НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

"___"_____2016 р.

Дипломна робота

__першого (бакалаврського)__ **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Прогнозування місцезнаходження заданих об'єктів в середовищі контекстно-залежних систем

Виконав (-ла): студент (-ка) __4__ курсу, групи _ДА-21_
(шифр групи)

Шаптала Роман Віталійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н. Кисельова А.Г. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічний __проф., д.е.н. Семенченко Н.В.___ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доцент, к.т.н. Хижняк Т.А. _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль ст. викладач Британов О.А.

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

«___»_____2016 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Шапталі Роману Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)) Прогнозування місцезнаходження заданих об'єктів в середовищі контекстно-залежних систем
керівник проекту (роботи)) Кисельова Анна Геннадіївна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст

2. Строк подання студентом проекту (роботи) 10.06.2016

3. Вихідні дані до проекту (роботи) _____

Методи та алгоритми прогнозування місцезнаходження: Нейронні Мережі, Дерева Рішень, Прихована Марковська Модель, Метод Голосування, Наївний Баєсовий Класифікатор, Метод Опорних Векторів, Випадкові Ліси, Метод Стимулювання Градієнту.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Прогнозування місцезнаходження у контекстно-залежних системах.
2. Розробка алгоритмів прогнозування місцезнаходження в контекстно-залежних системах.
3. Огляд результатів тестування розроблених алгоритмів.
4. Функціонально-вартісний аналіз програмного продукту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Прогнозування місцезнаходження заданих об'єктів в середовищі контекстно-залежних систем управління електроспоживанням – плакат.
2. Дослідження методів та алгоритмів прогнозування місцезнаходження заданих об'єктів в контекстно-залежних системах управління електроспоживанням – плакат.
3. Результати тестування досліджених методів та алгоритмів – плакат.
4. Збільшення точності прогнозування за допомогою підбору гіперпараметрів проаналізованих методів та алгоритмів – плакат.

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Семенченко Н.В., проф., д.е.н.		

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Дослідження предметної області та існуючих рішень.	28.02.2016	
4	Дослідження алгоритмів прогнозування місцезнаходження	10.03.2016	
5	Перевірка точності прогнозу обраних методів алгоритмів	15.03.2016	
6	Розробка алгоритму підбору гіперпараметрів	25.03.2016	

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

7	Виконання програмної реалізації алгоритмів	25.04.2016	
8	Перевірка точності обраних алгоритмів з підбором гіперпараметрів	30.04.2016	
9	Оформлення дипломної роботи	31.05.2016	
10	Отримання допуску до захисту та подача роботи в ДЕК	10.06.2016	

Студент _____ Р.В. Шаптала
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ А.Г. Кисельова
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

бакалаврської дипломної роботи Шаптали Романа Віталійовича на тему «Прогнозування місцезнаходження заданих об’єктів в середовищі контекстно-залежних систем»

Метою роботи є підбір алгоритму для найбільш точного прогнозування місцезнаходження об’єкту в середовищі контекстно-залежних систем. В роботі були проаналізовані наступні алгоритми: нейронні мережі, дерева рішень, наївний баєсів класифікатор, випадкові ліси, метод опорних векторів, прихована марковська модель, стимулювання градієнта та голосування. Було проведено налаштування гіперпараметрів даних алгоритмів, та визначено, що найбільшу точність на класичному наборі даних показали алгоритми випадкових лісів та дерев рішень.

Загальний обсяг роботи: 70 сторінок, 14 ілюстрацій, 13 таблиць, 1 додаток на 8 сторінках та 25 посилань.

Ключові слова: прогнозування місцезнаходження, контекстно-залежна система, управління електроспоживанням, нейронні мережі, дерева рішень, наївний баєсів класифікатор, випадкові ліси, метод опорних векторів, прихована марковська модель, метод стимулювання градієнта.

АННОТАЦИЯ

бакалаврской дипломной работы Шапталы Романа Витальевича на тему «Прогнозирование местонахождения заданных объектов в среде контекстно-зависимых систем»

Целью работы является подбор алгоритма для наиболее точного прогнозирования местонахождения объекта в среде контекстно-зависимых систем. В работе были проанализированы следующие алгоритмы: нейронные сети, деревья решений, наивный байесовский классификатор, случайные леса, метод опорных векторов, скрытая марковская модель, стимулирование градиента и голосование. Было проведено подбор гиперпараметров данных алгоритмов, и определено, что наибольшую точность на классическом наборе данных показали методы случайных лесов и деревьев решений.

Общий объем работы: 70 страниц, 14 иллюстраций, 13 таблиц, 1 приложение на 8 страницах и 25 ссылок.

Ключевые слова: прогнозирование местонахождения, контекстно-зависимая система, управление энергопотреблением, нейронные сети, деревья решений, наивный байесовский классификатор, случайные леса, метод опорных векторов, скрытая марковская модель, метод стимулирования градиента.

ABSTRACT

to the bachelor thesis by Roman Shaptala on " Location prediction for context-aware energy management systems "

The goal of the thesis is to select an algorithm with the best accuracy of location prediction of the object in a context-aware systems environment. The following algorithms were tested: neural networks, decision trees, naive bayes classifier, random forests, support vector machines, hidden markov model, gradient boosting and voting. Each method’s hyperparameters were tuned, and the result shows that the highest accuracy on the classic dataset was achieved by random forests and decision trees.

Includes 70 pages, 14 illustrations, 13 tables, 1 appendix with 8 pages and 25 references.

Keywords: location prediction, context-aware system, energy management, neural networks, decision trees, naive bayes classifier, random forests, support vector machines, hidden markov model, gradient boosting.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

C4.5	Алгоритм для побудови дерев рішень, який є вдосконаленою версією алгоритму ID3. Зокрема, в нову версію були додані відсікання гілок, можливість роботи з числовими атрибутами, а також можливість побудови дерева з неповної навчальної вибірки, в якій відсутні значення деяких атрибутів.
CART	Узагальнюючий термін, використовуваний для позначення дерев для класифікації та регресійних дерев.
CHAID	Різновид методу дерева рішень, заснований на скоригованій оцінці статистичної значущості. CHAID може бути використаний для прогнозування, а також класифікації і виявлення взаємодії між змінними.
GPS	Система глобального позиціонування
ID3	Алгоритм, що використовується для побудови дерева рішень з набору даних. ID3 є попередником алгоритму C4.5, і, як правило, використовується в області машинного навчання і областей обробки природної мови.
RFID	Радіочастотна ідентифікація
UNIX	Операційна система, в якій прийнятий відлік часу та система опису моментів в часі зручні для машинного сприйняття.
Wi-Fi	Загальновживана назва для стандарту IEEE 802.11 передачі цифрових потоків даних по радіоканалах
АЕТПМ	Аугсбурзький Еталонний Тест Прогнозування Місцезнаходження
ВЛ	Випадкові ліси
Г	Метод голосування між моделями випадкових лісів, опорних векторів та наївного баєсового класифікатора

ДР	Дерева рішень
КЗСУЕ	Контекстно-залежна система управління електроспоживанням
МОВ	Метод опорних векторів
МСГ	Метод стимулювання градієнту
НБК	Наївний баєсовий класифікатор
НМ	Нейронні мережі
ПММ	Приховані марковські моделі
РБФ	Радіальна базисна функція

ВСТУП

Контекстно-залежна система – інформаційна система, яка здатна отримувати та використовувати інформацію з різних джерел з метою адаптації під поточну ситуацію та прийняття відповідних рішень.

Актуальність теми дослідження полягає в тому, що сучасний стан контекстно-залежних систем вимагає аналізу точних і якісних методів прогнозування одного з контекстних параметрів, а саме місцезнаходження заданих об’єктів, що дозволить забезпечити вирішення задачі ефективного керування процесами генерації, трансформації, накопичення та споживання електричної енергії в єдиному інформаційному середовищі, з метою мінімізації електроспоживання та максимізації комфорту користувача.

Метою дипломної роботи є аналіз та порівняння методів прогнозування місцезнаходження заданих об’єктів в середовищі контекстно-залежних систем. Поставлена мета вимагає вирішення наступних наукових задач:

1) аналіз та прогнозування параметру контексту – місцезнаходження об’єктів;

2) збільшення точності прогнозування через підбір гіперпараметрів проаналізованих алгоритмів.

Об’єктом дослідження є процес прогнозування контекстного параметру місцезнаходження. Предметом дослідження є контекстно-залежна система.

Досягнення поставленої мети реалізовано з використанням методів інтелектуального аналізу даних та машинного навчання, в тому числі алгоритмів обробки часових рядів.

Наукова новизна дпиломної роботи полягає в тому, що було запропоновано та відлагоджено алгоритм, що досягає найкращої точності для поставленої задачі, а саме прогнозує місцезнаходження певного об'єкту в наступний часовий проміжок з найбільшим відсотком співпадінь з реальними даними на тестовій вибірці. Також запропоновано використання методів голосування між декількома алгоритмами для збільшення точності

прогнозування, що дозволило поєднати сильні сторони проаналізованих алгоритмів;

Потенційні застосування та практична цінність результатів дипломної роботи:

1) методи та алгоритми запропоновані в даній роботі можуть бути використані як частина контекстно-залежної системи управління електроспоживанням для прогнозування місцезнаходження об'єктів;

2) сформульовано рекомендації з підбору правильних гіперпараметрів для прогнозування параметрів контексту, за допомогою яких можна підняти точність таких прогнозів ще на 1-2%.

1 ПРОГНОЗУВАННЯ МІСЦЕЗНАХОДЖЕННЯ У КОНТЕКСТНО-ЗАЛЕЖНИХ СИСТЕМАХ

1.1 ОПИС КОНТЕКСТНО-ЗАЛЕЖНИХ СИСТЕМ

В останні роки, контекстно-залежні системи є предметом зростаючої уваги в області розподілених обчислень через їх корисність в декількох різних видах застосувань[1]. Контекстно-залежна система управління електроспоживанням (КЗСУЕ) є інтелектуальною інформаційною системою, яка інтегрує інформацію з різних гетерогенних джерел, таких як альтернативні джерела живлення, датчики, які характеризуються різними типами фізичних даних і навантажень та дозволяють вирішувати важливі завдання користувача в різноманітних місцях або зонах (будівлі або групи будівель). Основна мета КЗСУЕ полягає в створенні системи управління, яка буде в змозі розумно передбачати і реагувати на дії всіх електричних приладів (навантажень і генераторів), підключених до неї в єдиному інформаційному середовищі. Ця системна інтеграція дозволяє як контролювати енергетичні ресурси, так і приймати до уваги вимоги користувачів облікового запису[2].

1.2 ПАРАМЕТРИ КОНТЕКСТУ

Фактори системи управління, що впливають на прийняття рішень, називаються параметрами контексту[3]. Контекст може бути визначений як будь-яка інформація, яка може бути використана для характеристики становища суб'єкта, де суб'єктом може бути користувач, його місцезнаходження, інфраструктура, час, середовище або об'єкт, який напряму має відношення до взаємодії між користувачем і КЗСУЕ, в тому числі і сам користувач[4] (рис. 1).

Інтеграція інформації, отриманої з різнорідних джерел, в контекст створює модель поточного стану реального об'єкта, а також зменшує обсяг даних, що підлягають обробці, на підставі яких алгоритми управління можуть генерувати численні рішення управління.

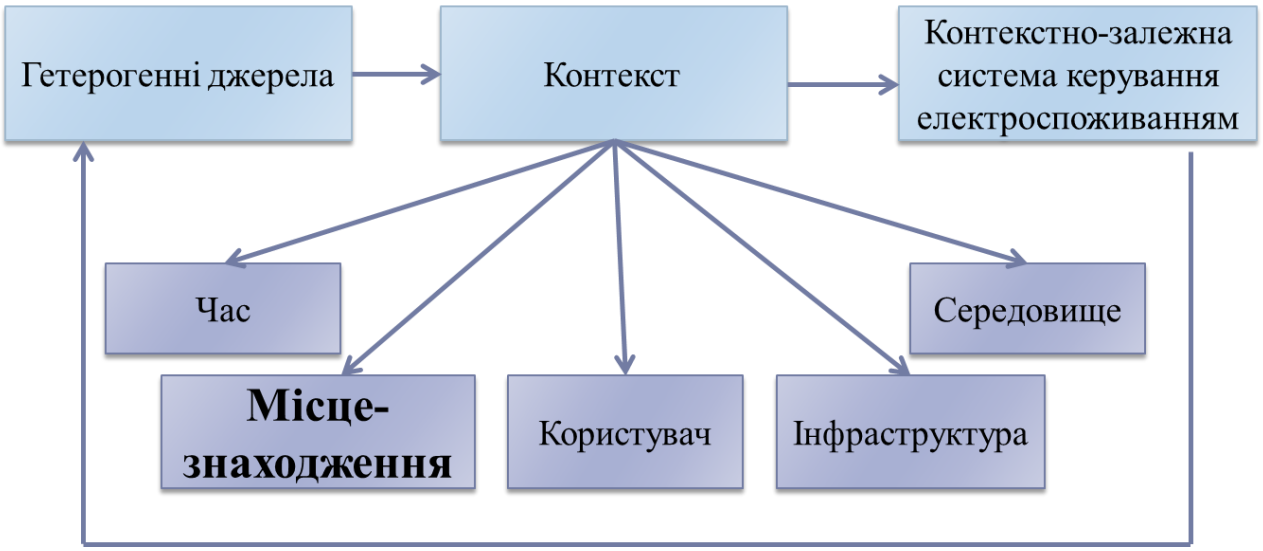

Рис. 1 – Структура контексту

Такий тип систем в значній мірі залежить від контекстної обізнаності та методів прогнозування контексту, які використовують відповідну інформацію користувача.

1.3 ОПИС ЗАВДАННЯ ПРОГНОЗУВАННЯ МІСЦЕЗНАХОДЖЕННЯ

Завдання прогнозування місцезнаходження має на меті охарактеризувати майбутній елемент контексту користувача на основі аналізу історії спостережуваного контексту, ряду контекстної інформації, яка показує, як користувачі пересуваються навколо певного обчислюваного середовища[5]. Класичне визначення завдання прогнозування місцезнаходження об’єктів має на вході набір відвіданих місць (кімнат, будівель) протягом певної кількості часових кроків, які називаються часовими мітками. Дані для прогнозування місцезнаходження, як правило, мають такий вигляд: <часова мітка Unix, об'єкт, місцезнаходження>. Приклад таких даних показаний у таблиці 1. З огляду на ряд попередніх візитів, метою прогнозування місцезнаходження є знаходження найбільш ймовірного місця перебування об’єкта для наступної часової мітки. Прогнозування місцезнаходження об’єкту в замкнутих системах обмежує

різноманіття об’єктів моделювання до приміщень або офісів всередині будівель чи споруд.

