Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)
факультет інформаційних технологій
(факультет)
Кафедра інформаційних систем та технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра
(бакалавра, спеціаліста, магістра)

Студента Ястребцева В'ячеслава Дмитровича
(ПІБ)
академічної групи 126м-20-1
(шифр)
спеціальності 126 « Інформаційні системи та технології»
(код і назва спеціальності)
за освітньо-професійною програмою

«Інформаційні системи та технології»
(офіційна назва)
на тему Розробка інформаційної системи моніторингу захворюваності на COVID-19 з використанням платформи Node.js
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Сергєєва К.Л.			

Рецензент	д.т.н., проф. Алексєєв Михайло Олександрович			

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри

інформаційних технологій
та комп'ютерної інженерії

(повна назва)

_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

«____»_____2022 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістр

(бакалавра, спеціаліста, магістра)

студенту Ястребцеву В.Д. **академічної групи** 126м-20-1

(прізвище та ініціали) (шифр)

спеціальності 126 « Інформаційні системи та технології »

за освітньою-професійною програмою _____

«Інформаційні системи та технології»

на тему Розробка інформаційної системи моніторингу захворюваності на COVID-19 з використанням платформи Node.js

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ № 2241-Л

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз стану області рішень	1.10.2021 – 31.10.2021
Розділ 2	Моделі на методи розв'язання задачі	1.11.2021 – 30.11.2021
Розділ 3	Розробка інформаційної системи моніторингу захворюванності на COVID-19	1.12.2021 – 21.12.2021

Завдання видано _____ Сергєєва К.Л.
(підпис керівника) (прізвище, ініціали)

Дата видачі 1.10.2021 р.

Дата подання до екзаменаційної комісії 23.12.2021 р.

Прийнято до виконання _____ Ястребцев В.Д.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 90 стор., 40 рис., 5 додатків, 27 джерел.

Об'єкт дослідження: методи побудування інформаційної системи моніторингу і аналізу даних.

Предмет дослідження: інформаційна система моніторингу захворюваності на COVID-19.

Мета магістерської роботи: створити інформаційну систему аналізу й візуалізації особливостей просторово-часової динаміки показників захворюваності на COVID-19, розробити технологію аналізу захворюваності за допомогою нечіткої логіки.

У вступі подано стан проблеми та виконана постановка задачі дослідження.

Перший розділ присвячено аналізу теми дослідження. Наведено огляд відкритих джерел та існуючі додатки аналізу даних.

В другому розділі наведено проектну складову вирішення завдання.

Розглянуто можливості автоматизованого збору даних, формування бази даних, а також методи прогнозування часових рядів і просторово-часова кластеризація за множиною показників.

Третій розділ присвячено розробці інформаційної системи на мові програмування Javascript та прогнозування розповсюдження хвороби. Виконано реалізацію інформаційної системи моніторингу захворюваності на COVID-19.

Практична цінність результатів полягає у тому, що розроблена інформаційна система розширює границі процесу моніторингу захворюваності на COVID-19 в Україні, шляхом додавання доступного цифрового сервісу на українській мові, також унікальність роботі додає обраний метод для прогнозування захворюваності – аналіз за допомогою нечіткої логіки.

МОВА JAVASCRIPT, ПЛАТФОРМА NODE.JS, ЦИФРОВІ СЕРВІСИ, НЕЧІТКА ЛОГІКА, ХМАРНІ СЕРВІСИ, КОМПЛЕКС ПУБЛІЧНИХ ПОСЛУГ

ABSTRACT

Explanatory note: 86 pages, 40 figures, 5 appendices, 28 sources.

Object of research: methods of building an information system for monitoring and data analysis.

Subject of research: COVID-19 incidence monitoring information system.

The purpose of the master's work: creation of the information system of the analysis and visualization of features of space-time dynamics of indicators of morbidity on COVID-19, to develop new technology of the analysis of morbidity by means of fuzzy logic.

The introduction presents the state of the problem and the task of research.

The first section is devoted to the analysis of the research topic. An overview of open sources and existing data analysis applications is provided.

The second section presents the project component of the problem.

Possibilities of automated data collection, database formation, as well as methods of time series forecasting and spatio-temporal clustering by a set of indicators are considered.

The third section is devoted to the development of an information system in the Javascript programming language and forecasting the spread of the disease. The implementation of the information system for monitoring the incidence of COVID-19 has been completed.

The practical value of the results is that the developed information system expands the boundaries of the process of monitoring the incidence of COVID-19 in Ukraine by adding an available digital service in Ukrainian.

