті для зберігання моделі, де K – число дерев;

– довга побудова моделі, для досягнення хороших результатів;

– складна інтерпретація моделі (сотні або тисячі великих дерев складні для інтерпретації).

2.5.1 Алгоритм навчання класифікатора Random Forest

Нехай навчальна вибірка складається з N прикладів, розмір простору ознак дорівнює M, і заданий параметр m (в задачах класифікації зазвичай $m \approx \sqrt{M}$)

Всі дерева комітету будуються незалежно один від одного за такою процедурою:

1. Генерується випадкова підвибірка з повторенням розміру N з навчальної вибірки (таким чином, деякі приклади потраплять в неї кілька разів, а приблизно $N / 3$ прикладів не ввійдуть в неї взагалі).

2. Будується вирішальне дерево, що класифікує приклади такої підвибірки, причому в ході створення чергового вузла дерева буде вибиратися ознака, на основі якої проводиться розбиття не з усіх M ознак, а лише m випадково обраних. Вибір найкращіх з цих m ознак може здійснюватися різними способами. В оригінальному коді використовується критерій Гіні, що застосовується також в алгоритмі

побудови вирішальних дерев CART. У деяких реалізаціях алгоритму замість нього використовується критерій приросту інформації.

3. Дерево будується до повного вичерпання підвибірки і не піддається процедурі прунінга (на відміну від вирішальних дерев, побудованих за такими алгоритмами, як CART і ID3).

Класифікація об'єктів проводиться шляхом голосування: кожне дерево комітету відносить об'єкт до одного з класів, і перемагає клас, за який проголосувала найбільша кількість дерев.

Оптимальне число дерев підбирається таким чином, щоб мінімізувати помилку класифікатора на тестовій вибірці. У разі її відсутності, мінімізується оцінка помилки out-of-bag: частка прикладів навчальної вибірки неправильно класифікуються комітетом, якщо не враховувати голоси дерев на прикладах, що входять в їх власну навчальну підвибірку.

2.5.2 Практичне використання Random Forest

Основні параметри класифікатора:

– n_estimators – число дерев. Чим більше дерев, тим краще якість, але час настройки і роботи RF також пропорційно збільшуються. Часто при збільшенні n_estimators якість на навчальній вибірці підвищується (може навіть доходити до 100%);

– max_features – число ознак для вибору розщеплення. При збільшенні max_features збільшується час побудови лісу, а дерева стають «більш одноманітними». За замовчуванням він дорівнює *sqrt(n)* в задачах класифікації та *n/3* в задачах регресії. Це найважливіший параметр, його налаштовують в першу чергу (при достатній кількості дерев у лісі);

– min_samples_split – мінімальна кількість об'єктів, при якому виконується розщеплення. Цей параметр, як правило, не дуже важливий і можна залишити значення за замовчуванням. При збільшенні параметра якість на навчанні падає, а час побудови RF скорочується;

– min_samples_leaf – обмеження на число об'єктів в листі. Все, що було описано про min_samples_split, годиться і для опису цього параметра. Часто можна залишити значення за замовчуванням;

– max_depth – максимальна глибина дерев. Чим менше глибина, тим швидше будується і працює RF. При збільшенні глибини різко зростає якість на навчанні, і на контрольній вибірці воно, як правило, збільшується. Рекомендується використовувати максимальну глибину (крім випадків, коли об'єктів занадто багато і виходять дуже глибокі дерева, побудова яких займає чимало часу). Неглибокі дерева рекомендують використовувати в задачах з великим числом шумових об'єктів (сплесків);

– criterion – критерій розщеплення. За змістом це дуже важливий параметр, але по факту тут немає варіантів вибору. Для класифікації реалізовані критерії "gini" і "entropy", які відповідають класичним критеріям розщеплення: джині і ентропійному. Простий перебір допоможе вибрати, що використовувати в конкретному завданні [19].

Для нашої моделі будемо використовувати такі параметри: estimators – 120, features – 64, max_depth – 20, criterion – “gini” [11]. Результати у таблиці 2.4.

Таблиця 2.4 – Результати роботи алгоритму RF.

Назва руху	Точність визначення
Стояння	87,5%
Сидіння	90,7%
Ходьба	89,8%
Стрибки	84,0%
Середня точність	88,0%

Також спробуємо підвищити швидкість і використовуємо такі настройки моделі: estimators – 60, features – 32, max_depth – 10, criterion – “gini”. Результати у таблиці 2.5.

Таблиця 2.5 – Результати роботи алгоритму RF з низькими настройками

Назва руху	Точність визначення
Стояння	85,1%
Сидіння	86,0%
Ходьба	82,3%
Стрибки	80,9%
Середня точність	83,6%

Можна помітити як сильно впливає на якість визначення зменшення кількості дерев, їх глибина і т.д. В даному випадку краще використовувати більш високі значення для параметрів так як в розроблювальній методиці якість розпізнавання має більш високий пріоритет ніж швидкість.

2.6 Перевірка алгоритмів

Для перевірки алгоритмів створювалась ознака, що складалась з шести параметрів: середнє значення по осях X, Y, Z та дисперсії по осях X, Y, Z. Ознака містила в себе значення протягом 2-х секунд з наступним зсувом для створення нової ознаки. Дані ознаки використовувались для навчання класифікаторів. Навчені класифікатори на тестовому дата-сеті показували ефективність роботи. Ефективність алгоритмів оцінювалась кількістю вірно розпізнаних рухів. Для порівняння алгоритмів було взято спеціальну програмну реалізацію за адресою https://github.com/christopherjenness/ML-lib.

2.7 Порівняння алгоритмів

Для вибору найкращого алгоритму потрібно порівняти їх всі за значеннями точності визначення рухів. Результати наведені у таблиці 2.6.

Таблиця 2.6 – Порівняння точності алгоритмів

Тип руху	Алгоритм			
	K-Nearest Neighbors	Naive Bayes Classifier	Decision Tree	Random Forest
Стояння	80,1%	88,8%	74,0%	87,5%
Сидіння	82,2%	86,4%	76,2%	90,7%
Ходьба	78,0%	86,0%	71,9%	89,8%
Стрибки	81,0%	82,1%	73,4%	84,0%
Середня точність	80,3%	85,8%	73,9%	88,0%

Random Forest класифікатор показав найкращі результати точності визначення рухів і буде використовуватися в розроблюваної методики в якості класифікатора.

2.8 Висновки до другого розділу

В даному розділі зроблений огляд поточних найефективніших алгоритмів класифікації, їхнє порівняння на тестовій вибірці, обран оптимальний алгоритм відповідно до поставлених критеріїв і визначені параметри налаштувань для нього. Обраний алгоритм Random Forest підтримує паралелізм що скорочує час навчання. Запропонована оптимізація процесу навчання за рахунок скорочення кількості даних. Запропонован список атрибутів в кількості 15 значень для створення ознаки. Була побудована методика розпізнавання людських руху використовуючи дані з мобільного пристрою.

3 МОДЕЛЮВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА

3.1 Розробка методики для визначення типу людських рухів

Методика розпізнавання людських рухів складається з наступних етапів (рис 3.1):

1. Збір даних по кожному типу рухів;
2. Збереження даних в базу даних;
3. Підготовка даних до навчання;
4. Створення ознак та іменних міток;
5. Вибір моделі (алгоритм класифікації даних);
6. Підбір параметрів моделі і алгоритму навчання;
7. Навчання моделі (автоматичний пошук інших параметрів моделі);
8. Аналіз якості навчання.