Таблиця 1 – Приклад даних

Часова мітка Unix	Об’єкт	Місцезнаходження
1462101290	Менеджер	Кухня
1462104028	Прибиральниця	Коридор

Крім класичного завдання, за наявності інших джерел даних, можна розширити проблему до такої, де контекстна інформація надається різними типами датчиків, такими як GPS, RFID і бездротовими пристроями, які можуть надати контекстну інформацію про місце, дію або зміни стану користувачів в фізичному середовищі[6].

1.4 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

За останнє десятиліття було опубліковано значну кількість статей і наукових робіт про алгоритми прогнозування розташування всередині приміщень в ряді областей застосування, в тому числі офісних[7] і університетських[8] будівель. Крім того, було запропоновано два сценарії застосунку таких методів: переадресація телефонного виклику до кімнати, у яку входить людина[8], і проект розумної дверної таблички[9], який вирішує, чи відвідувач повинен чекати шукану людину в своєму кабінеті, чи повинен перейти до передбаченого місця призначення даної людини. Дана функціональність спирається на застосування RFID тегів, які ідентифікують людину в області до кількох метрів. Стаття [8] показує досягнення 57% точності у класичній постановці. У роботі [9] зазначається необхідність в адаптивності таких алгоритмів, через часту тимчасову або постійну зміну людських поведінкових моделей. Також існують методи пов'язані з використанням мобільних показань датчиків від Wi-Fi або GPS з наступним використанням фреймворку Наступної Локації[6] для прогнозування місцезнаходження. При цьому вони пропонують проводити процес тренування

на даних прямо на мобільному пристрої для підвищення приватності та персоналізації системи. Такий підхід є багатообіцяючим через широку наявність мобільних пристроїв.

1.5 ВИСНОВОК

Контекстно-залежні системи управління електроспоживанням оперують контекстними параметрами, тому ефективність прийняття рішень ними значно спирається на якість прогнозування факторів контексту. Використання прогнозування місцезнаходження об’єктів дозволяє контекстно-залежній системі управління електроспоживанням вибрати найбільш ефективні стратегії для досягнення поставлених цілей, таких як максимізація комфорту користувачів та мінімізація використання енергії. Наприклад, нагрівач, підключений до такої системи може почати регулювати температуру в приміщенні, щоб задовольнити потреби конкретної людини, ще до моменту її входу до кімнати. Таким чином, кімната вже буде підготовленою для користувача під час його входу, в той час як сенсорні технології, реагуючи на рухи, будуть тільки ініціювати процедуру нагрівання після того як людина зайшла.

Прогнозування місцезнаходження є актуальним завданням, яке намагаються вирішити різними методами та підходами та яке має набір власних особливостей. Попри наявність певного числа модифікацій до даної проблеми, у роботі розглядається лише класичний варіант прогнозування місцезнаходження у замкнутих системах, адже всі видозмінені завдання напряму залежать від якості вирішення стандартного. Прогнозування місцезнаходження має потенційно найбільші можливості для застосування, порівняно з іншими контекстними параметрами, тому покращення роботи методів та алгоритмів вирішення даної задачі може напряму впливати на ефективність роботи систем, які базуються на контекстній інформації про локацію, в тому числі і КЗСУЕ.

2. РОЗРОБКА АЛГОРИТМІВ ПРОГНОЗУВАННЯ МІСЦЕЗНАХОДЖЕННЯ В КОНТЕКСТНО-ЗАЛЕЖНИХ СИСТЕМАХ

2.1 ЗАГАЛЬНИЙ ОПИС МЕТОДІВ

У даній роботі проводиться оцінка кількох відомих і загальноприйнятих методів прогнозування місця розташування (нейронні мережі, наївний баєсів класифікатор, дерева рішень, метод опорних векторів, прихована марковська модель), а також деякі алгоритми, які досягли високої точності на різних завданнях машинного навчання з вчителем (випадкові ліси, збірка декількох моделей через голосування та метод стимулювання градієнта).

Використані алгоритми можуть бути розділені на дві групи(табл. 2): збірні або автономні. Збірні моделі покладаються на навчання кількох різних або однотипних моделей для досягнення кращої точності, ніж автономні алгоритми.

Таблиця 2 – Класифікація використаних алгоритмів

Автономні	**Збірні**
Нейронні Мережі	Випадкові Ліси
Наївний Баєсів Класифікатор	Голосування
Приховані Марковські Моделі	Метод Стимулювання Градієнта
Метод Опорних Векторів	
Дерева Рішень	

Структура модуля прогнозування місцезнаходження об'єктів зображена на рисунку 2. Вона показує, що сам модуль є лише частиною КЗСУЕ та обмін даними відбувається з нею через мережевий інтерфейс. При цьому забезпечується легке переконфігурування та розширення такої системи через заміну модулів з однаковим інтерфейсом.

Аналогічного типу модулі можуть використовуватись і для розширеної проблеми прогнозування місцезнаходження, лише змінивши канал передачі даних.

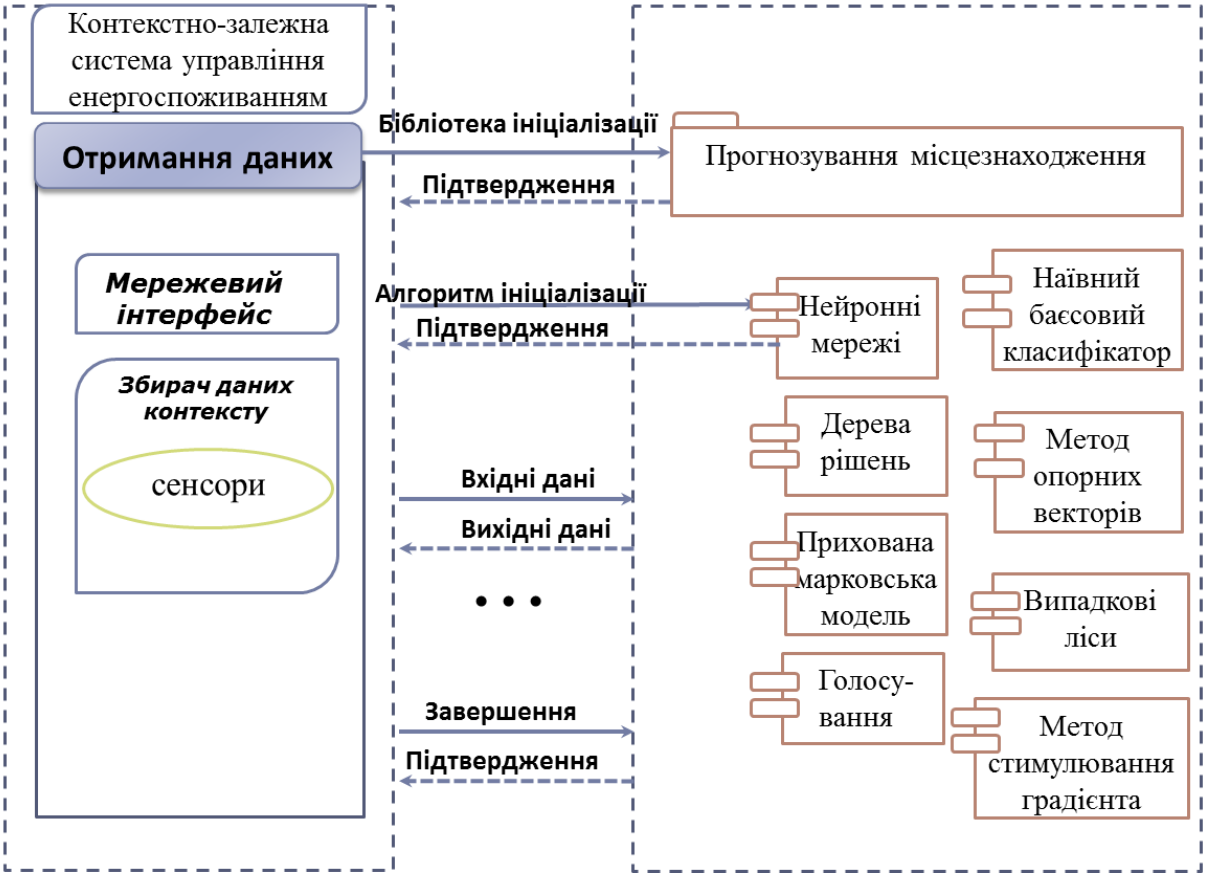

Рисунок 2 - Структура модуля прогнозування місцезнаходження

2.2 АЛГОРИТМИ ТА ПІДХОДИ

2.2.1 НЕЙРОННІ МЕРЕЖІ

Нейронні мережі — математичні моделі, а також їхня програмна імплементація, які базуються на принципах функціонування мереж нервових клітин. Архітектура і принцип роботи даного класу алгоритмів базується на подібності до мозку живих створінь. Основною складовою цих систем виступає нейрон як імітаційна модель нервової клітини мозку. З розвитком алгоритмів навчання, отримані моделі почали використовуватися для машинного навчання: в задачах прогнозування, для розпізнавання образів, в задачах класифікації[10].

Математична модель нейрона складається з зваженої суми входів та певної нелінійної функції, як показано у формулах (1), (2).

$$y = \sum_i w_i x_i, \quad (1)$$

де w_i – ваги;

x_i – вхідні значення;

y – зважена сума входів.

$$OUT = F(y), \quad (2)$$

де F – нелінійна функція активації;

OUT – вихідне значення нейрона.

Найбільш поширеними функціями активації є жорстка сходинка(3) та сигмоїда(4).

$$OUT = \begin{cases} 0, y < t \\ 1, y \geq t \end{cases}, \quad (3)$$

де t – граничний рівень.

$$OUT = \frac{1}{1+ e^{-y}}, \quad (4)$$

де e – число Ейлера.

У даній роботі була використана MAXOUT-функція активації (5) для прихованого шару із 100 нейронів, а також SOFTMAX-функція активації(6) для вихідного шару для генерації розподілу ймовірностей за прогнозованими місцями:

$$OUT = MAX(y), \quad (5)$$

де MAX – функція, яка повертає найбільше значення із входів.

$$OUT = \frac{e^y}{\sum_i e^{y_i}}. \quad (6)$$

Нейронна мережа тренувалась за допомогою алгоритму стохастичного градієнтного спуску, на кожній ітерації якого ваги змінювались за законом (7).

$$w \coloneqq w - \eta \nabla w, \quad (7)$$

де η – темп навчання;

∇w – градієнт функції втрат по параметрам.

Для дослідження темп навчання був постійним та рівним 0.01, а функцією втрат являлась середня категоріальна крос-ентропія, описана формулою (8).

$$L = \frac{1}{m}\sum_{k=1}^{m}(t_k \ln(OUT_k) + (1 - t_k)\ln(1 - OUT_k)), \quad (8)$$

де m – кількість точок для тренування;

t_k – істинний клас;

OUT_k – прогнозований клас.

Архітектура мережі, використаної в даній роботі зображена на рисунку 3.

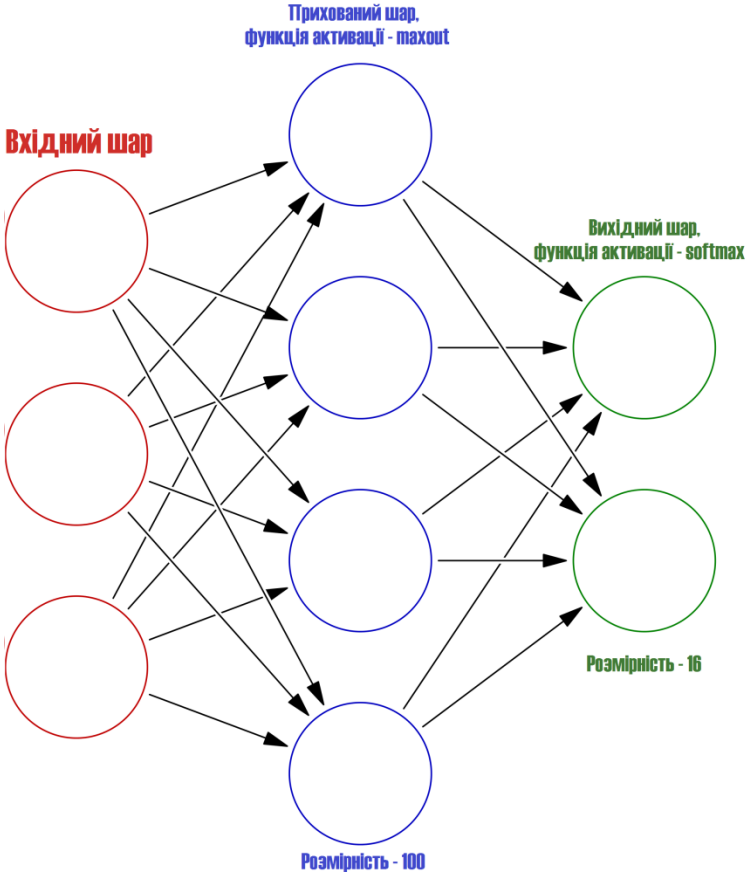

Рисунок 3 – Архітектура нейронної мережі

2.2.2 ДЕРЕВА РІШЕНЬ

Дерева рішень (інші назви: дерева класифікацій, регресійні дерева) — тип моделей, що використовується в галузі статистики та аналізу даних для прогнознозування та класифікації. Дерево складається з набору листів та гілок. На ребрах («гілках») дерева рішень знаходяться атрибути, від яких залежить функція втрат, в «листах» знаходяться значення даної функції, а в інших вузлах — атрибути, за якими відрізняються екземпляри. Щоб класифікувати

новий екземпляр, потрібно спуститися по дереву відповідно до його характеристик до листа і видати відповідне значення. Дерева рішень широко використовуються в інтелектуальному аналізі даних, так як мають просту інтерпретацію, в той же час досягаючи високої точності на задачах машинного навчання[11].

Кожен лист являє собою значення цільової змінної, тобто класом, яке залежиться від попередніх вузлів при русі від кореня. Дерево може бути навченим через поділ вихідних наборів змінних на підмножини, що базуються на зміні значень вхідних змінних. Такий процес буде повторюватись всередині кожної з отриманих підмножин. Такі рекурсивні дії закінчуються тоді, коли підмножина має ті ж значення цільової змінної, а отже, не додає цінності для прогнозу чи класифікації. Процедура «згори донизу» є прикладом жадібного алгоритму, і на сьогоднішній день є найбільш поширеною стратегією побудови дерев рішень, але є і інші[11]. В машинному навчанні, дерева рішень можуть використовуватись в якості математичних та програмних методів, для допомоги в описі, класифікації і узагальненні набору даних.