JAVASCRIPT LANGUAGE, NODE.JS PLATFORM, DIGITAL SERVICES, FUZZY LOGIC, CLOUD SERVICES, PUBLIC SERVICES COMPLEX

Зміст

Перелік умовних позначень

БД – база даних

ІС – інформаційна система

ІТ – інформаційні технології

ООП – об'єктно-орієнтоване програмування

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

ЧР – часові ряди

API – application programming interface

HTTP – Hyper Text Transfer Protocol

REST – Representational State Transfer — “передача стану уявлення” — архітектурний стиль взаємодії компонентів

COVID-19 – CoronaVirus Disease 2019 – інфекційна хвороба, яка була виявлена у людини в грудні 2019 року

Fuzzy logic – Нечітка логіка

ВСТУП

Нещодавно новий вірус під назвою «коронавірус» або відомий як «COVID-19» є предметом, який інтенсивно вивчається. SARS-CoV-2 – це велика родина вірусів, які можуть викликати смертельні проблеми зі здоров'ям у людей. Тому є необхідність створити систему моніторингу захворюванності, яка у реальному часі надає інформацію щодо поточного стану розповсюдження хвороби, яка б давала можливість не тільки можливість оцінити стан ситуації а також і спрогнозувати майбутні обставини.

ML використовується в різних галузях, включаючи медицину, щоб передбачити хворобу та прогнозувати її результати. У медицині правильний діагноз і правильний час є запорукою успішного лікування. Дослідники почали використовувати програми штучного інтелекту для передбачення розповсюдження хвороби.

Для передбачення та прогнозування майбутніх подій використовуються різні методи ML. Для передбачення використовуються наступні методи ML: машина опорних векторів, лінійна регресія, логістична регресія, наївний Байєс, дерева рішень, K-найближчий сусід і нейронні мережі. Аналогічно, деякі методи ML, які використовуються для прогнозування майбутніх подій — це наївний підхід, ковзне середнє, просте експоненціальне згладжування, модель лінійного тренду Холта, модель Холта-Вінтерса, сезонна авторегресивна інтегрована екскогенна модель ковзного середнього і авторегресивна інтегрована модель ковзного середнього .

Кожна методика має унікальні особливості та використовується по-різному залежно від результатів точності. Для передбачення або прогнозування вибирається модель з найкращою точністю під час процесу оцінки моделі. Таким чином, для новизни роботи було вирішено обрати метод нечіткої логіки для прогнозування розповсюдження хвороби.

Ціль роботи полягає у створенні інформаційної системи моніторингу захворюваності на COVID-19 з використанням платформи Node.js.
Відповідно до мети й предмета дослідження у кваліфікаційній роботи необхідно вирішити наступні завдання:

- дослідити відкриті дані щодо захворювання на COVID-19;
- дослідити принцип побудови інформаційних систем моніторингу захворюваності на COVID-19;
- розробити програмне забезпечення інформаційної системи моніторингу захворюваності на COVID-19;
- дослідити існуючі методи прогнозування;
- розробити автоматизоване програмне забезпечення аналізу та прогнозування даних на COVID-19

РОЗДІЛ 1
АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕНЬ

1.1 Відкриті статистичні дані захворюваності на COVID-19

Останнім часом на просторах інтернету з’явилося багато джерел для отримання даних по захворюванню на COVID-19, вони відрізняються параметрами даних, обсягом та форматом отримання цих даних. Розберемо наступні джерела:

1) https://data.humdata.org

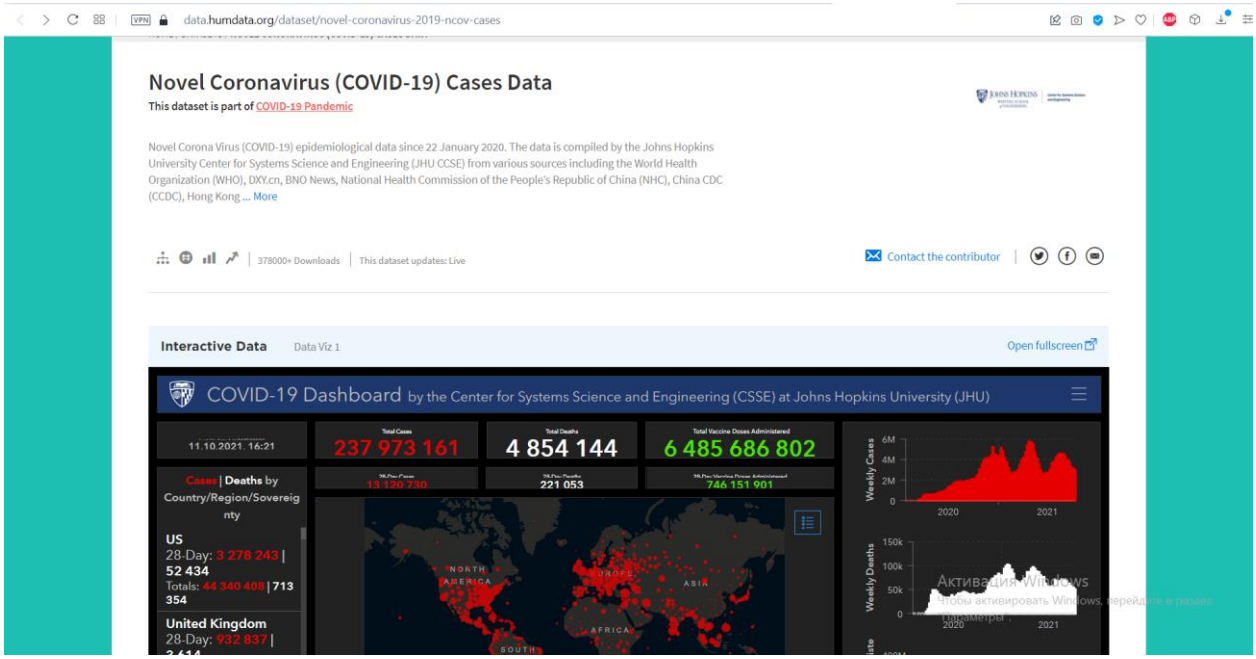

Рис. 1.1 Сайт Data Humdata

Цей сервіс надає можливість отримувати історію по випадкам Covid-19 в реальному часі(оновлюється кожен день). Інформація надається сервісом через файл CSV. Але, нажаль він надає лише інформацію про дату, країну та координати нового випадка, для моєї роботи цих даних буде замало.

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20
1		Afghanistan	33.93911	67.709953	0	0	0	0	0	0	0	0
2		Albania	41.1533	20.1683	0	0	0	0	0	0	0	0
3		Algeria	28.0339	1.6596	0	0	0	0	0	0	0	0
4		Andorra	42.5063	1.5218	0	0	0	0	0	0	0	0
5		Angola	-11.2027	17.8739	0	0	0	0	0	0	0	0
6		Antigua and Barbuda	17.0608	-61.7964	0	0	0	0	0	0	0	0
7		Argentina	-38.4161	-63.6167	0	0	0	0	0	0	0	0
8		Armenia	40.0691	45.0382	0	0	0	0	0	0	0	0
9	Australian Capital Territory	Australia	-35.4735	149.0124	0	0	0	0	0	0	0	0
10	New South Wales	Australia	-33.8688	151.2093	0	0	0	0	3	4	4	4
11	Northern Territory	Australia	-12.4634	130.8456	0	0	0	0	0	0	0	0
12	Queensland	Australia	-27.4698	153.0251	0	0	0	0	0	0	0	1
13	South Australia	Australia	-34.9285	138.6007	0	0	0	0	0	0	0	0
14	Tasmania	Australia	-42.8821	147.3272	0	0	0	0	0	0	0	0
15	Victoria	Australia	-37.8136	144.9631	0	0	0	0	1	1	1	1
16	Western Australia	Australia	-31.9505	115.8605	0	0	0	0	0	0	0	0