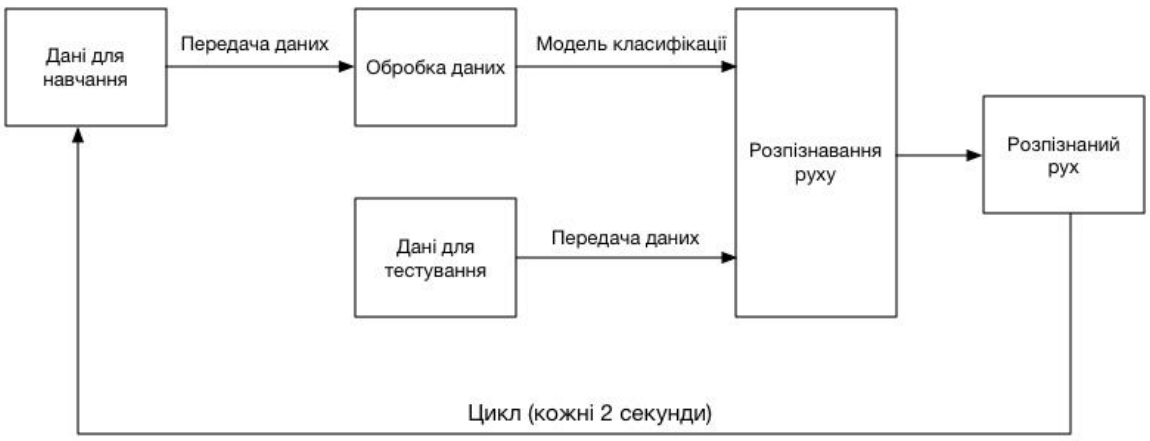

Рисунок 3.1 – Етапи розпізнавання людських рухів

Перший етап в розробці методики полягає в зборі даних з акселерометра смартфона. Дані надходять з клієнтської частини в сирому вигляді. Важлива частина на даному етапі в тому, що дані приходять в режимі реального часу

циклічно кожні 2 секунди. За цей час накопичується достатньо даних для побудови ознаки, так як середня частота роботи акселерометра близько 60 значень в секунду.

Наступний етап полягає в збереженні цих даних в базу даних. Щоб максимально підвищити продуктивність процесу, використовується NoSequel база даних, так як вона має JSON структуру, що дозволяє уникнути перетворень даних в процесі збереження і зчитування. Продуктивність у таких баз даних також знаходиться на високому рівні, крім того дані БД містить заздалегідь ряд вбудованих функцій, які дозволяють отримувати за допомогою агрегації середнє значення, дисперсію і т.д. [21]

Третій крок – підготовка даних. Деякі дослідники в своїх наукових работах згладжували дані, але це не дає ніякого приросту в точності, тому в розроблювальної моделі просто береться кожне десяте значення і це дозволяє уникнути багатьох повторюваних значень і зменшити обсяг даних для навчання (що в свою чергу підвищить швидкість навчання).

Четвертий крок – створення ознаки (feature). Це один з найважливіших кроків, так як від того, наскільки буде правильно створено ознаку залежить якість навчання і надалі – розпізнавання. Автори наукових робіт радять використовувати різні атрибути і підбирати кінцеву групу атрибутів для створення ознаки методом перебору [20]. Найчастіше використовуються такі атрибути:

– середнє значення по осі *X*;
– середнє значення по осі *Y*;
– середнє значення по осі *Z*;
– дисперсія по осі *X*;
– дисперсія по осі *Y*;
– дисперсія по осі *Z*;
– стандартне відхилення по осі *X*;
– стандартне відхилення по осі *Y*;
– стандартне відхилення по осі *Z*;
– середнє абсолютне відхилення по осі *X*;
– середнє абсолютне відхилення по осі *Y*;

– середнє абсолютне відхилення по осі *Z*;

– результуючий вектор;

– середній час між вершинами амплітуд;

– різниця між *X* і *Y* осями.

П'ятий крок – вибір алгоритму для навчання. Даний вибір зроблен на основі досліджень ефективних многоклассових алгоритмів класіфікації і найточнішим є Random Forest.

Шостий крок – вибір параметрів моделі. Цей крок також пройдено в процесі вибору алгоритму навчання. Застосовуються такі параметри – estimators – 120, features – 64, max_depth – 20, criterion – “gini”.

3.2 Алгоритм системи визначення типів руху.

Алгоритмічне представлення дозволяє розбити задачу на функціональні блоки переді її фактичною реалізацією.

Вхідні параметри: масив даних з акселерометра.

Вихідні параметри: розпізнані рухи.

Трудомісткість алгоритму: O (ntree · mtry · nlog(n)),

де ntree – кількість дерев, яка буде побудована.

mtry – кількість атрибутів, які будуть побудовані на кожному вузлі.

n – кількість записів.

Залежність споживання пам'яті від розміру вхідних параметрів: O(2n)

Отримане представлення алгоритму пожна зобразити у вигляді блок-схеми (рис. 3.1).

Рисунок 3.1 – Блок-схема алгоритму

Для експериментальної перевірки запропонованого алгоритму використані наступні технології та обладнання:

Raspberry Pi (рис.3.2) – одноплатний комп’ютер на на базі операційної системи Linux, мова програмування Python, яка дозволить ефекткивно використовувати можливості пристроів та акселерометра.

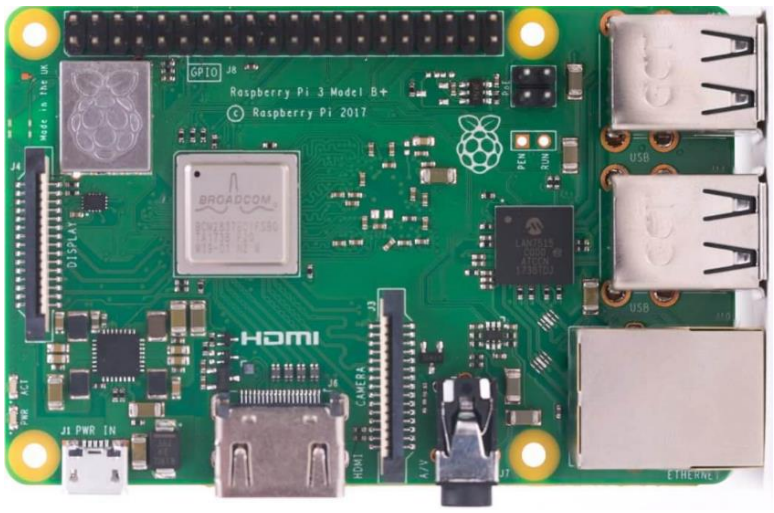

Рисунок 3.2 Raspberry Pi 3 Model B +

У якості датчика характеристик рухів використовується модуль акселерометр – mpu6050 (рис.3.3)

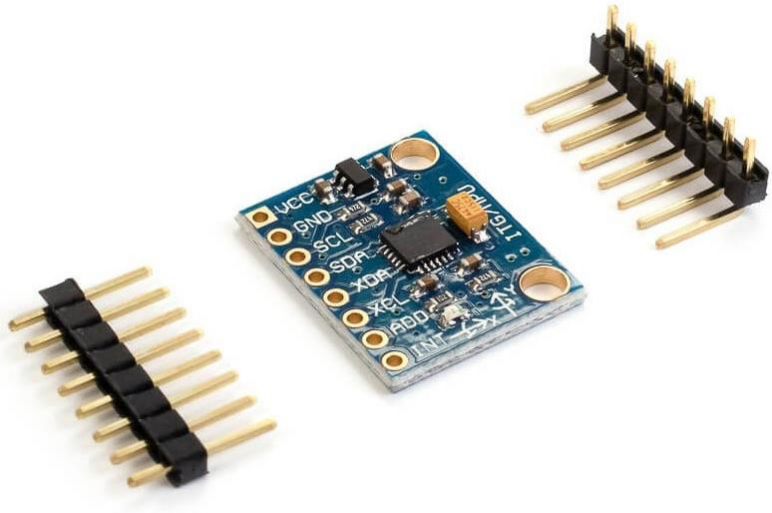

Рисунок 3.3 Акселерометр.