Загальний алгоритм побудови дерева рішень за тестовими прикладами виглядає таким чином:

- Вибираємо черговий атрибут x, поміщаємо його в корінь.
- Для всіх його значень i:
  - Залишаємо з тестових прикладів тільки ті, у яких значення атрибута x дорівнює i
  - Рекурсивно будуємо дерево в цьому нащадку

Вибір чергового атрибута зазвичай здійснюється на підставі коефіцієнту Джині(9) або на підставі ентропії чи приросту інформації(10).

$$G = 1 - \sum_{j=1}^{c} p_j^2, \tag{9}$$

де c – кількість класів;

p_j – частка записів з класу j у наборі даних.

$$H = -\sum_{j=1}^{c} p_j \log(p_j). \tag{10}$$

До інших способів вибору атрибутів відносяться:

- Алгоритм C4.5, де вибір атрибута відбувається на підставі нормалізованого приросту інформації.
- Алгоритм CART і його модифікації.
- Автоматичний детектор взаємодії Хі-квадрат (CHAID). Виконує багаторівневий поділ при розрахунку класифікації дерев[11].

Практично, використання останніх алгоритмів часто призводить до явища перенавчання, коли алгоритм запам’ятовує вхідний набір даних і не узагальнюється до нових екземплярів.

Серед інших методів інтелектуального аналізу даних, дерева рішень мають такі переваги:

- Прості для розуміння і інтерпретації. Люди можуть зрозуміти моделі дерева рішень після короткого пояснення;
- Вимагають невеликої підготовки даних. Інші методи часто вимагають нормалізації даних, введення фіктивних змінних і видалення порожніх значень;
- Можливість обробляти як числові так і категоріальні дані. Інші методи, як правило, спеціалізуються на аналізі масивів даних, які мають тільки один тип змінної (наприклад, правила співвідношення можуть бути використані тільки з номінальним типом змінних в той час як нейронні мережі можуть бути використані тільки з числовими змінними);
- Використовує модель білого ящика. Якщо дана ситуація спостерігається в моделі, пояснення цього легко випливає через булеву логіку. На противагу цьому, в моделі чорного ящика, пояснення результатів, як правило, важко зрозуміти, як наприклад, за допомогою штучної нейронної мережі.
- Можливість перевірки моделі з використанням статистичних тестів. Це дозволяє обгрунтувати надійність моделі.

- Стійкі. Працюють добре, навіть якщо припущення, на яких вони базуються, дещо порушуються, на відміну від істинної моделі, з якої були отримані дані.
- Добре працює з великими наборами даних. Великі обсяги даних можуть бути проаналізовані з використанням стандартних обчислювальних ресурсів в розумний часовий термін.

2.2.3 НАЇВНИЙ БАЄСОВИЙ КЛАСИФІКАТОР

Наївний баєсовий класифікатор — модель, що базується на теоремі Баєса для визначення ймовірності приналежності екземпляра до одного з класів C за умови того, що залежні змінні приймають задані значення. Це означає що, якщо на основі відомих змінних можна точно визначити, до якого класу належить екземпляр, баєсовий класифікатор підсумує, що ймовірність приналежності до даного класу рівна 1. У інших випадках, коли екземпляр може з різною ймовірністю належати до різних класів, результатом роботи класифікатора буде набір ймовірностей приналежності до певного класу[12].

Ідеальний баєсовий класифікатор в певному сенсі є оптимальним. Його результат не може бути полікращений, адже у всіх випадках, коли можна навести однозначну відповідь, він її дасть — а в інших випадках, результат кількісно характеризує міру цієї неоднозначності. Разом з тим, в оптимальності знаходиться і основний недолік ідеального баєсового класифікатора: для його побудови потрібно знайти вибірку, яка б містила всі можливі комбінації змінних — а розмір такої вибірки експоненційно зростає із зростанням числа змінних[]. Для подолання описаної вище проблеми на практиці використовують наївний баєсовий класифікатор — класифікатор, заснований на припущенні про те, що змінні незалежні. Використання даного припущення дозволяє обмежитись лише впливом кожної змінної окремо на приналежність екземпляра до одного з класів.

Перевагою підходу є те, що розмір вибірки може бути зменшеним від експоненційного до лінійного. Недоліком — те, що модель точна лише у

випадку, коли виконується припущення про незалежність, що буває надзвичайно рідко. В інших випадках обчислені ймовірності вже не є точними[12]. Однак зниженням точності часто можна знехтувати, і навіть у разі істотної залежності між змінними результат роботи класифікатора продовжує корелювати з істинною приналежністю укземпляра класів. При цьому достоїнства класифікатора (висока швидкість роботи, простота і масштабованість, помірні вимоги до пам'яті) часто переважують недоліки.

У роботі було прийнято поліноміальний вид даного класифікатора згладжуванням за Лапласом, де вектори атрибутів відображають частоти, з якими певні події були згенеровані за допомогою поліноміального розподілу з параметрами представленими в формулі (11).

$$\theta_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}, \quad (11)$$

де α=1 для згладжування за Лапласом;

N_{yi} – кількість разів атрибут і зустрічається в екземплярах класу y;

N_y – повна кількість атрибутів класу y;

n – кількість атрибутів.

2.2.4 ПРИХОВАНА МАРКОВСЬКА МОДЕЛЬ

Прихована марковська модель — це статистична марковська модель, у якій система, що моделюється, розглядається як марковський процес із неспостережуваними (прихованими) станами. ПММ може бути представлено як найпростішу динамічну баєсову мережу[13].

У простіших марковських моделях (таких як ланцюги Маркова) стан є безпосередньо видимим спостерігачеві, і тому ймовірності переходу станів є єдиними параметрами. У прихованій марковській моделі стан не є видимим безпосередньо, але вихід, залежний від стану, видимим є. Кожен стан має ймовірнісний розподіл усіх можливих вихідних значень. Тому послідовність символів, згенерована ПММ, дає якусь інформацію про послідовність станів. Зауважте, що прикметник «прихований» стосується послідовності станів, якою

проходить модель, а не параметрів моделі; модель все одно називають «прихованою» марковською моделлю, навіть якщо ці параметри відомі точно. Приховані марковські моделі відомі в першу чергу завдяки їхньому застосуванню в розпізнаванні часових шаблонів, таких як розпізнавання мовлення, рукописного введення, жестів, морфологічної розмітки, мелодій для акомпонуваня, часткових розрядів та в біоінформатиці.

Приховані марковські моделі можуть розглядатися як узагальнення сумішевої моделі, де приховані змінні, що контролюють, яка складова суміші обиратиметься для кожного спостереження, пов'язані марковським процесом, а не є незалежними одна від одної. Нещодавно приховані марковські моделі було узагальнено до подвійних марковських моделей та триплетних марковських моделей, що дозволяє розглядати складніші структури даних та моделювати нестаціонарні дані[13].

2.2.5 МЕТОД ОПОРНИХ ВЕКТОРІВ

Метод опорних векторів — метод класифікації, належить до групи граничних методів; визначає класи за допомогою меж просторів. Опорними векторами вважаються об'єкти множини, що лежать на цих межах. Класифікація вважається вдалою, якщо простір між межами — порожній[14].

Метод опорних векторів - це набір схожих алгоритмів виду «навчання із вчителем». Ці алгоритми зазвичай використовуються для задач класифікації та регресійного аналізу. Метод належить до розряду лінійних класифікаторів. Особливою властивістю методу опорних векторів є безперервне зменшення емпіричної помилки класифікації та збільшення проміжку. Тому цей метод також відомий як метод класифікатора з максимальним проміжком. Основна ідея методу опорних векторів – перевід вихідних векторів у простір більш високої розмірності та пошук роздільної гіперплощини з максимальним проміжком у цьому просторі. Дві паралельні гіперплощини будуються по обидва боки гіперплощини, що розділяє наші класи[14]. Роздільною гіперплощиною буде та, що максимізує відстань до двох паралельних

гіперплощин. Алгоритм працює у припущенні, що чим більша різниця або відстань між цими паралельними гіперплощинами, тим меншою буде середня помилка класифікатора.

Алгоритм побудови оптимальної роздільної гіперплощини, запропонований в 1963 році Володимиром Вапником та Олексієм Червоненкісом є алгоритмом лінійної класифікації. Однак у 1992 році Бернхард Босер, Ізабель Гійон та Вапник запропонували спосіб створення нелінійного класифікатору, в основі якого лежить перехід від скалярних добутків до довільних ядер - так званий kernel trick (вперше запропонований М. А. Айзерманом, Е. М. Броверманом і Л. В. Розоноером для методу потенційних функцій)[14]. Він дозволяє будувати нелінійні роздільники. Кінцевий алгоритм вкрай схожий на алгоритм лінійної класифікації, з тією лише різницею, що кожен скалярний добуток в наведених вище формулах замінюється нелінійною функцією ядра (скалярним добутком в просторі з більшою розмірністю). У цьому просторі вже може існувати оптимальна роздільна гіперплощина. Через те, що розмірність отримуваного простору може бути більшою розмірності вихідного, то перетворення, що зіставляє скалярні добутки, буде нелінійним. А отже функція, що відповідає у початковому просторі оптимальній роздільній гіперплощині, буде також нелінійною. Варто відзначити, що якщо початковий простір має досить високу розмірність, то можна сподіватися, що вибірка в ньому виявиться лінійно роздільною.

Найбільш поширені ядра:

- Поліноміальне (однорідне):

$$k(x, x') = (x \cdot x')^d,$$

де x – набір атрибутів екземпляра, точка;

x’ – набір атрибутів іншого екземпляра, інша точка;

d – степінь полінома.

- Поліноміальне (неоднорідне):

$$k(x, x') = (x \cdot x' + 1)^d.$$

- Радіальна базисна функція:

$$k(x, x') = \exp\left(-\gamma\left||x - x'|\right|^2\right), \quad (12)$$

де γ – гамма параметр.

- Радіальна базисна функція Гаусса:

$$k(x, x') = \exp\left(\frac{-\left||x-x'|\right|^2}{2\sigma^2}\right),$$

де σ – стандартне відхилення.

- Сигмоїд:

$$k(x, x') = \tanh(\mathrm{r}x \cdot \mathrm{x}' + c),$$

Де r, c – параметри.

Переваги методу опорних векторів

- найшвидший метод знаходження вирішальних функцій;
- метод зводиться до вирішення задачі квадратичного програмування у випуклій області, яка завжди має єдине вирішення;
- метод знаходить роздільну смугу максимальної ширини, що дозволяє надалі здійснювати кращу класифікацію;
- за умови різного вибору ядер можна емулювати інші підходи. Наприклад, великий клас нейронних мереж можна представити у вигляді методу опорних векторів з визначеними ядрами;
- теоретичне обґрунтування: кінцеве правило обирається не за допомогою деяких евристик, а відповідно до оптимізації деякої функції.

Недоліки методу опорних векторів.

- мала кількість параметрів для налаштування: після того як ядро зафіксували, єдиним варіативним параметром лишається коефіцієнт помилки C;
- метод чутливий до шумів та стандартизації даних;
- не існує загального підходу до автоматичного вибору ядра у випадку лінійної нероздільності класів;
- повільне навчання.

2.2.6 ВИПАДКОВІ ЛІСИ

Випадкові ліси — алгоритм машинного навчання, запропонований Лео Брейманом і Адель Катлер, що полягає у використанні комітету (збірки) дерев рішень. Алгоритм застосовується для задач класифікації, регресії і кластеризації.

Нехай навчальна вибірка складається з N прикладів, розмірність простору ознак дорівнює M, і заданий параметр m[15]. Усі дерева комітету будуються незалежно один від одного за такою процедурою:

- Згенеруємо випадкову підвибірку з повторенням розміром n з навчальної вибірки. (Таким чином, деякі приклади потраплять в неї кілька разів, а приблизно N / 3 прикладів не ввійдуть у неї взагалі)
- Побудуємо дерево рішень, яке класифікує приклади даної підвибірки, причому в ході створення чергового вузла дерева будемо вибирати ознаку, на основі якої проводиться розбиття, не з усіх M ознак, а лише з m випадково вибраних. Вибір найкращого з цих m ознак може здійснюватися різними способами. В оригінальному коді Брейман використовується критерій Джині, що застосовується також в алгоритмі побудови дерев рішень CART. У деяких реалізаціях алгоритму замість нього використовується критерій приросту інформації.
- Дерево будується до повного вичерпання підвибірки і не піддається процедурі відсікання.

Класифікація об'єктів проводиться шляхом голосування: кожне дерево комітету відносить об'єкт, який класифікується до одного з класів, і перемагає клас, за який проголосувало найбільше число дерев.

Оптимальне число дерев підбирається таким чином, щоб мінімізувати помилку класифікатора на тестовій вибірці[15]. Випадкові ліси, отримані в результаті застосування технік, описаних раніше, можуть бути природним чином використані для оцінки важливості змінних в задачах регресії та класифікації. Перший крок в оцінці важливості змінної в тренувальному наборі — тренування випадкового лісу на цьому наборі. Під час процесу побудови

моделі для кожного елемента тренувального набору вважається так звана out-of-bag — помилка. Потім для кожної сутності така помилка опосередковується по всьому випадковому лісі.

Для того, щоб оцінити важливість j -ого параметра після тренування, значення j-ого параметра перемішуються для всіх записів тренувального набору та out-of-bag — помилка рахується знову. Важливість параметра оцінюється шляхом усереднення по всіх деревах різниці показників out-of-bag — помилок до і після перемішування значень. При цьому значення таких помилок нормалізуються на стандартне відхилення.

Параметри вибірки, які дають більші значення, вважаються більш важливими для тренувального набору. Метод має наступний потенційний недолік — для категоріальних змінних з великою кількістю значень метод схильний вважати такі змінні більш важливими. Часткове переваження значень в цьому випадку може знижувати вплив цього ефекту.

Переваги:

- Здатність ефективно обробляти дані з великим числом ознак і класів.
- Нечутливість до масштабування (і взагалі до будь-яких монотонних перетворень) значень ознак.
- Однаково добре обробляються як безперервні, так і дискретні ознаки. Існують методи побудови дерев за даними з пропущеними значеннями ознак.
- Існують методи оцінювання значущості окремих ознак в моделі.
- Внутрішня оцінка здатності моделі до узагальнення.
- Здатність працювати паралельно в багато потоків.
- Масштабованість.

Недоліки:

- Алгоритм схильний до перенавчанню на деяких завданнях, особливо з великою кількістю шумів.

- Великий розмір отримуваних моделей. Потрібно O(NK) пам'яті для зберігання моделі, де K — число дерев.