Рис. 1.2 Приклад даних з сервісу Humdata

2) United Nations Department of Economic and Social Affairs

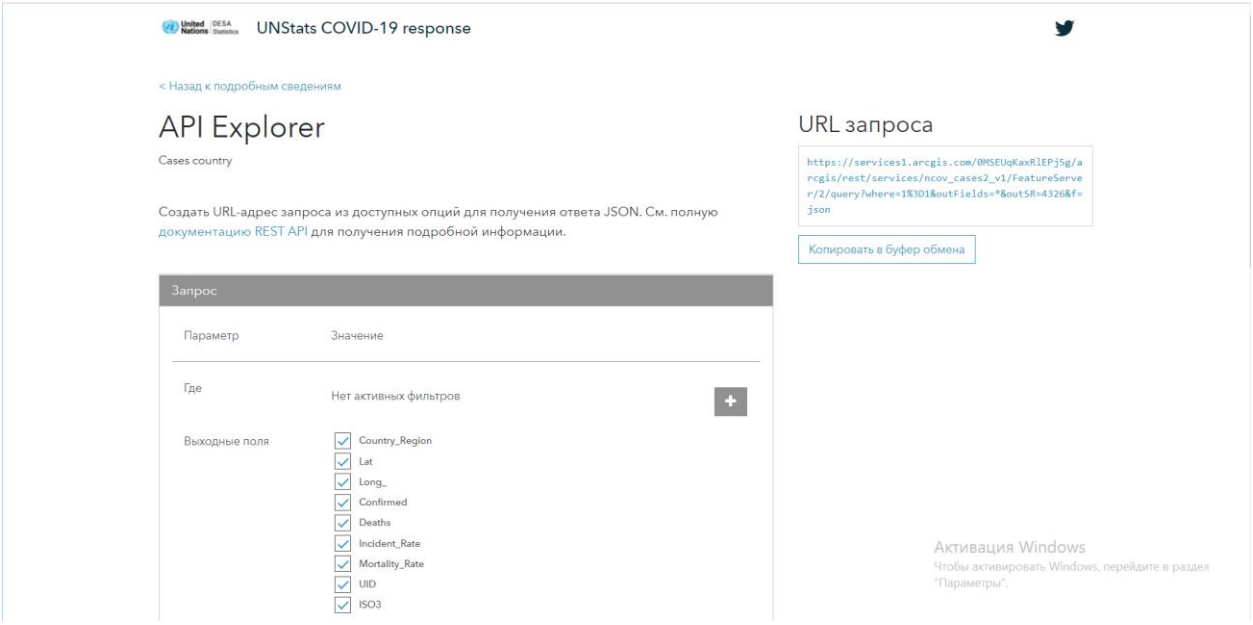

Рис. 1.3 Сайт UNStats

Сайт Unstats надає дозвіл на використання даних, що стосуються COVID-19, ці дані доступні у вигляді веб-сервісів геопросторових даних, придатних для виробництва карт та інших візуалізацій та аналізу даних, а також простих для завантаження у різних форматах. Інформація надається за допомогою REST API.

Сервіс надає більшу інформацію ніж попередній, виглядає вона наступним образом:

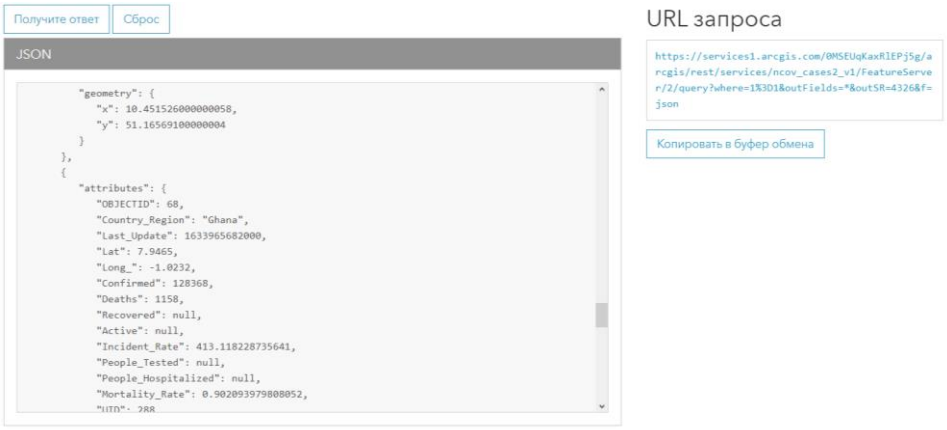

Рис. 1.4 Приклад даних з сервісу UNStats

3) Postman

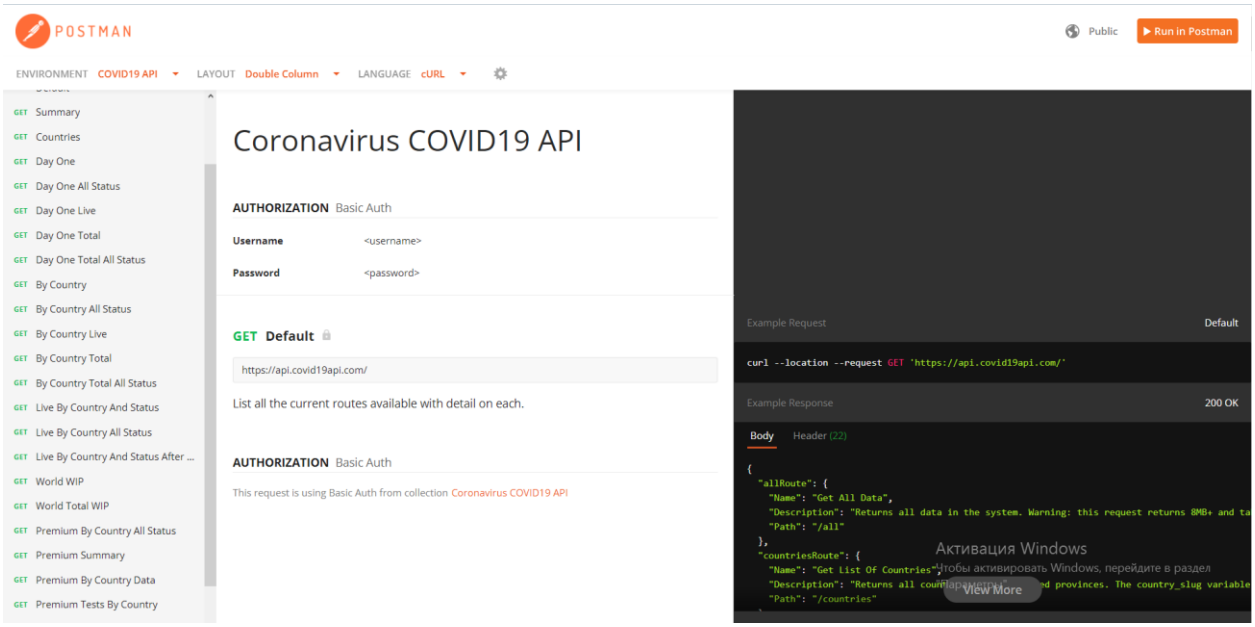

Рис. 1.5 Документація сервісу Postman

API поточних та історичних даних щодо коронавірусу розроблений командою Cloudeya Limited, щоб допомогти вченим, державним установам, медичним працівникам та громадськості ефективно зрозуміти та реагувати на коронавірус.