Пристрій на Raspberry Pi виконувується спільно з серверною платформою Node.js (рис 3.4).

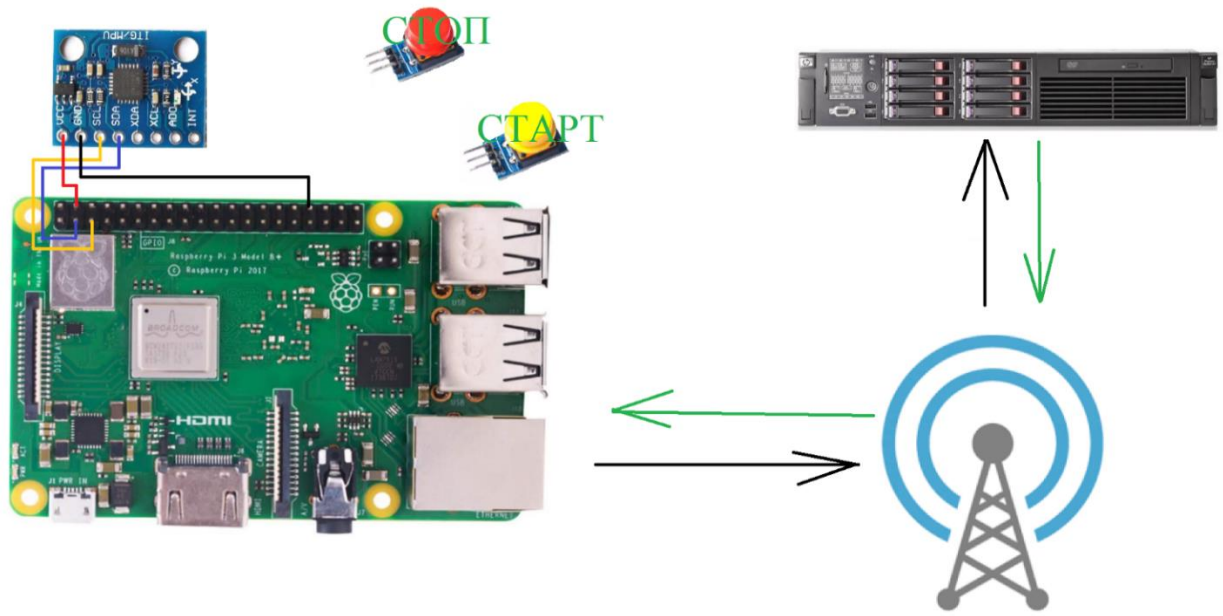

Рисунок 3.4 Взаємодія мобільного пристрою із серверною платформою.

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.

Після подключения Raspberry Pi до PC відкривається вікно з заголовком сторінки, невеликою підказкою по інтерфейсу з розпізнавання рухів користувачеві і кнопкою початку захоплення рухів "СТАРТ".

Після натискання на кнопку "СТАРТ" пристрий переходить в режим сполучення з мобільним пристроєм Raspberry Pi . Загоряється індекація LED, що відображає процес сполучення та захоплення рухів (рис 3.5).

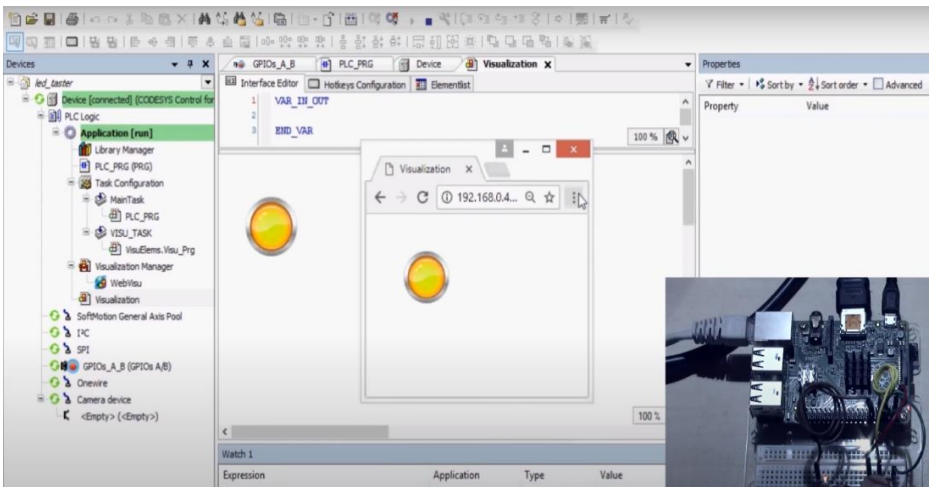

Рисунок 3.5 Сполучення з мобільним пристроєм Raspberry Pi

Скетч программи при передачі даних з акселерометра на Raspberry Pi наведено на (рис 3.6).

mpu_6050.py

```
from mpu6050 import mpu6050
import time
mpu = mpu6050(0x68)

while True:
    print("Temp : "+str(mpu.get_temp()))
    print()

    accel_data = mpu.get_accel_data()
    print("Acc X : "+str(accel_data['x']))
    print("Acc Y : "+str(accel_data['y']))
    print("Acc Z : "+str(accel_data['z']))
    print()

    gyro_data = mpu.get_gyro_data()
    print("Gyro X : "+str(gyro_data['x']))
    print("Gyro Y : "+str(gyro_data['y']))
    print("Gyro Z : "+str(gyro_data['z']))
    print()
    print("--------------------------------")
    time.sleep(1)
```

Рисунок 3.6 Листинг коду програми обробки даних з акселерометра

Скетч программи Raspberry Pi при передачі даних на сервер наведено у додатку Б.

Після натискання на кнопку "СТОП" захоплення рухів зупиняється і пристрій повертается до попереднього стану (рис.3.7)

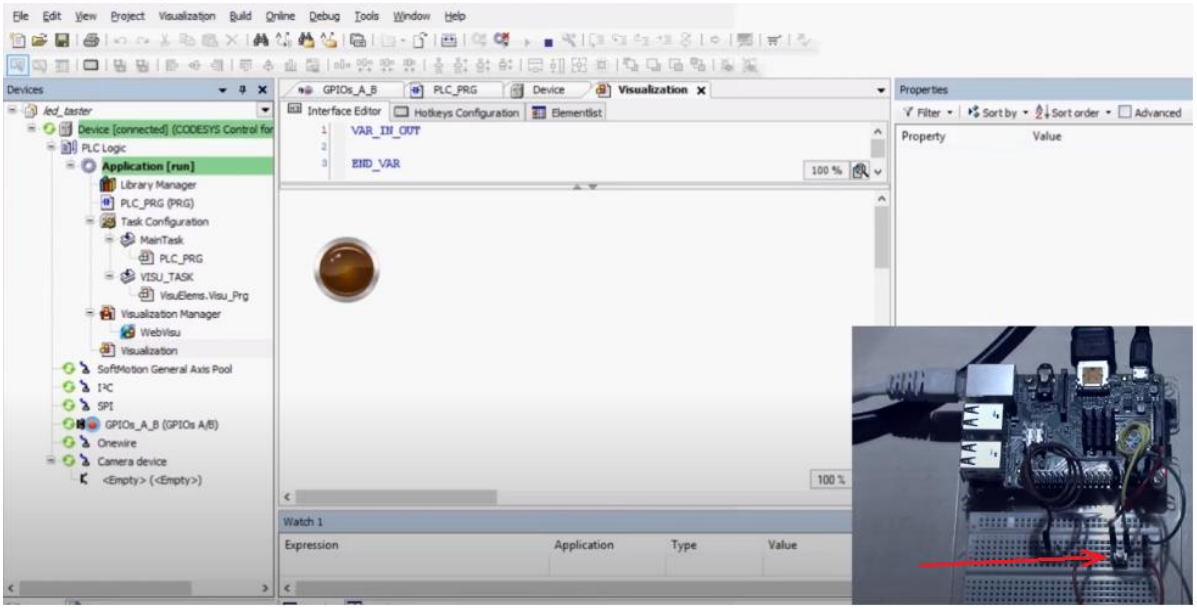

Рисунок 3.7 Після натискання на кнопку "СТОП" захоплення рухів зупиняється.