2.2.7 ГОЛОСУВАННЯ

У статистиці і машинному навчанні, методи голосування використовують різні алгоритми навчання для отримання кращої прогностичної продуктивності, ніж можна було б отримати окремо з будь-якого з складових алгоритмів навчання. На відміну від статистичного ансамблю в статистичній механіці, який, як правило, нескінченний, ансамбль моделей машинного навчання відноситься тільки до конкретної скінченної кількості альтернативних моделей, але, як правило, дозволяє набагато більш гнучкій структурі існувати серед цих альтернатив[16].

Алгоритми машинного навчання з вчителем зазвичай виконують пошук через простір гіпотез, щоб знайти підходящу гіпотезу, яка буде робити доцільні прогнози для конкретної проблеми. Навіть якщо простір гіпотез містить гіпотези, які дуже добре підходять для тієї чи іншої проблеми, знайти найкращу може бути дуже важко. Ансамблі об'єднують кілька гіпотез для формування кращої гіпотези. Термін ансамбль, як правило, зарезервований для методів, які генерують кілька гіпотез, використовуючи ту ж базу для навчання.

Оцінюючий прогноз ансамблю, як правило, вимагає більше обчислень, ніж оцінка передбачення однієї моделі, тому ансамблі можна розглядати як спосіб компенсації низької якості алгоритмів навчання, виконуючи багато додаткових обчислень. Швидкі алгоритми, такі як дерева рішень зазвичай використовуються з ансамблями, хоча і більш повільні алгоритми можуть отримати вигоду з ансамблевих методів.

Голосування серед кількох моделей - ансамблевий алгоритм навчання, який може бути навчений, а потім використагтй для прогнозування. В той же час такий натренований ансамбль являє собою єдину гіпотезу. Ця гіпотеза, однак, не обов'язково міститься в просторі гіпотез алгоритмів, з яких вона побудована. Таким чином збільшується гнучкість в функціях, які можуть бути

представлені. Така гнучкість може, теоретично, дати їм можливість більш точно моделювати навчальні дані, ніж окремі моделі, але також більш здатні до перенавчання та запам'ятовування тренувальних даних, що може призвести до гірших результатів.

Емпірично, голосування, як правило, дає кращі результати, коли існує значна різноманітність серед моделей. Більш випадкові алгоритми (наприклад, випадкові дерева рішень) можуть бути використані для отримання сильніших ансамблів, ніж добре продумані та підібрані алгоритми (наприклад, зменшення ентропії дерев рішень).

2.2.8 МЕТОД СТИМУЛЮВАННЯ ГРАДІЄНТА

Метод стимулювання градієнта - метод машинного навчання для проблем регресії і класифікації, яка створює модель прогнозування у вигляді ансамблю слабких моделей прогнозування, зазвичай дерев рішень. Він будує модель поетапно і дозволяє оптимізувати довільну функцію втрат[17].

Ідея методу стимулювання градієнта виникла в спостереженні Лео Браймана, що стимулювання може бути інтерпретоване як алгоритм оптимізації для підходящої функції втрат. Явні методи стимулювання градієнта були згодом розроблені Джеромом Х. Фрідманом одночасно з більш загальним підходом від Лью Мейсона, Джонатана Бакстера, Пітера Бартлетта і Маркуса Фріна. Вони представили абстрактне уявлення алгоритмів стимулювання як ітераційних алгоритмів функціонального градієнтного спуску[17]. Тобто, алгоритми, які оптимізують функцію втрат у просторі функцій шляхом ітеративного вибору функцій (слабких гіпотез), які вказують в негативному напрямку градієнта. Цей підхід через функціональний градієнт призвів до розробки алгоритмів стимулювання градієнту в багатьох областях машинного навчання і статистики за межами регресії і класифікації.

Повний алгоритм описаний нижче:

Вхід: набір даних для тренування у формі (13), диференційована функція втрат у формі (14) та кількість ітерацій M.

$$\{(x_i, y_i)\}_{i=1}^{n}, \quad (13)$$

де x_i - атрибути екземпляра;

y_i – клас екземпляра;

n - кількість екземплярів.

$$L\big(y, F(x)\big), \quad (14)$$

де L – функція втрат;

F – гіпотеза.

1. Ініціалізація моделі константою γ:

$$F_0(x) = \arg min \sum_{i=1}^{n} L(y_i, \gamma).$$

2. Цикл від m = 1 до M:

    1. Обраховуємо псевдо-нев’язки:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}.$$

    2. Пристосовуємо слабку гіпотезу $h_m(x)$ до псевдо-нев’язок.

    3. Обраховуємо множник γ_m за допомогою вирішення такого завдання оптимізації:

$$\gamma_m = \arg\min \sum_{i=1}^{n} L\big(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\big).$$

    4. Оновимо модель:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Результуюча модель: $F_M(x)$.

2.3 ОПИС ТЕСТОВИХ ДАНИХ

Велика кількість пов'язаних робіт використовує Аугсбурзький Еталонний Тест Прогнозування Місцезнаходження(АЕТПМ)[18] для оцінки запропонованих алгоритмів для вирішення задачі. Цей набір даних включає в

себе історію переміщень 4-х різних людей по одному поверху в офісній будівлі з часовими мітками. Поверх містить ряд виділених місць: офіси 402-412, коридор, кухня, вбиральня, принтерна(рис. 4). Відсутність певної особи в будівлі позначається записом "відсутній".

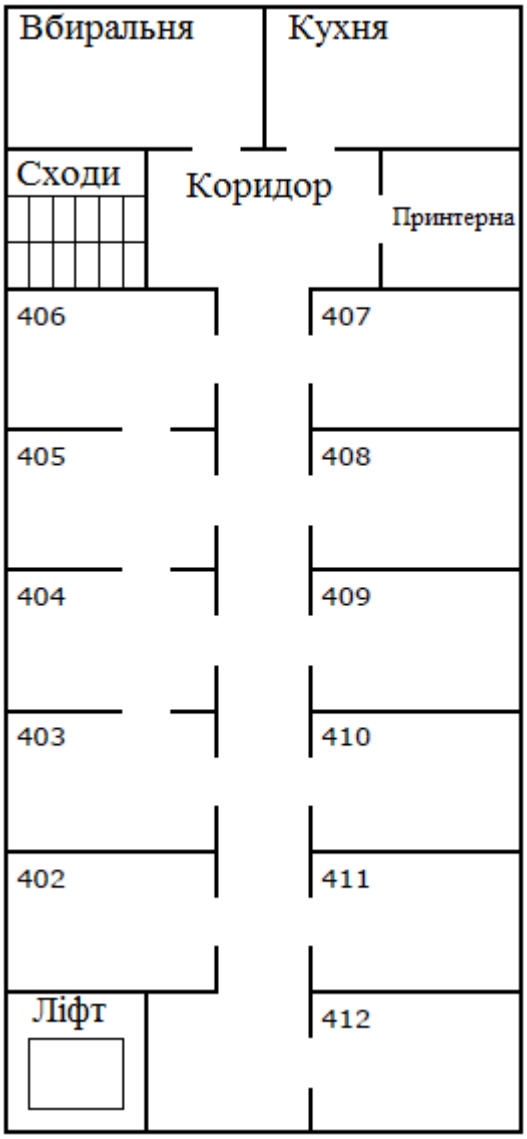

Рисунок 4 – План тестового поверху

Дані були отримані таким чином: за допомогою невеликої програми з простим графічним, призначеним для користувача, інтерфейсом, яка була реалізована на КПК, відображався план поверху. Всі піддослідні люди були обладнані КПК з даною програмою. Коли людина входить в певну кімнату, вона повинна була натиснути на відповідну мітку розташування на КПК[18]. Для кожного натиску в програмі, зберігалась мітка часу, місце розташування,

яке було введено(кімната), ім’я людини, і ще раз відмітка часу, цього разу в машинному форматі в мілісекундах, для спрощення процедури подальших обчислень.

В даній роботі ці файли конкатенуються для кожної людини окремо. Формат записів у файлі може бути представлений таким шаблоном:

рік.місяць.день година.хвилина.секунда;місцезнаходження;людина;часова_мітка,

де рік – 4-цифровий формат;

місяць, день, година, хвилина, секунда – 2-цифровий формат;

місцезнаходження – текстова строка з назвою кімнати;

людина – ідентифікатор людини, яка поставила мітку;

часова_мітка – машинний формат часу типу UNIX.

Гістограма відвідуваних місць для кожної людини, показана на рисунку 5, де кожен з 4-х різних людей представлений як людина А, Б, В або Г і має відповідний колір.

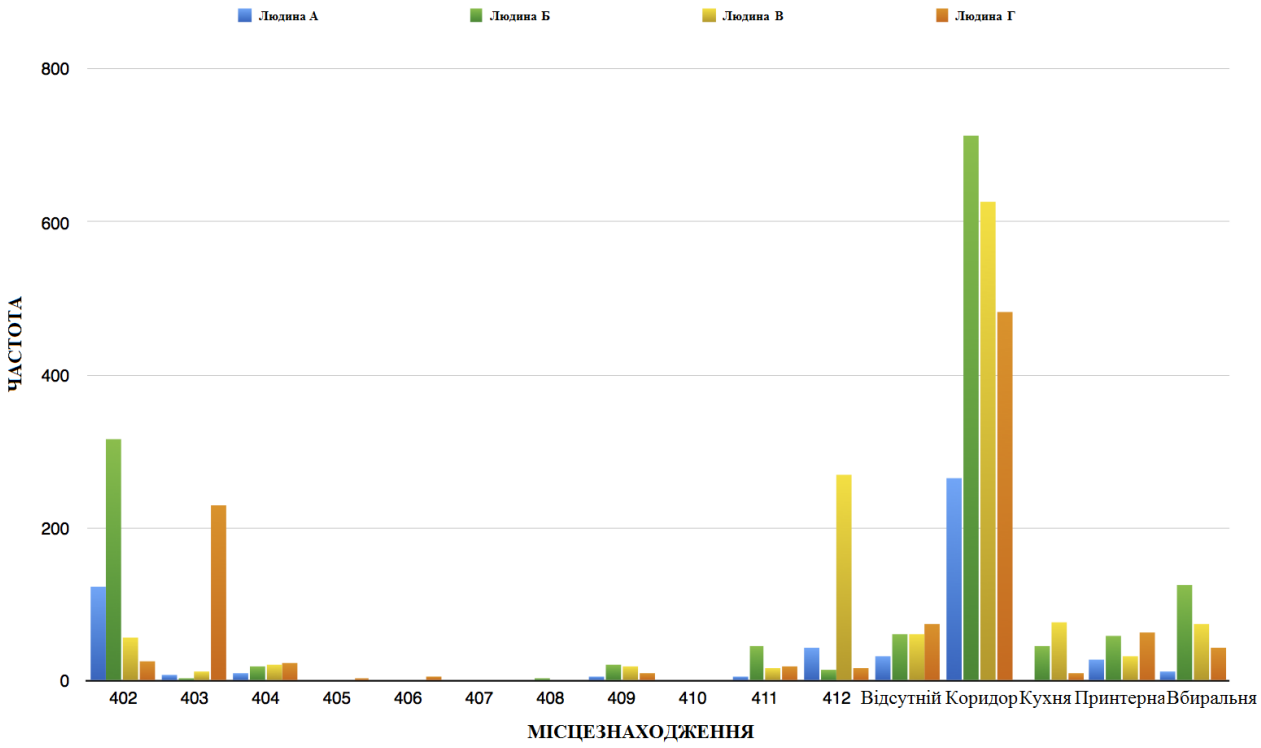

Рисунок 5 – Частота місцезнаходжень людей

Відсутність у АЕТПМ, крім місця розташування, інших контекстних даних робить його ідеальним тестовим середовищем для оцінки класичних

методів розв'язання завдання прогнозування місцезнаходження об’єктів у замкнутих системах.

Базова модель, яка б прогнозувала найчастіший клас, в даному разі – «Коридор», мала б точність близьку до 50 %, саме тому подолання бар’єру в 50% точності на тестовому наборі даних є одним із найголовніших показників успішності розробленого алгоритму.

2.4 ПРОЦЕС ОЦІНКИ АЛГОРИТМІВ

Для оцінки запропонованих рішень та алгоритмів використовується перехресна перевірка. Перехресна перевірка являє собою метод перевірки моделей машинного навчання для оцінки того, як результати статистичного аналізу зможуть узагальнитись до незалежного набору даних. Даний метод використовується в основному в тих місцях, де метою є прогнозування, і потрібно оцінити, наскільки точно модель прогнозування буде працювати на практиці. У задачі прогнозування, модель, як правило, отримує набір даних відомих даних, на яких проводиться навчання (навчального набору даних), а також набір невідомих даних (дані, які модель ще ніколи не бачила), на якому проводиться оцінка алгоритму (набір даних для тестування)[19]. Метою перехресної перевірки є визначення набору даних для оцінки моделі вже на етапі навчання, для того щоб зменшити вплив такої проблеми як перенавчання, а також дати уявлення про те, як модель буде узагальнюватись до незалежного набору даних (тобто, невідомого набору даних, наприклад, з реальної проблеми).

Різновиди перехресної перевірки:

1. Вичерпна
    a. Залишок N зовні
    b. Залишок 1 зовні
2. Невичерпна
    a. К-кратна
    b. 2-кратна

Для оцінки методів у даній роботі була обрана K-кратна перехресна перевірка. При цьому початкова вибірка випадковим чином розбивається на K підвиборок однакового розміру. З цих K підвиборок, одна зберігається в якості набору даних для тестування моделі, а підвибірки використовуються в якості навчальних даних. Процес перехресної перевірки повторюється K раз з кожною з K підвиборок. Результати K оцінок можуть бути усереднені (або комбіновані іншим способом) для отримання комплексної оцінки. Перевагою цього методу є те, що всі спостереження використовуються як для тренування, так і для перевірки, в той же час кожне спостереження використовується для перевірки в точності один раз. Зазвичай використовується 10-кратна перехресна перевірка, тому і в даній роботі K=10. Процес оцінки моделей зображений на рисунку 6.