Так само незалежні розробники та компанії можуть використовувати його для створення інтерактивної інформаційної панелі візуалізації даних, додатків (мобільних та веб -сторінок) та інтеграції їх у свої внутрішні інструменти.


```
[
  {
    "Country": "Switzerland",
    "CountryCode": "CH",
    "Lat": "46.82",
    "Lon": "8.23",
    "Cases": 0,
    "Status": "confirmed",
    "Date": "2020-01-22T00:00:00Z"
  },
  {
    "Country": "Switzerland",
    "CountryCode": "CH",
    "Lat": "46.82",
    "Lon": "8.23",
    "Cases": 0,
    "Status": "confirmed",
    "Date": "2020-01-23T00:00:00Z"
  },
```


Рис. 1.6 Приклад даних з сервісу Postman

4) https://ourworldindata.org

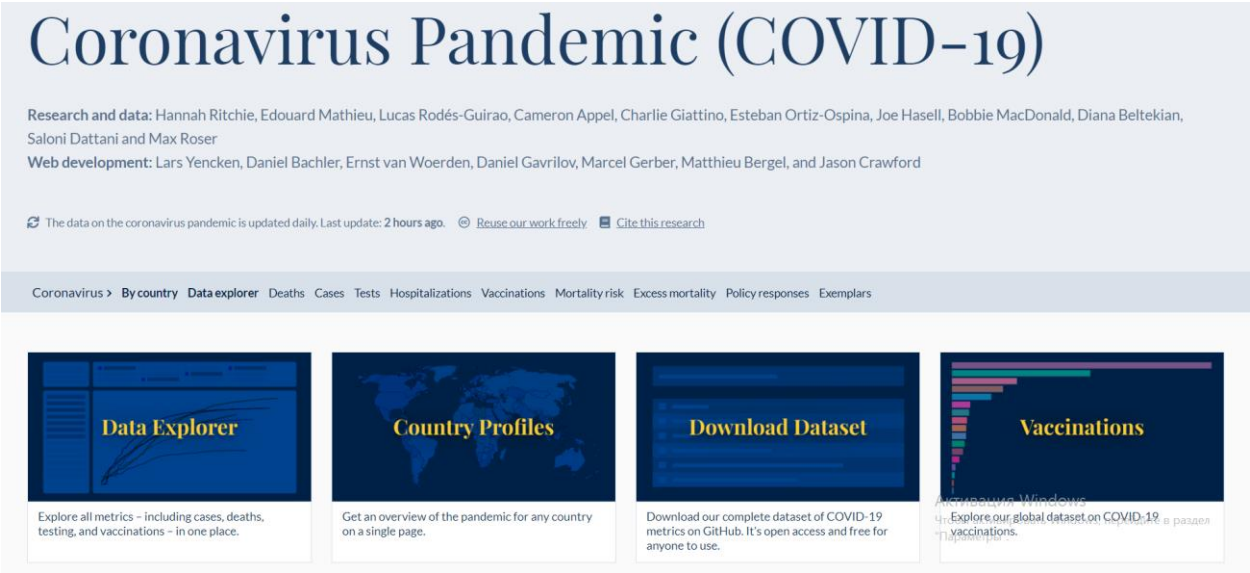

Рис. 1.7 Сайт Our world in data

Компанія створила 207 профілів країн, які дозволяють вивчити статистику пандемії коронавірусу для кожної країни світу.

У умовах пандемії, що швидко розвивається, нелегко визначити країни, які досягли найбільшого успіху в боротьбі з нею. Для всебічної оцінки вони відстежують вплив пандемії та формують профілі країн для 207 країн, щоб детально вивчити статистику пандемії коронавірусу для кожної країни світу.

Кожен профіль містить інтерактивні візуалізації, пояснення представлених показників та подробиці щодо джерел даних. Профіль кожної країни оновлюється щодня.

	column	source	category	description
2	iso_code	International Organization for Standardization	Others	ISO 3166-1 alpha-3
3	continent	Our World in Data	Others	Continent of the ge
4	location	Our World in Data	Others	Geographical locati
5	date	Our World in Data	Others	Date of observation
6	total_cases	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed cases	Total confirmed cas
7	new_cases	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed cases	New confirmed cas
8	new_cases_smoothed	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed cases	New confirmed cas
9	total_deaths	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed deaths	Total deaths attribu
10	new_deaths	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed deaths	New deaths attribut
11	new_deaths_smoothed	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed deaths	New deaths attribut
12	total_cases_per_million	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed cases	Total confirmed cas
13	new_cases_per_million	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed cases	New confirmed cas
14	new_cases_smoothed_per_million	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed cases	New confirmed cas
15	total_deaths_per_million	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed deaths	Total deaths attribu
16	new_deaths_per_million	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed deaths	New deaths attribut
17	new_deaths_smoothed_per_million	COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University	Confirmed deaths	New deaths attribut
18	reproduction_rate	Arroyo Marioli et al. (2020). https://doi.org/10.2139/ssrn.3581633	Reproduction rate	Real-time estimate