3.3 Оцінка ефективності роботи алгоритму класифікації

Для оцінки ефективності класифікатора були розроблені ознаки, які використовувались для розпізнавання рухів у складі вектору ознак. У програмному забезпеченні реалізована можливість налаштовувати вектор ознак, для визначення найефективнішого поєднання. Для перевірки ефективності процедури класифікації була проведена серія дослідів, у кожному з яких система працювала з певною конфігурацією вектору ознак з набором підготовлених тестових даних і на реальних даних також.

Всього було створено кілька ознак:

1. Середнє значення по осі *X*, *Y*, *Z*:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{3.1}$$

де x_i – значення *X*, *Y* або *Z*;

n – кількість записів у вибірці.

2. Дисперсія по осі *X*, *Y*, *Z*:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}, \tag{3.2}$$

де x_i – значення *X*, *Y* або *Z*;

n – кількість записів у вибірці;

μ – середнє значення по осі *X*, *Y* або *Z*.

3. Стандартне відхилення *X*, *Y*, *Z*:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}, \tag{3.3}$$

де x_i – значення *X*, *Y* або *Z*;

n – кількість записів у вибірці;

μ – середнє значення по осі *X*, *Y* або *Z*.

4. Середнє абсолютне відхилення *X*, *Y*, *Z*:

$$Mad = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2, \tag{3.4}$$

де x_i – значення *X*, *Y* або *Z*;

n – кількість записів у вибірці;

μ – середнє значення по осі *X*, *Y* або *Z*.

5. Результуючий вектор:

$$Res = \frac{1}{n}\sum_{i=1}^{n}\sqrt{x_i^2 + y_i^2 + z_i^2} \tag{3.5}$$

де x_i – значення *X*, y_i – значення *Y*, z_i – значення *Z*;

n – кількість записів у вибірці.

6. Середній час між вершинами амплітуд X і Z
7. Різниця між X і Y осями:

$$Diff = \mu(X) - \mu(Y) \tag{3.6}$$

де $\mu(X)$ – середнє значення по осі X, $\mu(Y)$ – середнє значення по осі Y;

Для оцінювання точності роботі векторів ознак були побудовані матриці помилок (confusion matrix). Матриця помилок - це матриця розміру N на N, де N - це кількість класів. Стовпці цієї матриці резервуються за рішеннями класифікатора, а рядки за експертними рішеннями. Коли класифікується документ з тестової вибірки то інкрементується число, що стоїть на перетині стовпця класу який повернув класифікатор і рядка класу до якого дійсно відноситься документ. У всіх матрицях використовуються такі позначення: 1 клас – є класом з типом руху "стояння", 2 – "сидіння", 3 – "ходьба", 4 – "стрибки".

Загальна точність вираховується за формулою:

$$Accuracy = \frac{n_1 + n_2 + n_3 + n_4}{N}, \tag{3.7}$$

де n_1, n_2, n_3, n_4 – діагональні елементи;

N – загальна кількість всіх розрахунків.

Результати використання вектору ознак [1, 2] приведені в таблиці 3.2.

Таблиця 3.2 – Матриця помилок для вектору ознак [1, 2]

		Прогнозовані класи				
		1	2	3	4	точність визначення руху
істинні класи	1	187	19	0	0	90,7%
	2	17	177	0	0	91,2%
	3	0	0	169	28	85,7%
	4	1	0	22	156	87,1%

Загальна точність класифікатора з вектором ознак [1, 2] $= \frac{689}{776} = 88{,}7\%$

Наступним тестом йде використання вектору ознак [1, 2, 3], результати приведені у таблиці 3.3.

Таблиця 3.3 – Матриця помилок для вектору ознак [1, 2, 3]

		Прогнозовані класи				
		1	2	3	4	точність визначення руху
істинні класи	1	191	15	0	0	92,7%
	2	12	182	0	0	93,8%
	3	0	0	176	21	89,3%
	4	0	0	21	158	88,2%

Загальна точність класифікатора з вектором ознак [1, 2, 3] $= \frac{707}{776} = 91{,}1\%$

Наступним тестом йде використання вектору ознак [1, 2, 3, 4], результати приведені у таблиці 3.4.

Таблиця 3.4 – Матриця помилок для вектору ознак [1, 2, 3, 4]

		Прогнозовані класи				
		1	2	3	4	точність визначення руху
істинні класи	1	195	9	2	0	94,6%
	2	9	185	0	0	95,3%
	3	1	0	185	11	93,9%
	4	0	0	17	162	90,5%

Загальна точність класифікатора з вектором ознак [1, 2, 3, 4] $= \frac{727}{776} = 93{,}6\%$

Наступним тестом йде використання вектору ознак [5, 6, 7], результати приведені у таблиці 3.5.

Таблиця 3.5 – Матриця помилок для вектору ознак [5, 6, 7]

		Прогнозовані класи				
		1	2	3	4	точність визначення руху
істинні класи	1	179	21	6	0	86,8%
	2	23	170	1	0	87,6%
	3	7	0	160	30	81,2%
	4	0	0	22	156	87,1%

Загальна точність класифікатора з вектором ознак [5, 6, 7] $= \frac{665}{776} = 85{,}6\%$

Останнім тестом йде використання всіх ознак, результати приведені у таблиці 3.6.

Таблиця 3.6 – Матриця помилок для вектору з всіма ознаками

		Прогнозовані класи				
		1	2	3	4	точність визначення руху
істинні класи	1	144	57	5	0	69,9%
	2	51	136	6	1	70,1%
	3	4	0	131	62	66,4%
	4	5	0	42	132	73,7%

В результаті проведеного експерименту було визначено, що найкращі результати показав вектор з ознаками номер 1, 2, 3, 4. У той же час якщо використовувати всі доступні ознаки, то результат виходить найгіршим. Швидше за все це пов'язано з перенавчанням дерев. Середнє значення точності найкращого набору вектора ознак становить 93,6%.

Також було проведено дослідження на швидкість роботи алгоритму класифікації. Швидкість оцінювалася часом, витраченим на очікування відповіді з сервера виключаючи з нього пінг (рис 3.8). Швидкість роботи класифікатора склала 20.5 мс. Це відмінний показник, і він дорівнює приблизно 10 кадрам в секунду. Тобто у браузера є ще 50 кадрів з 60 щоб обробити відповідь і вивести її на екран. Якщо виходить працювати з даними в рамках від 30 до 60 кадрів, то користувач не бачить ніяких пригальмовувань інтерфейсу додатку.

Але повний витрачений час визначення рухів складається з часу витраченого на захоплення самих рухів плюс час витрачений на відправку запиту, роботу класифікатора і час витрачений на відправку та отримання відповіді.

Найбільш оптимальним результатом тривалості захоплення рухів є 2 секунди, за цей час акселерометр захоплює достатню кількість даних, що б можна було визначити тип руху. Близько 2 мс йде на відправку запиту розпізнавання і передачі даних, 20 мс займає работа класифікатора, і близько 5 мс відправка відповіді до браузера.