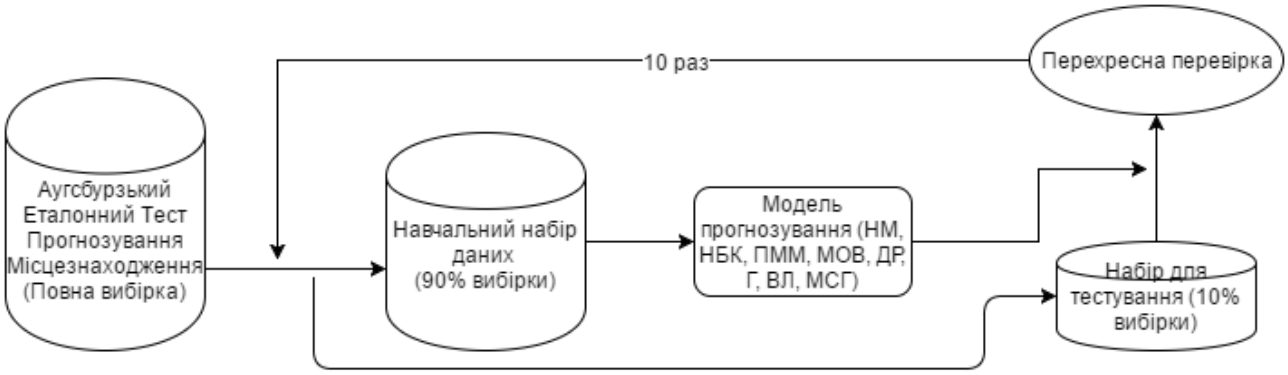

Рисунок 6 – Процес оцінки алгоритмів

2.5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для програмної реалізації описаних методів та алгоритмів було обрано мову програмування Python версії 3.3. Python підтримує кілька парадигм програмування, в тому числі об'єктно-орієнтованого, імперативного та функціонального програмування або процедурних стилів. Вона має динамічну систему типів і автоматичне керування пам'яттю, а такожмає велику кількість бібліотек різного призначення. Його філософія дизайну підкреслює читаність коду, а його синтаксис дозволяє програмістам висловлювати поняття в меншій кількості рядків коду, ніж це можливо в таких мовах, як C ++ або Java. Python інтерпретатори доступні для багатьох операційних систем, дозволяючи запускати код написаний на Python на найрізноманітніших системах. За

допомогою сторонніх інструментів Python код може бути упакований в автономні виконувані програми для деяких з найбільш популярних операційних систем, так що програмне забезпечення Python може бути поширеним і використовуватись у різних середовищах, навіть там де інтерпретатор Python не встановлений. Замість того, щоб вимагати всіх функцій у ядрі мови, Python був розроблений, щоб бути максимально розширюваним. Основою реалізації стала бібліотека scikit-learn[19] – відкрита безкоштовна бібліотека для машинного навчання, у якій зібрані алгоритми для вирішення різних задач – класифікації, прогнозування, регресії чи кластеризації. Також вона побудована на основі чисельних та наукових Python-бібліотек Numpy та Scipy, що робить її зручним інструментом для роботи з чисельними даними такого типу як АЕТПМ. Лістинг програми, яка перевіряє запропоновані методи відображений у додатку А.

2.6 ВИСНОВОК

У розділі було описано загальний процес підбору оптимального алгоритма для вирішення задачі прогнозування місцезнаходження об’єктів у контекстно-залежних системах. Серед запропонованих методів знаходяться як збірні, так і автономні алгоритми, а саме нейронні мережі, наївний баєсів класифікатор, дерева рішень, метод опорних векторів, прихована марковська модель, випадкові ліси, збірка декількох моделей через голосування та метод стимулювання градієнта. Для оцінки прогнозувальної здатності обраних підходів використовується 10-кратна перехресна перевірка з усередненням, а вибіркою для тренування та валідації створених моделей виступає Аугсбурзький Еталонний Тест Прогнозування Місцезнаходження. Тестування запропонованих алгоритмів проводиться в двох сценаріях: прогнозування наступного місцезнаходження лише на основі одного, останнього, місця і з використанням 4 попередніх місць.

3 ОГЛЯД РЕЗУЛЬТАТІВ ТЕСТУВАННЯ РОЗРОБЛЕНИХ АЛГОРИТМІВ

3.1 РЕЗУЛЬТАТИ ДЛЯ ПЕРШОГО СЦЕНАРІЮ

Результати оцінки алгоритмів для сценарію тестування з використанням лише одного попереднього місцезнаходження зведені у таблицю 3. Найбільша точність в колонці виділена жирним шрифтом.

Таблиця 3 – Результати оцінки на основі одного попереднього місця

Алгоритм	Точність			
	Людина А	Людина Б	Людина В	Людина Г
Дерева Рішень	0.728	0.717	0.709	0.698
Нейронні Мережі	0.728	0.717	0.709	0.672
Випадкові Ліси	0.728	0.717	0.709	**0.699**
Метод Опорних Векторів	**0.776**	**0.745**	0.709	0.695
Прихована Марковська Модель	0.694	0.706	**0.715**	0.656
Наївний Баєсів Класифікатор	0.5	0.498	0.497	0.48
Голосування	0.728	0.717	0.709	0.695
Метод Стимулювання Градієнта	0.728	0.717	0.709	0.698

Як бачимо, використовуючи лише дані одного попереднього місцезнаходження людини в системі, можна спрогнозувати кімнату, до якої вона переміститься в найближчий час з точністю близько 75% в залежності від якості даних для тренування. Варто зазначити, що НБК при такому сценарії звівся до базової моделі з точністю 50%, адже він отримує на вхід лише один параметр, і тому враховується лише найчастіший клас. Найбільшу точність при порівняно великій кількості даних (люди А, Б, В) досягнули метод опорних векторів та прихована марковська модель. В цей же час при не настільки обширних даних найбільше здібним узагальнитись виявився алгоритм

випадкових лісів. Графік точності прогнозування місцезнаходження в даному сценарії зображений на рисунку 7.

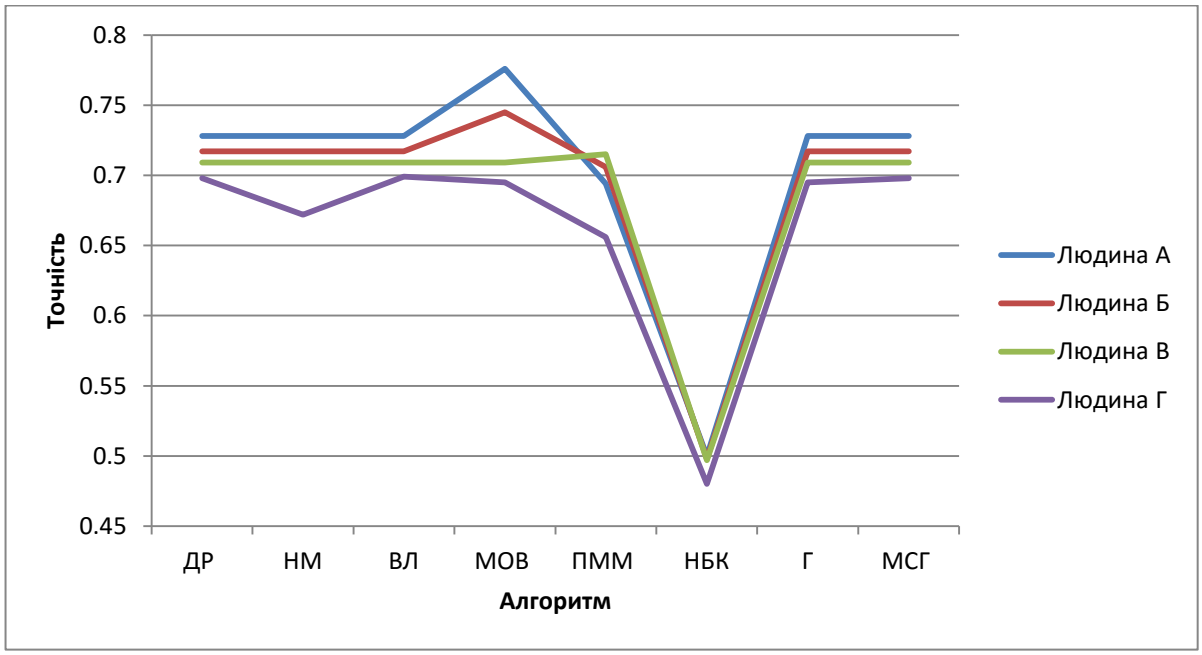

Рисунок 7 – Графік точності алгоритмів за першим сценарієм

3.2 РЕЗУЛЬТАТИ ДЛЯ ДРУГОГО СЦЕНАРІЮ

Результати оцінки алгоритмів для сценарію тестування з використанням чотирьох попередніх місцезнаходжень зведені у таблицю 4. Найбільша точність в колонці виділена жирним шрифтом.

Як бачимо, використання додаткових параметрів, а саме останніх чотирьох місцезнаходжень людини в системі, приводить до збільшення точності на 3-5 % і дозволяє наблизитись до 80 % бар'єру. Варто зазначити, що НБК при такому сценарії починає давати конкурентноспроможні результати. Найбільшу точність серед автономних алгоритмів знову досягнули метод опорних векторів та випадкові ліси. Графік точності прогнозування місцезнаходження в даному сценарії зображений на рисунку 8.

Серед збірних алгоритмів виділився метод голосування між трьома моделями, який зміг правильно спрогнозувати найбільшу кількість точок у наборі даних для тестування для людей Б та В. Так як даний метод вимагає тренування трьох окремих моделей, що є обчислювально важко, в обмежених за часом методах прийняття рішень та системах можна використати МОВ та ВЛ, адже вони

Таблиця 4 – Результати оцінки на основі чотирьох попередніх місць

Алгоритм	Точність			
	Людина А	Людина Б	Людина В	Людина Г
Дерева Рішень	0.79	0.769	0.749	0.752
Нейронні Мережі	0.723	0.685	0.659	0.541
Випадкові Ліси	**0.796**	0.763	0.753	0.756
Метод Опорних Векторів	0.789	0.77	0.755	**0.762**
Наївний Баєсів Класифікатор	0.741	0.748	0.7	0.641
Голосування	0.79	**0.773**	**0.759**	0.751
Метод Стимулювання Градієнта	0.79	0.769	0.752	0.754

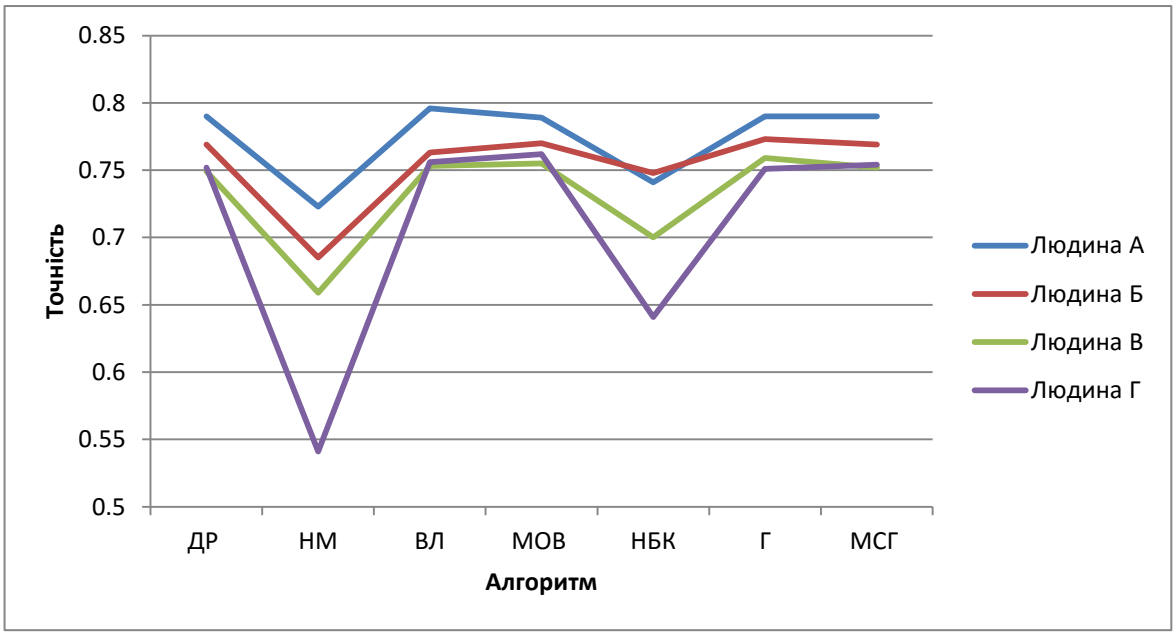

Рисунок 8 - Графік точності алгоритмів за другим сценарієм

показали 2 і 3 результати відповідно.

3.3 МЕТОД ПІДБОРУ ГІПЕРПАРАМЕТРІВ

Для збільшення продуктивності алгоритмів машинного навчання, потрібно підібрати оптимальний набір гіперпараметрів - факторів, які модифікують процедуру навчання. Різниця підбору гіперпараметрів від

тренування параметрів моделі полягає в тому, що гіперпараметри контролюють поведінку при тренуванні параметрів, а отже є окремим завданням оптимізації(рис. 9).

Сітковий пошук є загальновживаним способом коригування гіперпараметрів, який заключається в простому вичерпному пошуці через певну підмножину простору гіперпараметрів. Оскільки простір пошуку зазвичай містить дійсні значення, то його потрібно дискретизувати перед початком підбору [17].

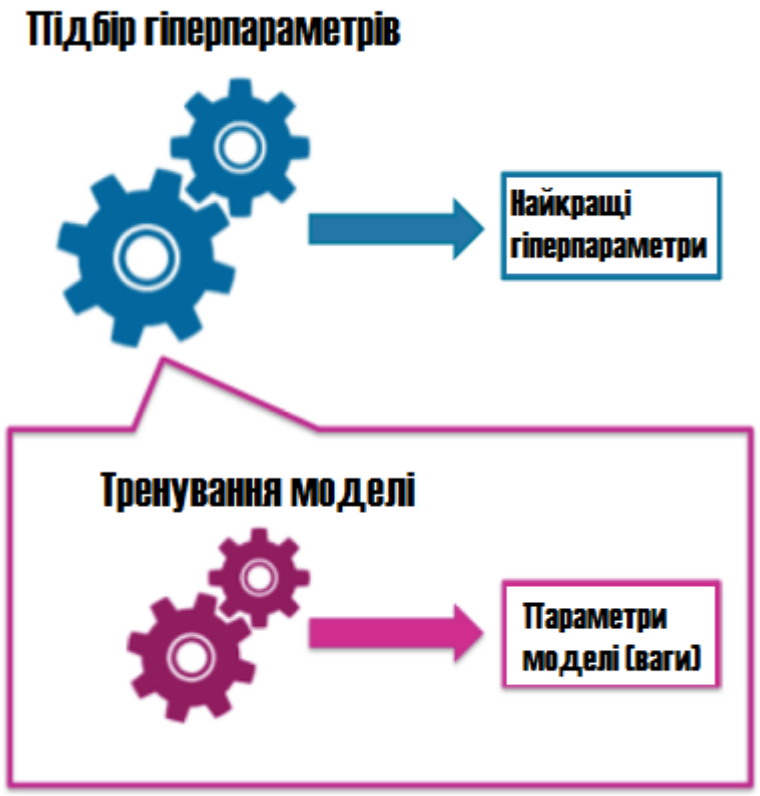

Рис 9 – Різниця між підбором гіперпараметрів та тренуванням моделі

Очевидно, що для різних алгоритмів машинного навчання простір гіперпараметрів відрізняється.