Рис. 1.8 Приклад даних з сервісу Our world in data

1.2 Існуючі програмні додатки аналізу й візуалізації даних

1.2.1 Огляд програмних платформ та їх функціональних можливостей

Важливість аналітики даних та попит на ринку породили багату кількість вакансій по всьому світу. Скласти список кращих інструментів та методів для аналізу даних стає нелегко, так як інструменти з відкритим вихідним кодом більш популярні, зручні і орієнтовані на продуктивність, чого не сказати про платні версії. Є багато інструментів з відкритим

вихідним кодом, які не вимагають будь-якого кодування і забезпечують хороші результати, ніж платні версії, наприклад – якщо говорити про програмування то це мова R = Tableau public для видобутку даних та Python для візуалізації даних.

Нижче приведено список з 10 кращих методів та інструментів для аналізу даних, як з відкритим вихідним кодом, так і платних версій, заснований на їх популярності, вивченості та продуктивності.

1) R

Рис. 1.9 Логотип мови R

R - провідний аналітичний інструмент в галузі, широко використовуваний для статистики і моделювання даних. З його допомогою можна легко маніпулювати даними і представляти їх в різних формах. Він перевершив SAS за багатьма параметрами, таким як ємність даних, продуктивність і результати. R компілюється і працює практично на всіх платформах починаючи з UNIX та закінчуючи MacOS. Він містить больше 10 тисяч пакетів і дозволяє переглядати пакети за категоріями. R також надає інструменти для автоматичного встановлення всіх пакетів відповідно до вимог користувача, що також може бути плюсом при роботі з Великими даними.

2) Tableau Public

Рис. 1.10 Логотип Tableau Public

Tableau Public - це безкоштовне програмне забезпечення, яке підключає будь-яке джерело даних, будь то корпоративне сховище даних, Microsoft Excel або веб-дані, і створює візуалізації даних, карти, приладові панелі і т.д. з поданням оновлень в реальному часі в мережі Інтернет. Ними також можна поділитися через соціальні мережі або з клієнтом. Він дозволяє завантажувати файли в різних форматах. Якщо ви хочете побачити всю міць Tableau, то у вас повинен бути дуже якісне джерело даних.

Можливості Tableau по роботі з великими даними роблять їх зручними, і можна аналізувати і візуалізувати дані краще, ніж будь-яке інше програмне забезпечення для візуалізації даних на ринку.

3) Python

Рис. 1.11 Логотип мови Python

Python - це об'єктно-орієнтована мова сценаріїв, яку легко читати, писати, підтримувати і є безкоштовним інструментом з відкритим вихідним

кодом. Він був розроблений Гвідо ван Россум в кінці 80-х років і підтримує як функціональні, так і структурні методи програмування.

Python легко вивчати, так як він дуже схожий на JavaScript, Ruby і PHP. Крім того, в Python є дуже хороші бібліотеки машинного навчання: Scikitlearn, Theano, Tensorflow і Keras. Ще однією важливою особливістю Python є те, що його можна зібрати на будь-якій платформі, наприклад, на SQL-сервері, базі даних MongoDB або JSON. Python також відмінно справляється з текстовими даними.

4) SAS

Рис. 1.12 Логотип SAS

Sas - це середовище програмування і мова для маніпулювання даними та лідер в області аналітики, розроблений SAS Institute в 1966 році і отримав подальший розвиток в 1980-х і 1990-х роках. SAS легко доступний, керований і може аналізувати дані з будь-яких джерел. У 2011 році SAS представила великий набір продуктів для аналізу клієнтів і численні модулі SAS для аналітики веб-сайтів, соціальних мереж і маркетингу, які широко використовуються для складання профілів клієнтів і перспектив. Він також може прогнозувати їхню поведінку, управляти і оптимізувати комунікації.

5) Apache Spark

Рис. 1.13 Логотип Apache Spark

Apache Spark - це швидкий механізм обробки великомасштабних даних, що виконує додатки в кластерах Hadoop в 100 разів швидше в пам'яті і в 10 разів швидше на диску. Spark побудований на науці про дані, і його концепція робить науку про дані легкою. Spark також популярний для конвеєрів даних і розробки моделей машинного навчання. Лабораторія AMP Каліфорнійського університету в Берклі розробила Apache в 2009 році.

Spark включає бібліотеку - MLlib, яка надає прогресивний набір машинних алгоритмів для повторюваних методів науки про дані, таких як класифікація, регресія, коллаборативна фільтрація, кластеризація і т.д.