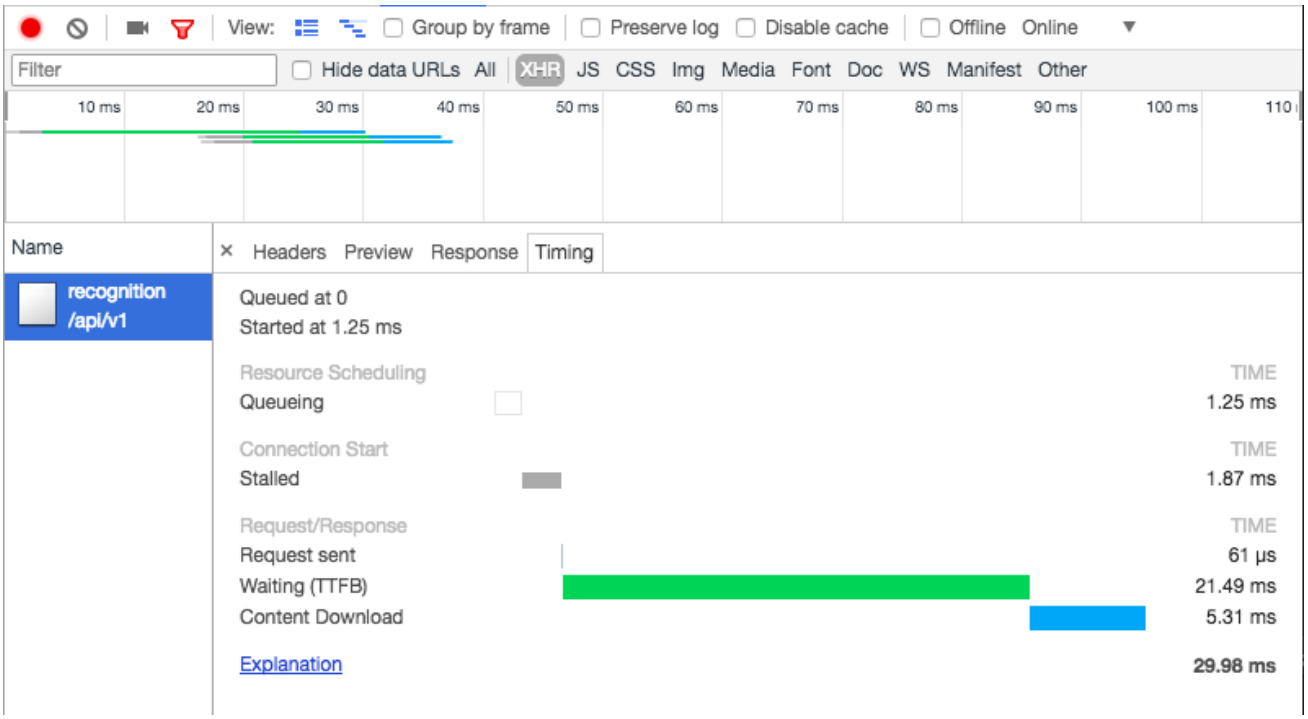

Рисунок 3.8 – Швидкість роботи класифікатора

Швидкість роботи класифікатора залежить від навантаженості і потужності сервера, а швидкість відправки та отримання відповіді з сервера від якості інтернету у користувача.

Також були зроблені графіки послідовності значень прискорення під час рухів різного типу для візуального уявлення їх характеру. (рис 3.9 – 3.11).

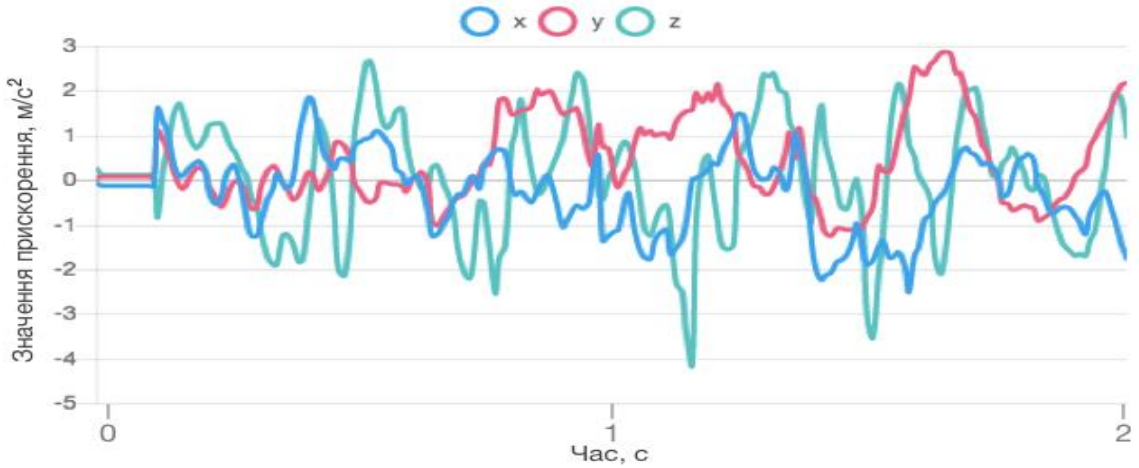

Рисунок 3.9 – Графік змін значень характеристик руху при ходьбі

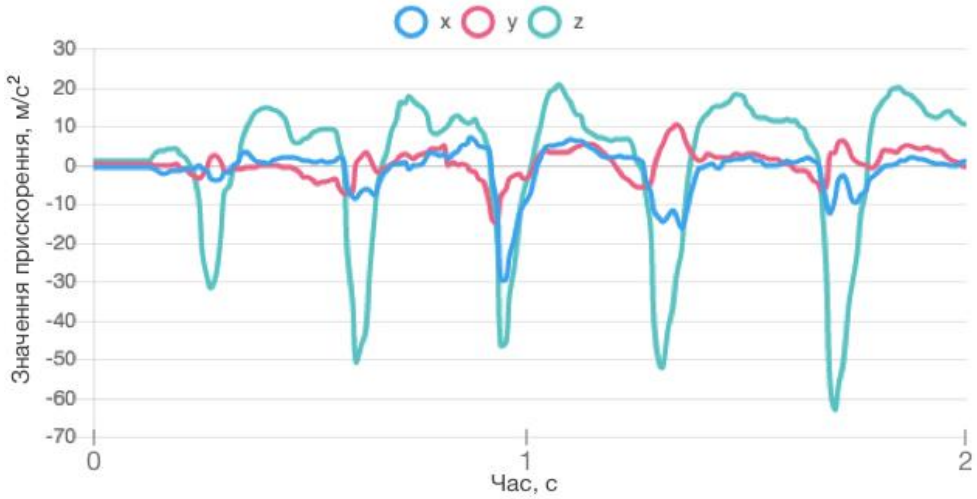

Рисунок 3.10 – Графік змін значень характеристик руху при стрибках

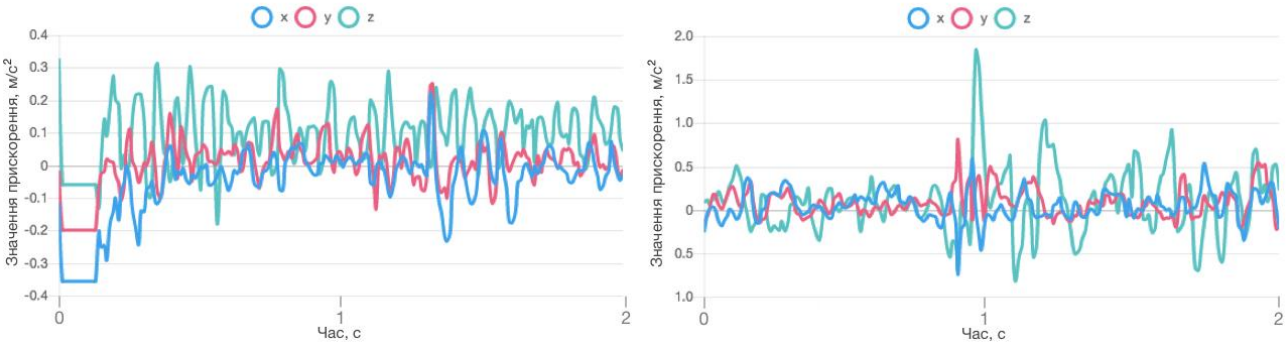

Рисунок 3.11 – Графік змін значень характеристик руху при сидінні та стоянні

3.4 Висновки до третього розділу

У третьому розділі була проведена оцінка точності запропонованого рішення. Перевірка здійснювалася на тестовій вибірці і реальних даних, і критерієм якості оцінювання була кількість правильних розпізнавань рухів в процентному співвідношенні. Був проведений аналіз швидкості роботи алгоритму, описана сфера використання пристрою та системи в цілому, визначені недоліки та напрямки подальших досліджень.