3.4 ПРОСТОРИ ГІПЕРПАРАМЕТРІВ АЛГОРИТМІВ

3.4.1 ГІПЕРПАРАМЕТРИ ДЕРЕВ РІШЕНЬ

Для покращення точності прогнозування за допомогою алгоритму дерев рішень підбирають наступні гіперпараметри:

- Критерій вибору чергового атрибута, можливі значення якого – критерій Джині, описаний формулою (9), та критерій ентропії, описаний формулою (10);
- Мінімальна кількість екземплярів щоб сформувати листок, яка вимагає, щоб кожна клітина (тобто кожен листовий вузол) охоплював задану мінімальну кількість екземплярів, випробовані значення - 1, 5 та 10 екземплярів;
- Максимальна кількість листків, яка обмежує максимальну кількість листових вузлів, таким чином регулюючи кількість розбиттів, випробовані значення – Всі екземпляри можуть бути листками, 5, 10 і 20 листків.
- Максимальна глибина дерева, яка напряму обмежує кількість розбиттів, випробовані значення – Повна глибина(необмежено), 2, 5, 10 розбиттів;
- Мінімальна кількість екземплярів для поділу регулює кількість поділів через обмеження кількості екземплярів, які залишаться після розбиття, випробовані значення – 2, 10, 20 екземплярів.

3.4.2 ГІПЕРПАРАМЕТРИ МЕТОДУ ОПОРНИХ ВЕКТОРІВ

Метод опорних векторів теж має набір гіперпараметрів, які напряму впливають на результуючу модель та її точність прогнозування на наборі даних для тестування.

Серед таких параметрів виділяються:

- Коефіцієнт штрафу - параметр, який дозволяє розміняти помилку навчання на складність моделі. Якщо він занадто великий, для нероздільних точок ставиться у відповідність високий штраф, тому зберігається багато опорних векторів і модель може перенавчитись. Якщо він занадто малий, модель може недовчитись. Випробовані значення параметру штрафу – 1, 10, 100, 1000;

- Тип ядра – спосіб переводу заданих атрибутів у більш зручний для лінійного класифікатора простір. Випробовані типи ядер – лінійне та через радіальну базисну функцію, описану формулою (12);
- Параметр γ – гіперпараметр ядра, який відповідає за те, наскільки значення кожного з екземплярів впливає на вибір опорних векторів. Його можна розглядати як зворотню величину до радіуса впливу екземплярів, яких модель обрала в якості опорних векторів. Малі значення параметру означають далекий вплив, великі – близький вплив. Випробовані значення параметру - 0.01, 0.1, 0.25, 0.5, 0.9.

3.4.2 ГІПЕРПАРАМЕТРИ ВИПАДКОВИХ ЛІСІВ

Так як випадкові ліси являються збірним(ансамблевим) методом, то гіперпараметри окремих алгоритмів, які є частиною даної збірки переносяться на випадкові ліси.

В даному разі окремі алгоритми представляють собою дерева рішень, тому такі гіперпараметри як мінімальна кількість екземплярів щоб сформувати листок, максимальна кількість листків, максимальна глибина дерева та мінімальна кількість екземплярів для поділу мають аналогічну інтерпретацію у моделі випадкових лісів, як і у моделі дерев рішень. В той же час додається найважливіший гіперпараметр зібрних моделей – кількість оцінювачів. Тому загальний набір гіперпараметрів виглядає так:

- Мінімальна кількість екземплярів щоб сформувати листок, випробовані значення - 1, 5 та 10 екземплярів;
- Максимальна кількість листків, випробовані значення – Всі екземпляри можуть бути листками, 5, 10 і 20 листків.
- Максимальна глибина дерева, випробовані значення – Повна глибина(необмежено), 2, 5, 10 розбиттів;
- Мінімальна кількість екземплярів для поділу, випробовані значення – 2, 10, 20 екземплярів.

- Кількість оцінювачів - відповідає за кількість дерев у лісі, а тому дозволяє розмінювати складність моделі і обсяг обчислень на точність прогнозування, випробовані значення – 5, 10, 15 та 20 оцінювачів.

3.4.3 ГІПЕРПАРАМЕТРИ МЕТОДУ СТИМУЛЮВАННЯ ГРАДІЄНТА

Метод стимулювання градієнта також є збірним алгоритмом, тому залежить від кількості оцінювачів та гіперпараметрів слабких гіпотез. Такими гіпотезами у даній роботі виступають дерева рішень, тому їх гіперпараметри переносяться до даного методу, а саме мінімальна кількість екземплярів щоб сформувати листок, максимальна кількість листків, максимальна глибина дерева та мінімальна кількість екземплярів для поділу.

Дані гіперпараметри та їх можливі значення наведені нижче:

- Мінімальна кількість екземплярів щоб сформувати листок, випробовані значення - 1, 5 та 10 екземплярів;
- Максимальна кількість листків, випробовані значення – Всі екземпляри можуть бути листками, 5, 10 і 20 листків.
- Максимальна глибина дерева, випробовані значення – Повна глибина(необмежено), 2, 5, 10 розбиттів;
- Мінімальна кількість екземплярів для поділу, випробовані значення – 2, 10, 20 екземплярів.
- Кількість оцінювачів - відповідає за кількість дерев у лісі, а тому дозволяє розмінювати складність моделі і обсяг обчислень на точність прогнозування, випробовані значення – 50 та 100 оцінювачів.

3.4.4 ПОВНИЙ ПРОСТІР

Усі значення гіперпараметрів, які були випробовані під час розробки алгоритму, зібрані у таблиці 5.

Таблиця 5 – Простір гіперпараметрів

Алгоритм	Критерій	Мін. кількість екземплярів для поділу	Макс. Глибина	Мін. кількість екземплярів щоб сформувати листок	Макс. кількість листків	Кількість оцінювачів	Тип ядра	γ	Штраф
ДР	Джині, ентропія	2, 10, 20	Повна, 2, 5, 10	1, 5, 10	Всі, 5, 10, 20				
МОВ							рбф, лінійне	0.01, 0.1, 0.25, 0.5, 0.9	1, 10, 100, 1000
ВЛ		2, 10, 20	Повна, 2, 5, 10	1, 5, 10	Всі, 5, 10, 20	5, 10, 15, 20			
МСГ		2, 10, 20	Повна, 2, 5, 10	1, 5, 10	Всі, 5, 10, 20	50, 100			

3.5 РЕЗУЛЬТАТИ НАЛАШТОВАНИХ АЛГОРИТМІВ

Після проведення сіткового пошуку у повному просторі гіперпараметрів на описаному тестовому наборі даних, виявилось, що РБФ ядро для методу опорних векторів у всіх випадках краще прогнозує наступне місцезнаходження, ніж лінійне ядро. Також помічено, що у методі стимулювання градієнту завжди краще працювала більша кількість оцінювачів(у даному випадку 100), що не можна сказати про метод випадкових лісів, хоч вони обидва збірні. Інші гіперпараметри показали різні оптимальні значення, що може свідчити про важливість їх оптимізації окремо під кожного користувача. Таким чином оптимальний набір гіперпараметрів зведений у таблицю 6.

Порівняння точностей до та після підбору гіперпараметрів показаний у таблиці 7. Найвищі показники точності виділені жирним шрифтом. Як бачимо підбір гіперпараметрів дозволив обійти бар’єр точності у 80% і в цілому підняв якість прогнозування на 2-3%. Найчастіше найвищу точність показали дерева рішень та випадкові ліси, в той же час результати за методом опорних векторів змінились найменше. Підбір гіперпараметрів методу стимулювання градієнта

показав достатньо високу точність, але випадкові ліси та дерева рішень перевершили його.

Таблиця 6 – Оптимальні значення гіперпараметрів

Алгоритм	Людина	Критерій	Мін. кількість екземплярів для поділу	Макс. Глибина	Мін. кількість екземплярів щоб сформувати листок	Макс. кількість листків	Кількість оцінювачів	Тип ядра	γ	Штраф
ДР	А	Джині	2	Повна	10	Всі				
	Б	Джині	2	Повна	1	5				
	В	Ентропія	2	Повна	1	20				
	Г	Джині	20	5	1	Всі				
МОВ	А							Рбф	0.5	10
	Б							Рбф	0.5	1
	В							Рбф	0.9	1
	Г							Рбф	0.25	1
ВЛ	А		20	10	5	20	5			
	Б		20	Повна	5	20	20			
	В		10	10	5	Всі	15			
	Г		20	Повна	5	20	20			
МСГ	А			Повна	2	5	100			
	Б			2	20	Всі	100			
	В			2	2	Всі	100			
	Г			2	2	Всі	100			

Таблиця 7 – Вплив підбору гіперпараметрів

	До підбору				Після підбору			
Алгоритм	А	Б	В	Г	А	Б	В	Г
ДР	0.790	0.769	0.749	0.752	**0.805**	0.778	**0.767**	0.773
ВЛ	0.796	0.763	0.753	0.756	**0.805**	**0.782**	0.766	**0.78**
МОВ	0.789	0.770	0.755	0.762	0.79	0.775	0.761	0.762
МСГ	0.790	0.769	0.752	0.754	0.802	0.774	0.764	0.772

3.6 ВИСНОВОК

Результати показують, що введення додаткової інформації у вигляді збільшення кількості попередніх точок для прогнозу наступної значно збільшують точність передбачення. При цьому найкращі показники відповідності передбачених місцезнаходжень реальним отримали випадкові ліси, метод опорних векторів та голосування між випадковими лісами, методом опорних векторів та наївним баєсовим класифікатором. Також було помічено, що збільшення кількості даних позитивно впливає на результати і прогнозування людської поведінкової моделі стає легшим.

Підбір гіперпараметрів для методів прогнозу місцезнаходження об'єктів у контекстно-залежних системах управління електроспоживанням виявився корисним, про що свідчить 2-3 відсоткове збільшення точності на заданій тестовій вибірці, а саме AETPM. Сітковий пошук рекомендується для застосування як метод підбору гіперпараметрів, адже попри свою обчислювальну складність є надійним методом, який гарантовано знайде оптимальний набір параметрів на заданій сітці. При цьому даний метод легко піддається процесу розпаралелювання і може використовувати можливості розподілених систем. Таким чином нівелюється проблема з обчислювальною складністю. Враховуючи підбір гіперпараметрів, найточнішим алгоритмом для пронозування місцезнаходження виявився метод випадкових лісів, який зміг подолати 80% бар’єр точності на AETPM. Варто зазначити, що дерева рішень також показали високий результат і можуть бути використані на системах з меншими обчислювальними можливостями, аніж потрібно для випадкових лісів. Метод опорних векторів від процедури підбору гіперпараметрів отримав найменше збільшення точності і показав результати близькі до тренування з параметрами за замовчуванням.

4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 ВСТУП

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для перевірки методів прогнозування місцезнаходження об’єктів у контекстно-залежних системах управління електроспоживанням. Програмний продукт був розроблений за допомогою мови програмування Python.

Програмний продукт є крос-платформенним та рекомендується для використання на персональних комп’ютерах під управлінням операційних систем Windows, Linux чи Mac.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями[20].

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій[21].

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам:

ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат;

– для кожної функції визначаються повні річні витрати й кількість робочих часів;

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат;

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.2 ПОСТАНОВКА ЗАДАЧІ ТЕХНІКО-ЕКОНОМІЧНОГО АНАЛІЗУ

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи аналізу місцезнаходження об’єктів та його прогнозування. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти[23]. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для вибору методу прогнозування місцезнаходження об’єктів у контекстно-залежних системах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.2.1 ОБҐРУНТУВАННЯ ФУНКЦІЙ ПРОГРАМНОГО ПРОДУКТУ

Головна функція F_0 – розробка програмного продукту, який будує набір моделей прогнозу та перевіряє їх точність на певному наборі даних. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – використання готових бібліотек;

F_3 – вибір методу перевірки.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Python;

б) мова програмування Java;

Функція F_2:

а) написання алгоритмів вручну;

б) використання готових бібліотек;

Функція F_3:

а) перевірка точності на наборі даних для тестування;

б) 10-кратна перехресна перевірка.

4.2.2 ВАРІАНТИ РЕАЛІЗАЦІЇ ОСНОВНИХ ФУНКЦІЙ

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 10). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 8).

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП[24].

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам.

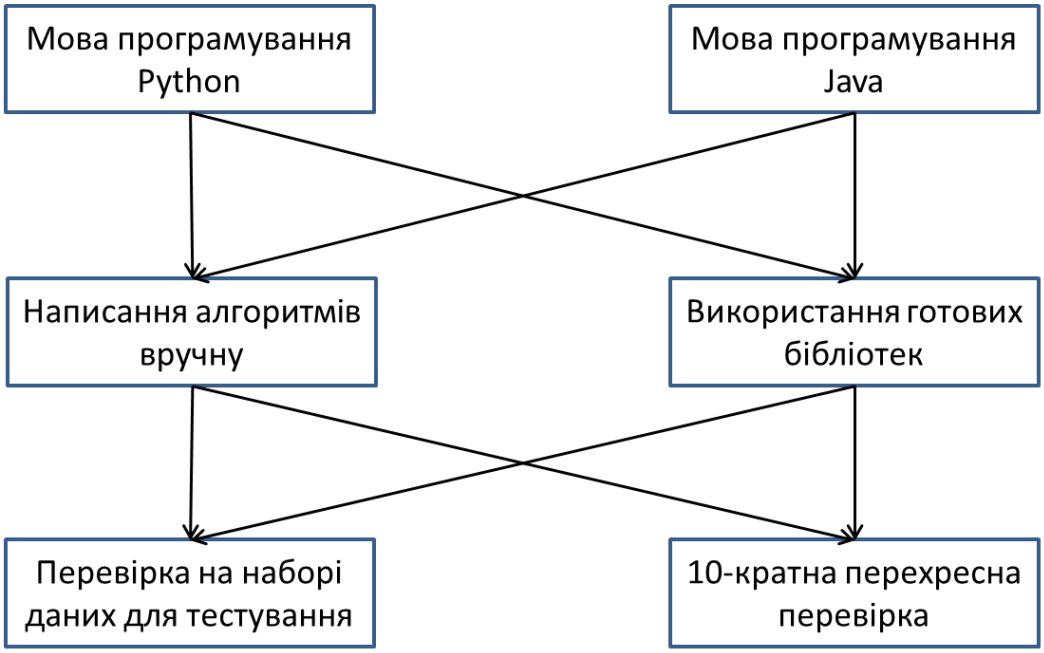

Рисунок 10 – Морфологічна карта

Таблиця 8 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Займає менше часу при написанні коду	Код повільніше виконується
	Б	Код швидше виконується	Займає більше часу при написанні коду
F2	А	Більша гнучність у використанні	Займає більше часу при написанні коду
	Б	Займає менше часу при написанні коду	Менша гнучкість у використанні
F3	*А*	Обчислювально легший	Оцінка точності не завжди відповідає реальній
	Б	Дає кращу оцінку точності	Обчислювально важчий

Функція *F1:*

Оскільки проводиться оцінка великої кількості методів, потрібно швидко їх реалізовувати, тому варіант б) має бути відкинутий.