6) Excel

Рис. 1.14 Логотип Excel

Excel - це базовий, популярний і широко використовуваний аналітичний інструмент практично у всіх галузях. Незалежно від того, чи є ви експертом в Sas, R або Tableau, вам все одно доведеться використовувати Excel. Excel стає важливим, коли виникає потреба в аналітиці внутрішніх даних клієнта. Він аналізує складну задачу, узагальнюючи дані з попереднім переглядом поворотних таблиць, які допомагають в фільтрації даних відповідно до вимог клієнта.

Excel має розширені можливості бізнес-аналітики, які допомагають в моделюванні даних з попередньо вбудованими опціями, такими як автоматичне визначення взаємозв'язків, створення заходів DAX і угруповання за часом.

7) RapidMiner

Рис. 1.15 Логотип RapidMiner

RapidMiner - це потужна інтегрована платформа науки про дані, розроблена тією ж компанією, яка виконує передбачуванний аналіз і інші види розширеної аналітики, такі як видобуток даних, текстова аналітика, машинне навчання і візуальна аналітика, без будь-якого програмування. RapidMiner може працювати з будь-якими типами джерел даних, включаючи Access, Excel, Dbase Microsoft SQL, Tera data, Oracle, Sybase, IBM DB2, Ingres, IBM SPSS, MySQL та ін. Інструмент є дуже потужним і може

генерувати аналітику на основі реальних параметрів перетворення даних, тобто ви можете контролювати формати і набори даних для інтелектуального аналізу.

8) KNIME

Рис. 1.16 Логотип KNIME

KNIME Розроблено в січні 2004 року командою інженерів-програмістів з Університету Констанца. KNIME є провідним відкритим вихідним кодом, інструментом звітності та інтегрованої аналітики, який дозволяє аналізувати і моделювати дані за допомогою візуального програмування, він інтегрує різні компоненти для видобутку даних і машинного навчання за допомогою модульної концепції конвейеризации даних.

9) QlikView

Рис. 1.17 Логотип Tableau Public

QlikView має безліч унікальних особливостей, таких як запатентована технологія і обробка даних в пам'яті, яка дуже швидко видає результат

кінцевим користувачам і зберігає дані в самому звіті. Асоціація даних в QlikView підтримується автоматично і може бути стиснута майже до 10% від початкового розміру. Взаємозв'язок даних візуалізується за допомогою квітів - певний колір присвоюється пов'язаним даними, а інший колір непов'язаним.

10) Javascript (Node.js)

Рис. 1.18 Логотип Node.js

Node.js - це програмна платформа для масштабованих серверних і мережевих додатків. Додатки Node.js написані на JavaScript і можуть бути запущені в середовищі виконання Node.js на Mac OS X, Windows і Linux без змін.

Додатки Node.js розроблені для максимальної пропускної спроможності та ефективності, використовуючи неблокуче введення-виведення і асинхронні події. Додатки Node.js працюють в однопоточном режимі, хоча Node.js використовує кілька потоків для файлових і мережевих подій. Node.js зазвичай використовується для додатків реального часу завдяки своїй асинхронної природі. Node.js має наступні бібліотеки для машинного навчаня: Tensorflow.js, Brain.js, stdlib.js і т.д.[1]

Нижче перераховані деякі з важливих особливостей, які роблять Node.js першим вибором архітекторів програмного забезпечення.

- Асинхронність - Всі API бібліотеки Node.js є асинхронними, тобто неблокуючими. Це означає, що сервер на базі Node.js ніколи не чекає, поки API поверне дані. Сервер переходить до наступного API після його виклику, а механізм Подій бібліотеки Node.js допомагає серверу отримати відповідь від попереднього виклику API.
- Швидкість - Будучи побудованою на движку V8 JavaScript Engine Google Chrome, бібліотека Node.js дуже швидко виконує код.
- Однопоточність, але з високою маштабованістю - Node.js використовує однопоточну модель з циклом подій. Механізм подій допомагає сервера відповідати неблокуючим чином і робить сервер дуже масштабованим на відміну від традиційних серверів, які створюють обмежену кількість потоків для обробки запитів. Node.js використовує однопоточну програму, і ця ж програма може обслуговувати набагато більшу кількість запитів, ніж традиційні сервери, такі як Apache HTTP Server.
- Відсутність буферизації - додатки Node.js ніколи не буферизують дані. Ці додатки просто виводять дані по частинах.

1.2.2 Переваги та недоліки існуючих рішень

Серед існуючих рішень розглянемо детальніше лише мови програмування, так як лише за допомогою них можна зробити гнучкий аналіз та візуалізацію даних без обмежень.

1) Python

Python - це мова програмування. Багато програмістів люблять його за простоту. Те, що він може здатися простим, це не означає, що він малофункціональний.