На основі проведених дослідів було доведено, що методика задовольняє вимогам щодо точності і швидкості розпізнавання.

ВИСНОВКИ

В кваліфікаційній роботі магістра були проведені розробка та дослідження методики визначення типів руху людини на основі даних з мобільного пристрою. Була досліджена актуальність проблеми визначення типів руху людини, поставлена мета розробити методику, що підвищить точність визначення в порівнянні з вже існуючими аналогами, складений перелік задач, що необхідно вирішити для досягнення мети роботи, означені об’єкт та дослідження, описана практична цінність роботи.

У першому розділі був проведен огляд наукових робіт з проблеми визначення типів руху людини. В межах огляду були проаналізовані сучасні алгоритми класифікації, сформовано список для використання в методиці. Далі був проведений огляд існуючих програмних рішень з теми визначення рухів, проаналізовано їх переваги та недоліки.

У другому розділі була проведена розробка методики визначення типів руху людини. Даний процес вимагав розгляду і перевірці алгоритмів класифікації, сформованому в першому розділі. Для використання в методиці був обраний алгоритм Random Forest, який показав найкращу точність визначення, і до того ж споживав малу кількість системних ресурсів.

У третьому розділі було проведене моделювання та створено програмно-апаратну модель системи визначення типів руху людини з використанням запропонованої методики. Була проведена оцінка точності методики за допомогою серії тестів на тестовій вибірці і реальних даних. Був проведений аналіз швидкості роботи алгоритму, описана сфера використання додатку та методики, описані недоліки розробленої методики. Дослідження відобразили, що середня точність становить 93,6%, що є хорошим показником, кращим ніж точність на даний момент у існуючих методик.

По результатах роботи була підготовлена наукова стаття .

ПЕРЕЛІК ПОСИЛАНЬ

1. The State of Obesity [Електронний ресурс] / The State of Childhood Obesity – стаття. Режим доступу: https://stateofobesity.org/childhood-obesity-trends/

2. MDPI (Multidisciplinary Digital Publishing Institute) [Електронний ресурс] / Human Movement Recognition Based on the Stochastic Characterisation of Acceleration Data – стаття. Режим доступу: http://www.mdpi.com/1424-8220/16/9/1464

3. University of Porto [Електронний ресурс] / Smartphone Gesture Learning – стаття. Режим доступу: https://sigarra.up.pt/fcup/pt/pub_geral .show_file?pi_gdoc_id=324668

4. Роццеті М. On the Design and Player Satisfaction Evaluation of an Immersive Gestural Game: the Case of Tortellino X-Perience at the Shanghai World Expo / М. Роццеті, А. Самераро, Г. Марфіа – Італія: ACM, 2011. – 87 с.

5. Раві Н. Activity Recognition from Accelerometer Data / Н. Раві, Н. Дандекар, П. Мусоре – США: PA, 2005. – 120 с.

6. Medium Online Publishing Platform [Електронний ресурс] / Visualizing the progress of machine learning – стаття. Режим доступу: https://medium.com/enrique-dans/visualizing-the-progress-of-machine-learning-154ff0f7ad13

7. International Journal of Innovative Research in Computer and Communication Engineering [Електронний ресурс] / An Overview of Knowledge Discovery Database and Data mining Techniques – стаття. Режим доступу: http://www.rroij.com/open-access/an-overview-of-knowledge-discovery-databaseand-data-mining-techniques.php?aid=48833

8. Scalable Data Science [Електронний ресурс] / Activity Recognition from Accelerometer using Decision Tree – стаття. Режим доступу: https://lamastex.github.io/scalable-data-science/sds/2/2/db/021_recognizeActivity ByDecisionTree/

9. Angular. One framework [Електронний ресурс] / What is Angular? – стаття. Режим доступу: https://angular.io/docs

10. Medium Online Publishing Platform [Електронний ресурс] / Angular Introduction to Reactive Extensions (RxJS) – стаття. Режим доступу: https://medium.com/google-developer-experts/angular-introduction-to-reactive-extensions-rxjs-a86a7430a61f

11. Medium Online Publishing Platform [Електронний ресурс] / A Quick Introduction to K-Nearest Neighbors Algorithm – стаття. Режим доступу: https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

12. Apply Machine Learning Algorithms [Електронний ресурс] / Naive Bayes for Machine Learning – стаття. Режим доступу: https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

13. Марченко О. О., Россада Т. В. Актуальні проблеми Data Mining: Навчальний посібник для студентів факультету комп’ютернихнаук та кібернетики. / Марченко О. О., Россада Т. В. –Київ. –2017. –150 с.

14. Towards Data Science [Електронний ресурс] / Decision Trees in Machine Learning – стаття. Режим доступу: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

15. Останкова Л. А. Теорія прийняття рішень: посібник для студ. техн. та екон. спеціальностей / Л. А. Останкова, Н. Ю. Шевченко, К. М. Бабенко. – Краматорськ : ДДМА, 2011. – 124 с.

16. The University of Vermont [Електронний ресурс] / Top 10 algorithms in data mining – стаття. Режим доступу: http://www.cs.uvm.edu/~icdm/algorithms/10Algorithms-08.pdf

17. Analytics Vidhya [Електронний ресурс] / Introduction to Random forest – стаття. Режим доступу: https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/

18. Брейман Л. Classification and Regression Trees / Л. Брейман, Х. Фрідман, Р. Олшен, С. Стоун – Нью-Йорк: Чапман и Халл, 1993. – 101 с.

19. Анализ малых данных [Електронний ресурс] / Случайный лес (Random Forest) – стаття. Режим доступу: https://dyakonov.org/2016/11/14/случайный-лес-random-forest/

20. Temple University Computer Services [Електронний ресурс] / Stochastic Gradient Boodting – стаття. Режим доступу: https://astro.temple.edu/~msobel/courses_files/StochasticBoosting(gradient).pdf

21. Rafael Irizarry Own Site [Електронний ресурс] / Classification Algorithms and Regression Tree – стаття. Режим доступу: https://rafalab.github.io/pages/649/section-11.pdf