Функція *F2:*

Оскільки методи написані вручну та за допомогою готових бібліотек будуть давати однакові результати, вважаємо варіанти а) та б) гідними розгляду.

Функція *F3:*

Точна оцінка важливіша для даної роботи, ніж складність обчислень, тому варіант а) має бути відкинутий.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3б

2. F1а – F2б – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 ОБҐРУНТУВАННЯ СИСТЕМИ ПАРАМЕТРІВ ПП

4.3.1 ОПИС ПАРАМЕТРІВ

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

4.3.2 КІЛЬКІСНА ОЦІНКА ПАРАМЕТРІВ

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 9.

Таблиця 9 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			Гірші	середні	кращі
Швидкодія мови програмування	X1	Оп/мс	2000	11000	19000
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки даних алгоритмом	X3	мс	800	420	60
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 9 будуються графічні характеристики параметрів – рис. 11 – рис. 14.

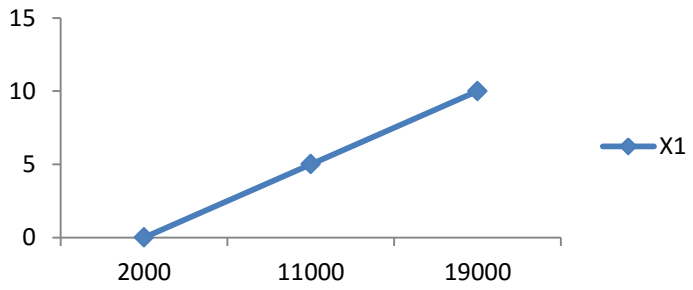

Рисунок 11 – X1, швидкодія мови програмування

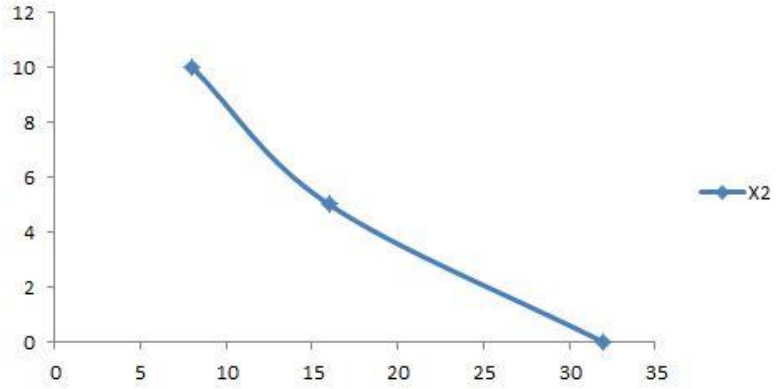

Рисунок 12 – X2, об’єм пам’яті для збереження даних

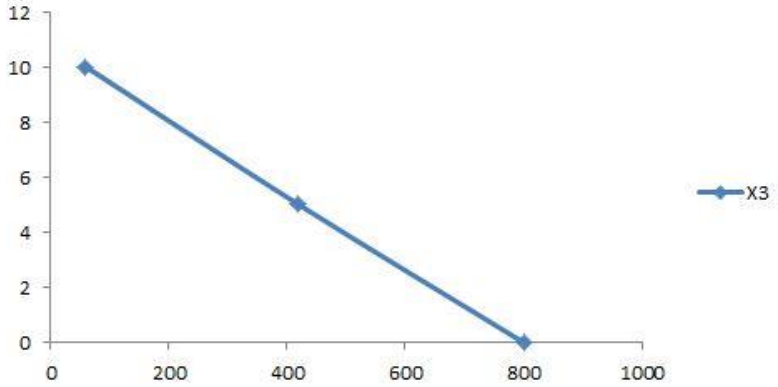

Рисунок 13 – X3, час обробки даних алгоритмом

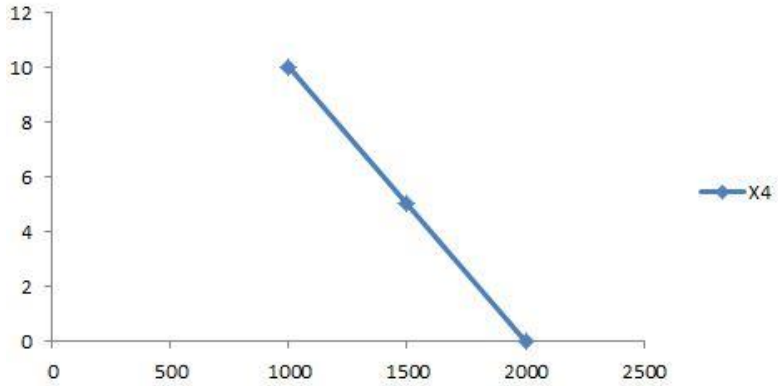

Рисунок 14 – X4, потенційний об’єм програмного коду

4.3.3 АНАЛІЗ ЕКСПЕРТНОГО ОЦІНЮВАННЯ ПАРАМЕТРІВ

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості[25].

Результати експертного ранжування наведені у таблиці 10.

Таблиця 10 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0.75	0.56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1.25	1.56
X3	Час обробки даних алгоритмом	Мс	2	2	1	2	1	2	2	12	-14.25	203.06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14.75	217.56
	Разом		15	15	15	15	15	15	15	105	0	420.75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26.25.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T.$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420.75$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420.75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0.67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 11.

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}$,де $b_i = \sum_{i=1}^{N} a_{ij}$.

Таблиця 11 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0.5
X1 і X3	<	<	<	<	<	<	<	<	0.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	>	>	>	>	>	>	>	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}$,де $b_i' = \sum_{i=1}^{N} a_{ij} b_j$.

Як видно з таблиці 12, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 12 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1.0	0.5	0.5	1.5	3.5	0.219	22.25	0.216	100	0.215
X2	1.5	1.0	0.5	1.5	4.5	0.281	27.25	0.282	124.25	0.283
X3	1.5	1.5	1.0	1.5	5.5	0.344	34.25	0.347	156	0.348
X4	0.5	0.5	0.5	1.0	2.5	0.156	14.25	0.155	64.75	0.154
Всього:					16	1	98	1	445	1

4.4 АНАЛІЗ РІВНЯ ЯКОСТІ ВАРІАНТІВ РЕАЛІЗАЦІЇ ФУНКЦІЙ

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об'єм пам'яті для збереження даних) *X1* (швидкодія мови програмування) та *X3* відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X4 (*потенційний об'єм програмного коду) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 1800 або варіанту б) 1200.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 13):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Таблиця 13 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3.6	0.215	0.774
F2(X3)	А, Б	800	2.4	0.348	0.835
F2(X4)	А	1800	2	0.154	0.308
	Б	1200	8	0.154	1.232
F3(X2)	Б	16	3.4	0.283	0.962

За даними з таблиці 13 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0.774 + 0.835 + 0.308 + 0.962= 2.879

K_{K2} = 0.774 + 0.835 + 1.232 + 0.962 = 3.803

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.5 ЕКОНОМІЧНИЙ АНАЛІЗ ВАРІАНТІВ РОЗРОБКИ ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

При цьому варіант 3 має додаткове завдання:

3. Реалізація методів аналізу;

А варіант 4 має інше додаткове завдання:

4. Обробка інтерфейсу готових бібліотек.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 2.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М},$$

де T_P – трудомісткість розробки ПП; $K_П$ – поправочний коефіцієнт; $K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість

дорівнює: T_P = 90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм другої групи складності, степінь новизни Б), тобто T_P = 27 людино-днів, $К_П$ = 0.9, $К_{СК}$ = 1, $К_{СТ}$ = 0.8:

$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Для третього завдання (використовується алгоритм другої групи складності, ступінь новизни Г з використанням перемінної інформації):

Тр = 12 людино-днів;

Кп = 0.72; Кст = 0.8;

То = 12 • 0.72 • 0,8 = 6.91.

Для четвертого завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

Тр = 8 людино-днів;

Кп = 0.6; Кст = 1;

То = 8 • 0.6 • 1 = 4.8.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (122.4 + 19.44 + 6.91) \cdot 8 = 1190$ людино-годин;

$T_{II} = (122.4 + 19.44 + 4.8) \cdot 8 = 1173.12$ людино-годин;

Найбільш високу трудомісткість має варіант І.

В розробці беруть участь два програмісти з окладом 6000 грн., один фінансовий аналітик з окладом 9000 грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн.,$$

де М – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$С_ч = \frac{6000 + 6000 + 9000}{3 \cdot 21 \cdot 8} = 41.67 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д,$$

де $С_ч$ – величина погодинної оплати праці програміста; $Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 41.67· 1190 · 1.2 = 59504.76 грн.

II. $С_{ЗП}$ = 41.67· 1173.12 · 1.2 = 58660.69 грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 59504.76· 0.22 = 13091.05 грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 58660.69 · 0.22 = 12905.35 грн.

Тепер визначимо витрати на оплату однієї машино-години($С_М$).

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0.2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З$ = 12 · 6000 · 0.2 = 14400 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З)$ = 14400 · (1 + 0.2)=17280 грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22$ = 17280 · 0.22 = 3801.6 грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706.4 \cdot 0.156 \cdot 0.2 \cdot 2.0218 = 107.64$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0.67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 17280 + 3801.6 + 2300 + 460 + 107.64 + 5360 = 29309.24$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС} / Т_{ЕФ} = 29309.24 / 1706.4 = 17.18$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$С_М = С_{М-Г} \cdot Т$

I. $С_М = 17.18 \cdot 1190 = 20444.2$ грн.;

II. $C_М = 17.18 \cdot 1173.12 = 20154.2$ грн.;

Накладні витрати складають 67% від заробітної плати:

$C_Н = C_{ЗП} \cdot 0.67$

I. $C_Н = 59504.76 \cdot 0.67 = 39868.19$ грн.;

II. $C_Н = 58660.69 \cdot 0.67 = 39302.66$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н$

I. $C_{ПП} = 59504.76 + 13091.05 + 20444.2 + 39868.19 = 132908.2$ грн.;

II. $C_{ПП} = 58660.69 + 12905.35 + 20154.2 + 39302.66 = 131022.9$ грн.;

4.6 ВИБІР КРАЩОГО ВАРІАНТА ПП ТЕХНІКО-ЕКОНОМІЧНОГО РІВНЯ

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$К_{ТЕРj} = К_{Кj} / С_{Фj}$,

$К_{ТЕР1} = 2.879 / 132908.2 = 0.22 \cdot 10^{-4}$;

$К_{ТЕР2} = 3.803 / 131022.9 = 0.29 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0.29 \cdot 10^{-4}$.

4.7 ВИСНОВОК

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0.29 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Python;

– використання готових бібліотек;

– 10-кратна перехресна перевірка.

ВИСНОВКИ

В результаті виконання даної роботи було здійснено аналіз алгоритмів прогнозування місцезнаходження об’єктів у контекстно-залежних системах управління електроспоживанням. Вхідними даними для таких алгоритмів є один із контекстних параметрів, а саме місцезнаходження. Алгоритми ж намагаються відтворити поведінкової моделі об’єктів.

Запропоновані алгоритми були налаштовані лише на класичному завданні прогнозування місцезнаходження і не мають додаткових джерел інформації, крім історії переміщень об’єктів. Не виключається, що впровадження додаткової інформації, в тому числі різнорідних сигналів від мобільних пристроїв користувача може позитивно впливати на точність відповідних прогнозів.

Найкращий з досліджених алгоритмів покращує точність прогнозу відносно базової моделі, яка прогнозує місце, яке найчастіше зустрічається, більш ніж на 30%. Такого результату досягає алгоритм випадкових лісів з підбором гіперпараметрів за допомогою сіткового пошуку. У разі обмежених обчислювальних ресурсів рекомендується використовувати метод дерев рішень з врахуванням підбору гіперпараметрів або ж метод опорних векторів. На основі таких методів, КЗСУЕ зможе:

1) аналізувати та прогнозувати місцезнаходження об’єктів, що забезпечить вплив на енерговитрати та комфорт користувачів;

2) містити відомості про поведінкові моделі користувачів, з подальшою можливості виділення спільностей та перетренування до типів користувачів, а не до кожного окремо;

У роботі висвітлена теоретична база кожного з алгоритмів та методів, а також показані можливі значення точності пре відтворенні даних процедур на реальних об'єктах. Для демонстрації цього був використаний Аугсбурзький Еталонний Тест Прогнозування Місцезнаходження, який містить практичні дані, зібрані з поведінкових моделей реальних осіб. Були наведені всі способи

тестування та підбору найкращих методів для вирішення класичної задачі прогнозування місцезнаходження.

Результатом роботи є готовий алгоритм прогнозування місцезнаходження, який досягає 80% точності прогнозу і може бути застосований у КЗСУЕ.

Так як КЗСУЕ набувають все більшого поширення, потенційні застосування даної роботи також розширюються та включають в себе:

1) розробку програмного забезпечення, що на основі прогнозу місцезнаходження дозволить виконувати визначені функції всередині будівель різного типу та призначення;

2) використання отриманих результатів як базової моделі для порівняння з новими підходами, методами та алгоритмами до прогнозування місцезнаходження;

Крім КЗСУЕ запропоновані методи можуть використовуватись у будь-яких системах в межах будівлі, які не обов'язково мають на меті мінімізацію електроспоживання, але вимагають автоматизацію певних процесів залежних від місця перебування об'єктів. Як приклад можна навести високонавантажений ліфт, який для економії часу користувачів заздалегідь враховує на яких поверхах його будуть викликати і підлаштовується під це.

Отже, при підборі гіперпараметрів за допомогою сіткового пошуку для методу випадкових лісів можна досягнути достатньої для використання точності прогнозування наступного місцезнаходження людини у певному інформаційному середовищі. Серед алгоритмів, які показали достатньо високі, але не найкращі результати варто відмітити метод опорних векторів, метод дерев рішень з підбором гіперпараметрів, метод стимулювання градієнта з підбором гіперпараметрів, а також голосування між методами опорних векторів, випадкових лісів та наївним баєсовим класифікатором, хоч останній метод і є досить затратним в плані обчислювальних ресурсів. Саме перелічені алгоритми рекомендовано для реалізації як частину контекстно-залежної системи управління електроспоживанням.