ДОДАТОК А

ПРОГРАМНИЙ КОД МОДЕЛІ

```
import { Component } from '@angular/core';
import { HttpErrorResponse } from '@angular/common/http';
import { Router } from '@angular/router';

import AuthService from './Auth.service';
import UserService from '../services/User.service';
import ErrorCodeList from '../config/ErrorCodeList';
import LocalStorage from '../helpers/LocalStorage';
import IUser from '../models/IUser';
import User from '../models/User';
import IAuthResponse from '../models/IAuthResponse';

@Component({
   selector: 'cmp-signin',
   templateUrl: './signin.component.html',
})
class SignInComponent {

   public user: IUser;
   private __router: Router;
   private __authService: AuthService;
   private __userService: UserService;

   public constructor(router: Router, authService: AuthService, userService: UserService) {
      this.user = new User(Olegsey, 'qwerty');
      this.__router = router;
      this.__authService = authService;
      this.__userService = userService;
   }

   public submit(): void {
      this.__authService.signIn(this.user).subscribe(this._submitHandler, this._submitCatch);
   }

   protected _submitHandler = (response: IAuthResponse): void => {
      LocalStorage.setItem('token', response.data.token);
      this.__userService.setUser(response.data.user);
      this.__router.navigate(['/admin/training']);
   }

   protected _submitCatch = (err: HttpErrorResponse): void => {
      const errorCode = err.error.code;

      switch (errorCode) {
         case ErrorCodeList.USER_NOT_FOUND: {
            break;
         }
```

```
            case ErrorCodeList.INCORRECT_PASSWORD: {
                break;
            }
            default: {

            }
        }
    }
}

export default SignInComponent;

import { Component } from '@angular/core';
import { HttpErrorResponse } from '@angular/common/http';

import MovementTypeList from '../../config/MovementTypeList';
import MotionCaptureService from '../../services/MotionCapture.service';
import MLService from '../../services/ML.service';
import DataService from '../../services/Data.service';
import TimerService from '../../services/Timer.service';
import TrainingStateList from '../../config/TrainingStateList';
import TrainingResponseList from '../../config/TrainingResponseList';
import IDictionary from '../../models/IDictionary';
import ISuccessResponse from '../../models/ISuccessResponse';
import ITrainData from '../../models/ITrainData';
import ThemeService from '../../services/Theme.service';

@Component({
    selector: 'cmp-training',
    templateUrl: './adminTraining.component.html',
})
class AdminTrainingComponent {

    public ticks: number;
    public MovementTypeList: MovementTypeList;
    public TrainingStateList: TrainingStateList;
    public TrainingResponseList: TrainingResponseList;
    public trainData: ITrainData[];
    public movementType: string;
    public trainingState: number;
    public defaultMovementType: string;
    public currentTheme: string;
    public response: IDictionary<string | number>;

    private __mlService: MLService;
    private __dataService: DataService;
    private __motionCaptureService: MotionCaptureService;
    private __timerService: TimerService;
    private __themeService: ThemeService;

    public constructor(mlService: MLService,
                       dataService: DataService,
```

```
                motionCaptureService: MotionCaptureService,
                timerService: TimerService,
                themeService: ThemeService) {
    this.__mlService = mlService;
    this.__dataService = dataService;
    this.__motionCaptureService = motionCaptureService;
    this.__timerService = timerService;
    this.__themeService = themeService;

    this.MovementTypeList = MovementTypeList;
    this.TrainingStateList = TrainingStateList;
    this.TrainingResponseList = TrainingResponseList;
    this.defaultMovementType = MovementTypeList.NOT_SELECTED;
    this.trainingState = TrainingStateList.NOT_ACTIVE;
    this.ticks = 0;
    this.response = {};

    this.__themeService.getThemeObservable().subscribe((theme: string) => {
        this.currentTheme = theme;
    });
}

public setMovementType(movementType: keyof typeof MovementTypeList): void {
    this.movementType = movementType;
}

public captureMovement(): void {
    if (this.movementType !== void 0) {
        this.__timerService.start((ticks: number) => {
            this.ticks = ticks;
        });

        this.__motionCaptureService.setMovementType(this.movementType);
        this.__motionCaptureService.capture();
        this.response = {};
        this.trainingState = TrainingStateList.CAPTURING;
    }
}

public stop(): void {
    this.__timerService.stop();
    this.__motionCaptureService.stopCapture();
    this.trainData = this.__motionCaptureService.getData();
    this.trainingState = TrainingStateList.STOPPED;
}

public removeData(): void {
    this.ticks = 0;
    this.trainData = [];
    this.trainingState = TrainingStateList.NOT_ACTIVE;
    this.__motionCaptureService.removeData();
}
```

```
    public send(): void {
        if (this.trainData.length > 0) {
            this.__dataService
                .sendTrainData(this.trainData)
                .subscribe((response: ISuccessResponse<IDictionary<any>>): void => {
                    this.trainingState = TrainingStateList.NOT_ACTIVE;
                    this.response = {
                        data: response.data.insertedCount,
                        type: TrainingResponseList.DATA,
                    };
                }, (error: HttpErrorResponse): void => {});
        }
    }

    public train(): void {
        this.__mlService
            .train()
            .subscribe((response: any): void => {
                console.log(response);
            }, (error: HttpErrorResponse): void => {});
    }

    public getPathToIcon(icon: string): string {
        return '../../../assets/images/training/movements/' + this.currentTheme + '-' + icon + '.svg';
    }
}

export default AdminTrainingComponent;

import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import IDictionary from '../../models/IDictionary';

@Component({
    selector: 'cmp-data',
    templateUrl: './adminChartData.component.html',
})
class AdminChartDataComponent implements OnInit {

    private static readonly COLOR_TRANSPARENT: string = 'rgba(0,0,0,0)';
    private static readonly COLOR_BLUE: string = 'rgb(54, 162, 235)';
    private static readonly COLOR_RED: string = 'rgb(255, 99, 132)';
    private static readonly COLOR_GREEN: string = 'rgb(75, 192, 192)';
    private static readonly CHART_TYPE: string = 'line';
    private static readonly LABELS_SIZE: number = 200;

    public chartType: string;
    public chartLabels: number[];
    public chartOptions: IDictionary<any>;
    public chartColors: IDictionary<string>[];
    public chartData: IDictionary<object | string>[];
```

```
    private __route: ActivatedRoute;

    public constructor(route: ActivatedRoute) {
        this.__route = route;

        this.chartType = AdminChartDataComponent.CHART_TYPE;
        this.chartOptions = {
            responsive: true,
            scales: {xAxes: [{display: false}]},
            legend: {labels: {usePointStyle: true}},
        };
        this.chartColors = [
            {
                backgroundColor: AdminChartDataComponent.COLOR_TRANSPARENT,
                borderColor: AdminChartDataComponent.COLOR_BLUE,
            },
];
        this.chartLabels = [...Array(AdminChartDataComponent.LABELS_SIZE).keys()];
    }

    public ngOnInit(): void {
        this.__route.data.subscribe((response: IDictionary<object>) => {
            this.chartData = response.chartData as IDictionary<object | string>[];
        });
    }
}

export default AdminChartDataComponent;
```

```
import math
import operator

class KNN( ):
        def __init__(self, train, test, k):
                self.train = train
                self.test = test
                self.k = k
                self.predictions = []

        def predict(self):
                for x in range(len(self.test)):
                        aux = neighbors(self.train, self.test[x], self.k)
                        result = sumSmiles(aux)
                        self.predictions.append(result)

        def accuracy(self):
                achunta = 0
                for x in range(len(self.test)):
                        if self.test[x][-1] == self.predictions[x]:
                                achunta += 1
                return (achunta/float(len(self.test))) * 100.0
```

```
def distance(a, b, length):
	distancia = 0
	for x in range(length):
		distancia += pow((float(a[x]) - float(b[x])), 2)
	return math.sqrt(distancia)

def neighbors(train, instancia, k):
	length = len(instancia)-1

	distancias = [(train[x], distance(instancia, train[x], length)) for x in range(len(train))]

	distancias.sort(key=operator.itemgetter(1))
	return [distancias[x][0] for x in range(k)]

def sumSmiles(neighbors):
	clases = {}
	for x in range(len(neighbors)):
		c = neighbors[x][-1]
		if c in clases:
			clases[c] += 1
		else:
			clases[c] = 1

	return sorted(clases.items(), key=operator.itemgetter(1), reverse=True)[0][0]

from sfloresLib import *
from knn import *
import numpy as np
import argparse

#######  Functions  #######
def GUI( ):
	ventana = createWindow()
	ventana.setTitle("KNN")
	ventana.setSize(600,600)
	ventana.setColor("white")
	ventana.setCloseConfirm()
	ventana.createButton("Browse" , browse)
	return ventana

def browse( ):
	file_name = ventana.askopenfile((("CSV", "*.csv"),("All files", "*.*")))
	test(file_name, True, kMax, sMax)

def readFile(filename, delimiter):
	aux = open(filename).read()
	aux = [item.split(delimiter) for item in aux.split('\n')[:]]
	return aux

def test(database, plot, kNum, sMax):
	ks = []
	accuracies = []
```

```
        #leemos la base de datos
        iris = readFile(database, ",")
        examples = len(iris)
        columns = len(iris[0])

        if(kNum>examples or kNum<1):
                quit()
        if(sMax<1 or sMax>examples):
                quit()
        indexes = np.random.permutation(examples)
        iris = np.asarray(iris)[indexes].tolist()

        elemSeg =  len(iris)/sMax
        segmentos = [iris[int(i*elemSeg):int(i*elemSeg+elemSeg)] for i in range(sMax)]

        for k in range(1,kNum+1):
           accuracy = 0
           for i in range(sMax):

                index = np.delete([x for x in range(sMax)], i)
                train = np.concatenate(np.asarray(segmentos)[index].tolist())
                test = segmentos[i]

                knn = KNN(train,test,k) #train data, test data, k value
                knn.predict()
                accuracy += knn.accuracy()
           accuracy /= sMax
           print("Accuracy: " , accuracy, "k: " , k)
           if(plot):
                ks.append(k);
                accuracies.append(accuracy)

        if(plot):
                ventana.deleteAllButton()
                ventana.graphPlot(ks,accuracies,"k's","Accuracy","Accuracy Graph")


#######  Main Program  #######
if __name__ == '__main__':
   parser = argparse.ArgumentParser()
   parser.add_argument('-f', metavar='filename', action="store", dest="file", help='database file path
name.')
   parser.add_argument('-k', metavar='k', action="store", dest="k", help='max k number to test (Default
= 10)')
   parser.add_argument('-s', metavar='splits', action="store", dest="s", help='number of segments to
validate (Default = 10)')
   args = parser.parse_args()
   kMax = 10
   if(args.k != None):
        kMax= int(args.k)
```

```
    sMax = 10
    if(args.s != None):
        sMax= int(args.s)

    if(args.file == None):
        ventana = GUI()
        ventana.keepAlive()
    else:
        test(args.file, False, kMax, sMax)


#!/usr/bin/python

import random;
import sys;
import csv;

class NaiveBayes:

        def __init__(self, n):

                self.data = [];#the data
                self.dim = n;#dimension of the data
                self.values = {};#possible values
                self.totals = {};#number of times a value appears
                self.counts = [{} for i in range(n)];#number of feature value appearences by index and
label value
                self.possible = [set() for i in range(n)];#range possible values for each feature

        def getData(self):
                return self.data;

        def getDim(self):
                return self.dim;

        """
        Adds a list of data points to the classifier.
        data a list of tuples consisting of data point, label such that
                        len(data point) = self.dim
        """
        def addData(self, data):

                for d in data:
                        if len(d[0]) != self.dim:
                                return False;

                self.data += data;

                for d in data:

                        if d[1] in self.values:
                                self.values[d[1]] += 1;
```

```
                else: self.values[d[1]] = 1;

        return True;

def train(self):

        for d in self.data:

                e = d[0];  # example
                v = d[1]; # value

                # update totals for this value
                if v in self.totals:
                        self.totals[v] += 1;
                else: self.totals[v] = 1;

                # go through the fields of the example
                for i in range(self.dim):

                        if v not in self.counts[i]:
                                self.counts[i][v] = {};

                        f = e[i];
                        self.possible[i].add(f);

                        # update the count for this value of i with label v
                        if f in self.counts[i][v]:
                                self.counts[i][v][f] += 1;
                        else: self.counts[i][v][f] = 1;

def predict(self, obj):

        if len(obj) != self.dim:
                raise ValueError('Unclassifiable object. Wrong feature space.');

        newData = {};
        ps = [];

        for i in range(self.dim):
                if obj[i] not in self.possible[i]:
                        newData[i] = obj[i];

        for v in self.values:
                p = float(self.values[v])/len(self.data); # probability of label v
                i = 0;
                while p > 0.0 and i < self.dim:
                        if i not in newData: # can't predict based on things not trained on
                                f = obj[i];
                                if f in self.counts[i][v]: # f has been seen at position i with value v
                                        p *= float(self.counts[i][v][f])/self.totals[v];
                                else:
```

```
                                        p = 0.0;
                                i += 1;
                        ps.append((p, v));

                return max(ps, key=lambda x: x[0]), newData;

def test():

        # read training data
        examples = [];
        print "\nTraining...\n"
        sheetReader = csv.reader(open('training.csv', 'r'));
        for example in sheetReader:
                for i in range(len(example)):
                        example[i] = example[i].strip();
                v = example.pop();
                if example[0]:
                        examples.append((example, v));
                        print example, v;


        # read testing data
        tests = [];
        print "\nTesting...\n";
        sheetReader = csv.reader(open('testing.csv', 'r'));
        for row in sheetReader:
                for i in range(len(row)):
                        row[i] = row[i].strip();
                v = row.pop();
                if row[0]:
                        tests.append((row, v));
                        print row, v;


        nb = NaiveBayes(len(examples[0][0]));
        nb.addData(examples);
        nb.train();

        d = len(tests); # number of test examples
        n = 0; # number correct

        for e in tests:

                test = nb.predict(e[0]);
                if test[0][1] == e[1]: n += 1;

        print 'percent correct:', float(n)/d;
```

ДОДАТОК Б

ПРОГРАМНИЙ КОД КЛІЄНТУ ТА СЕРВЕРУ

Код серверу:

```
mport socket
import numpy as np
import encodings

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 65432        # Port to listen on (non-privileged ports are > 1023)

def random_data():

    x1 = np.random.randint(0, 55, None)        # Dummy temperature
    y1 = np.random.randint(0, 45, None)        # Dummy humidigy
    my_sensor = "{},{}".format(x1,y1)
    return my_sensor                           # return data seperated by comma

def my_server():

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        print("Server Started waiting for client to connect ")
        s.bind((HOST, PORT))
        s.listen(5)
        conn, addr = s.accept()

        with conn:
            print('Connected by', addr)
            while True:

    data = conn.recv(1024).decode('utf-8')

    if str(data) == "Data":

    print("Ok Sending data ")

     my_data = random_data()

      x_encoded_data = my_data.encode('utf-8')

      conn.sendall(x_encoded_data)

      elif  str(data) == "Quit":
      print("shutting down server ")
                break


       if not data:
                 break
       else:
```

```
            pass


if __name__ == '__main__':
    while 1:
        my_server()
```

Код клієнту:

```
import socket
import threading
import time


HOST = '192.168.0.111'  # The server's hostname or IP address
PORT = 65432        # The port used by the server


def process_data_from_server(x):
    x1, y1 = x.split(",")
    return x1,y1


def my_client():
    threading.Timer(11, my_client).start()

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))

        my = input("Enter command ")

        #my = "Data"

        my_inp = my.encode('utf-8')

        s.sendall(my_inp)

        data = s.recv(1024).decode('utf-8')

        x_temperature,y_humidity = process_data_from_server(data)

        print("Temperature {}".format(x_temperature))
        print("Humidity {}".format(y_humidity))

        s.close()
        time.sleep(5)


if __name__ == "__main__":
    while 1:
        my_client()
```