ПЕРЕЛІК ПОСИЛАНЬ

1. Yuan Hu An Ontology Based Context-Aware Model for Semantic Web Services / Yuan Hu, Xinke Li // Knowledge Acquisition and Modeling. - 2009. - №1. - С. 426-429.

2. Zhuikov V. Integration of context-aware control system in microgrid / Zhuikov V., Kyselova A. // Proceedings of Electronics and Nanotechnology (ELNANO) IEEE XXXIII International Scientific Conference. - 2013. – С. 386-390.

3. Hong J. Context-aware Systems: A Literature Review and Classification / Hong J., Suh E., Kim S. // Expert Systems with Applications . – 2008. – С. 12-34.

4. Жуйков В.Я. Метод принятия решений по управлению сетью с полупроводниковыми преобразователями электроэнергии / Жуйков В.Я., Киселев Г.Д., Киселева. А.Г. // Технічна електродинаміка. – 2014. - № 5. – С. 38-40.

5. Perera C. Context Aware Computing for The Internet of Things: A Survey / Perera C., Zaslavsky A., Christen P., Georgakopoulos D. // Communications Surveys & Tutorials, IEEE. – 2014. - № 16. – С. 414 – 454.

6. Gomes J.B. Where will you go? Mobile data mining for next place prediction / Gomes J.B., Phua C., Krishnaswamy S. // Bellatreche, L., Mohania, M.K. DaWaK LNCS. - 2013. - № 8057. – С. 146–158.

7. Lee Sang-Yong Users' Habits-based Context Prediction applied to Smart Buildings / Lee Sang-Yong, Do-Luong Tran // Proceedings of the World Congress on Engineering and Computer Science. – 2010. - № 1. – С. 120-131.

8. Jan Petzold Prediction of indoor movements using bayesian networks / Jan Petzold, Andreas Pietzowski, Faruk Bagci, Wolfgang Trumler, Theo Ungerer // Location-and Context-Awareness. Springer Berlin Heidelberg. - 2005. – С. 211-222.

9. Smart Doorplate : In First Inter-national Conference on Appliance Design (1AD) Bristol GB May 2003. – 2003. – С. 201-216.

10. Штучні нейронні мережі (НС) – Режим доступу: https://knhelp.wordpress.com/2012/04/19/л8-штучні-нейронні-мережі-нс/ - Дата доступу – 01.05.2016.

11. Дерево прийняття рішень - Режим доступу: https://uk.wikipedia.org/wiki/Дерево_прийняття_рішень - Дата доступу – 02.05.2016.

12. Наївний баєсів класифікатор - Режим доступу: https://uk.wikipedia.org/wiki/Наївний_баєсів_класифікатор - Дата доступу – 03.05.2016.

13. What is a hidden Markov model? - Режим доступу: http://www.nature.com/nbt/journal/v22/n10/full/nbt1004-1315.html - Дата доступу – 04.05.2016.

14. Метод опорных векторов - Режим доступу: http://www.statistica.ru/branches-maths/metod-opornykh-vektorov-supported-vector-machine-svm/ - Дата доступу – 05.05.2016.

15. Friedman, Jerome H. Greedy function approximation: a gradient boosting machine / Friedman, Jerome H. // Annals of statistics. – 2001. – С. 1189-1232.

16. Random forests - Режим доступу: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm - Дата доступу – 06.05.2016.

17. Ensemble learning - Режим доступу: https://en.wikipedia.org/wiki/Ensemble_learning - Дата доступу – 07.05.2016.

18. Petzold J. Augsburg Indoor Location Tracking Benchmarks / Petzold J. // Technical Report, Institute of Computer Science, University of Augsburg, Germany. – 2004. – С. 1-10.

19. Hackeling Gavin. Mastering Machine Learning with scikit-learn / Hackeling Gavin. - Packt Publishing Ltd. - 2014. – С. 1-32.

20. Яловий Г.К. Економіка та організація виробництва / Яловий Г.К., Пашін В.П., Сичов В.С. - К.: "Політехніка". - 2004. – С. 80.

21. В.Є. Богданюк. Методичні вказівки до виконання організаційно-

економічного розділу дипломних проектів / В.Є. Богданюк, К.В. Березовський, В.П. Пашін - К.: НТУУ "КПІ". - 1999. - С. 66.

22. Долинская М.Г. Маркетинг и конкурентоспособность промышленной продукции / Долинская М.Г., Соловьев И.А. - М.: «Стандарт». - 1991. - С. 125.

23. Пашин В.П. Функционально-стоимостный анализ конструкторско-технологических решений / Пашин В.П. - К.: РДЭНТП «Знание» УССР. - 1989. – С. 22.

24. Пашін В.П. Оцінка конкурентоспроможності електронних пристроїв на стадії проектування / Пашин В.П. - К. Економічний вісник НТУУ „КПІ". - 2006. - С. 252-255.

25. Пашин В.П. Управление качеством изделий на основе функционально-стоимостного анализа / Пашин В.П. - К.: «Технология и организация производства». - 1989. - С. 17-19.

ДОДАТОК А

```
import csv
import pandas as pd
import numpy as np
from sklearn import cross_validation
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sknn.mlp import Classifier, Layer
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import GradientBoostingClassifier
from seqlearn.hmm import MultinomialHMM
from sklearn.grid_search import GridSearchCV



def merge_fall_summer(fallfile, summerfile, resultfile):
    """
    Merges summer and fall data from Augsburg Location Tracking Benchmark
    :param fallfile: String with fall data filepath
    :param summerfile: String with summer data filepath
    :param resultfile: String with filepath to save merged data to
    """
    filenames = [fallfile, summerfile]
    with open(resultfile, 'w') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(['time', 'location', 'person', 'timestamp'])
        for fname in filenames:
            with open(fname) as infile:
                for line in infile:
                    processed = line.split(';')
                    processed[3] = processed[3][:-2]
                    print(processed)
                    writer.writerow(processed)
```

```
# Merging process
merge_fall_summer("augsburger/a_fall.data", "augsburger/a_summer.data", "preprocessed/a.csv")
merge_fall_summer("augsburger/b_fall.data", "augsburger/b_summer.data", "preprocessed/b.csv")
merge_fall_summer("augsburger/c_fall.data", "augsburger/c_summer.data", "preprocessed/c.csv")
merge_fall_summer("augsburger/d_fall.data", "augsburger/d_summer.data", "preprocessed/d.csv")



def encode(series):
    """
    Encodes the location values with values 0 - n, where n is the number of classes
    :param series: [String] location values
    :return: [Int] encoded data
    """
    dict = {}
    counter = 0
    for i in series:
        if i not in dict:
            dict[i] = counter
            counter += 1
    encoded = [dict[i] for i in series]
    return encoded



def sliding_window(series, window=1):
    """
    Creates windows of fixed time steps with labels
    :param series: [Int] initial data
    :param window: Int number of time steps to consider
    :return: [([Int], Int)] pairs of features and labels
    """
    windows = []
    for i in range(window, len(series)):
        windows.append((series[i - window:i], series[i]))
    return windows
```

```
def eval_random_forest(x_data, y_data, tune=False):
    """
    Evaluates random forest on the given data
    :param x_data: [[Int]] features
    :param y_data: [Int] labels
    :param tune: Bool whether to use hyperparameter tuning
    """
    if tune:
        tuned_parameters = {'n_estimators': [5, 10, 15, 20],
                            "criterion": ["gini", "entropy"],
                           "min_samples_split": [2, 10, 20],
                           "max_depth": [None, 2, 5, 10],
                           "min_samples_leaf": [1, 5, 10],
                           "max_leaf_nodes": [None, 5, 10, 20]}
        clf = GridSearchCV(RandomForestClassifier(), tuned_parameters, cv=10)
        clf.fit(x_data, y_data)
        print("RF:")
        print("Best parameters set found on development set:")
        print()
        print(clf.best_params_)
        print("Best score:")
        print(clf.best_score_)
    else:
        clf = RandomForestClassifier()
        scores = cross_validation.cross_val_score(clf, x_data, y_data, cv=10)
        print("Rf", np.mean(scores))



def eval_neural_network(x_data, y_data):
    """
    Evaluates neural network on the given data
    :param x_data: [[Int]] features
    :param y_data: [Int] labels
```

```
    """
    clf = Classifier(
            layers=[
               Layer("Maxout", units=100, pieces=2),
               Layer("Softmax")],
            learning_rate=0.1,
            n_iter=25)
    x_data = np.array(x_data)
    y_data = np.array(y_data)
    scores = cross_validation.cross_val_score(clf, x_data, y_data, cv=10)
    print("NN", np.mean(scores))



def eval_decision_tree(x_data, y_data, tune=False):
    """
    Evaluates decision trees on the given data
    :param x_data: [[Int]] features
    :param y_data: [Int] labels
    :param tune: Bool whether to use hyperparameter tuning
    """
    if tune:
        tuned_parameters = {"criterion": ["gini", "entropy"],
             "min_samples_split": [2, 10, 20],
             "max_depth": [None, 2, 5, 10],
             "min_samples_leaf": [1, 5, 10],
             "max_leaf_nodes": [None, 5, 10, 20],
             }
        clf = GridSearchCV(DecisionTreeClassifier(), tuned_parameters, cv=10)
        clf.fit(x_data, y_data)
        print("DT:")
        print("Best parameters set found on development set:")
        print()
        print(clf.best_params_)
        print()
        print("Best score:")
```

```
        print(clf.best_score_)
    else:
        clf = DecisionTreeClassifier()
        scores = cross_validation.cross_val_score(clf, x_data, y_data, cv=10)
        print("DT", np.mean(scores))


def eval_support_vector_machine(x_data, y_data, tune=False):
    """
    Evaluates support vector machine on the given data
    :param x_data: [[Int]] features
    :param y_data: [Int] labels
    :param tune: Bool whether to use hyperparameter tuning
    """
    if tune:
        tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-2, 0.1, 0.25, 0.5, 0.9],
                             'C': [1, 10, 100, 1000]},
                            {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
        clf = GridSearchCV(SVC(probability=True), tuned_parameters, cv=10, verbose=1)
        clf.fit(x_data, y_data)
        print("SVM:")
        print("Best parameters set found on development set:")
        print()
        print(clf.best_params_)
        print()
        print("Best score:")
        print(clf.best_score_)
    else:
        clf = SVC(probability=True)
        scores = cross_validation.cross_val_score(clf, x_data, y_data, cv=10, n_jobs=4)
        print("SVC", np.mean(scores))


def eval_naive_bayes(x_data, y_data, tune=False):
    """
```

```
    Evaluates naive bayes on the given data
    :param x_data: [[Int]] features
    :param y_data: [Int] labels
    :param tune: Bool whether to use hyperparameter tuning
    """
    if tune:
        tuned_parameters = {'alpha': [0.0, 0.25, 0.5, 0.75, 1.0]}
        clf = GridSearchCV(MultinomialNB(), tuned_parameters, cv=10, verbose=1)
        clf.fit(x_data, y_data)
        print("NB:")
        print("Best parameters set found on development set:")
        print()
        print(clf.best_params_)
        print("Best score:")
        print(clf.best_score_)
    else:
        clf = MultinomialNB()
        scores = cross_validation.cross_val_score(clf, x_data, y_data, cv=10)
        print("NB", np.mean(scores))


def eval_voting(x_data, y_data):
    """
    Evaluates voting between nb, svm and rf on the given data
    :param x_data: [[Int]] features
    :param y_data: [Int] labels
    """
    clf1 = RandomForestClassifier()
    clf2 = MultinomialNB()
    clf3 = SVC(probability=True)
    eclf = VotingClassifier(estimators=[('rf', clf1), ('nb', clf2), ('svc', clf3)], voting='soft')
    scores = cross_validation.cross_val_score(eclf, x_data, y_data, cv=10)
    print("Voting", np.mean(scores))
```

```
def eval_gradient_boosting(x_data, y_data, tune=False):
    """
    Evaluates gradient boosting on the given data
    :param x_data: [[Int]] features
    :param y_data: [Int] labels
    :param tune: Bool whether to use hyperparameter tuning
    """
    if tune:
        tuned_parameters = {
                        "min_samples_split": [2, 10, 20],
                        "max_depth": [None, 2, 5, 10],
                        "min_samples_leaf": [1, 5, 10],
                        "max_leaf_nodes": [None, 5, 10, 20]}
        clf = GridSearchCV(GradientBoostingClassifier(), tuned_parameters, cv=10)
        clf.fit(x_data, y_data)
        print("GB:")
        print("Best parameters set found on development set:")
        print()
        print(clf.best_params_)
        print("Best score:")
        print(clf.best_score_)
    else:
        clf = GradientBoostingClassifier()
        scores = cross_validation.cross_val_score(clf, x_data, y_data, cv=10)
        print("GB", np.mean(scores))



def eval_HMM(x_data, y_data):
    """
    Evaluates hidden markov models on the given data
    :param x_data: [[Int]] features
    :param y_data: [Int] labels
    """
    clf = MultinomialHMM()
    x_train = x_data[:int(0.7 * len(x_data))]
```

```
    y_train = y_data[:int(0.7 * len(x_data))]
    x_test = x_data[int(0.7 * len(x_data)):]
    y_test = y_data[int(0.7 * len(x_data)):]
    clf.fit(x_train, y_train, [len(x_data)])
    y_pred = clf.predict(x_test)
    err = 0
    for i in range(len(x_test)):
        if y_pred[i] == y_test[i]:
            err += 1
    print("HMM", err / len(x_test))


# Main script
people = ["preprocessed/a.csv", "preprocessed/b.csv", "preprocessed/c.csv", "preprocessed/d.csv"]
for person in people:
    print("Person: %s" % person[-5])
    p = pd.read_csv(person)
    series = p["location"].values
    encoded = encode(series)
    data = sliding_window(encoded, window=4)
    x_data = [i for (i, _) in data]
    y_data = [i for (_, i) in data]
    print(y_data.count(0))
    print("Baseline accuracy: %f" % (float(y_data.count(0)) / len(y_data)))
    eval_decision_tree(x_data, y_data, True)
    eval_neural_network(x_data, y_data)
    eval_random_forest(x_data, y_data, True)
    eval_support_vector_machine(x_data, y_data, True)
    eval_HMM(x_data, y_data)
    eval_adaboost(x_data, y_data)
    eval_naive_bayes(x_data, y_data, True)
    eval_voting(x_data, y_data)
    eval_gradient_boosting(x_data, y_data, True)
```