Переваги Python для аналізу даних:

- Універсальна багатоцільова мова програмування: можна виконувати не тільки обробку даних, а й їх пошук, а також використовувати результат обробки в веб-додатку.
- Інтерактивність мови програмування (обчислення без компіляції): програмісти також цінують Python за вбудований інтерпретатор, який дозволяє кодувати на ходу. В Data Science це актуально для перевірки гіпотез в інтерактивному режимі.
- Динамічний розвиток мови: ця мова швидко і інтенсивно розвивається. З кожною версією підвищується продуктивність мови і поліпшується синтаксис. Наприклад, у версії 3.8 з'явився новий оператор моржування -: =, що є досить серйозною подією для будь-якої мови. У низькорівневих мовах, таких як C ++ або Java, швидкість змін помітно нижче - вони затверджуються спеціальним комітетом, який збирається раз на кілька років.
- Вбудовані можливості для оптимізації вихідного коду: вбудований інтерпретатор корисний для розробників. Оскільки Python пропонує неявну і динамічну типізацію даних, оцінити ступінь оптимізації можна тільки під час виконання коду, для чого корисний інтерпретатор. Він переводить вихідний код в машинні інструкції, які можуть підказати ідею для оптимізації. Наприклад, порівнявши дві інструкції, можна зрозуміти, чому одна працює швидше за іншу. Це важлива перевага для роботи з Великими Даними, тому що крім аналізу даних є багато роботи щодо поліпшення алгоритмів їх обробки.
- В Python процес стандартизації більш відкритий для спільноти, кожен може висувати свої ідеї, і їх кількість швидко зростає.

Недоліки Python для аналізу даних:

Незважаючи на те, що Python є об'єктно-орієнтованою мовою програмування, у нього є деякі недоліки для Data Science. Можливо, вони не є вирішальними, але кожен програміст або команда Data Science перед

початком роботи над проектом Data Science вирішує їх самостійно.

- Візуалізація. Ця можливість є важливим критерієм при виборі програмного забезпечення для аналізу даних. Хоча в Python є приємні бібліотеки для візуалізації, такі як Seaborn, Bokeh і Pygal, вибір може бути занадто великий. Більш того, в порівнянні з R, візуалізація в Python набагато складніше, і її результати іноді не дуже зрозумілі.
- Відсутність загального сховища та відсутність альтернатив для багатьох бібліотек R.
- Python - це мова з динамічною типізацією. Це значно прискорює розробку програм, але також ускладнює пошук важко відстежуються помилок, викликаних неправильним присвоєнням різних даних одним і тим же змінним.
- Глобальне блокування інтерпритатора. На даний момент це основна проблема продуктивності в Python. Вона також пов'язана з поганою реалізацією багатопоточності. Код GIL не змінювався з часів першої версії мови програмування. Це явно вказує на те, що він застарілий.

2) R

Переваги R для аналізу даних:

- Мова була створена спеціально для задач аналізу даних: запис мовних конструкцій зрозуміла багатьом фахівцям. Зручні і зрозумілі мовні конструкції - безперечна перевага для непрофесійних програмістів.
- Більшість функцій для роботи з аналізом даними з коробки, тобто вони вшиті в саму мову. Наприклад, така задача, як перевірка статистичних гіпотез може зайняти лише пару рядків коду.
- Установка IDE (RStudio) і необхідних пакетів обробки даних гранично спрощена.
- R має безліч структур даних, операторів і параметрів. У ньому є багато: від масивів до матриць, від циклів до рекурсії, а також інтеграція з іншими мовами програмування, такими як C, C ++ і Fortran.

- R в основному використовується для статистичних обчислень. У ньому є набір алгоритмів, які використовуються інженерами і консультантами по машинному навчання. Він використовується при аналізі часових рядів, класифікації, кластеризації, лінійному моделюванні і т.д.
- Велика бібліотека пакетів та тесті майже для всіх функцій машинного навчання та Data Science
- Кілька якісних пакетів для візуалізації даних для різних завдань (ggplot2, lattice, ggvis, googleVis, rCharts і ін.). Можна будувати як двовимірні графіки (діаграми, боксплоти), так і тривимірні моделі.
- Основні статистичні методи реалізовані у вигляді стандартних функцій, що значно збільшує швидкість розробки.
- Для R існує величезна кількість додаткових пакетів на будь-який смак. Це може бути пакет для аналізу тексту (так зване моделювання природної мови) або пакет з даними з Twitter. З кожним днем пакетів стає все більше, і більшість з них зібрані в одному місці - в спеціальному сховищі CRAN.

Недоліки R для аналізу даних:

Як і будь-яка мова програмування, R має деякі недоліки. Кожен програміст сам вирішує, який недолік можна ігнорувати, а на який не варто звертати уваги.

- Низька продуктивність. Однак в системі є пакети, що дозволяють програмістам збільшити швидкість (pqR, renjin, FastR, Riposte і ін.). При роботі з Big Data рекомендується використовувати бібліотеки data.table і dplyr.
- Специфічність в порівнянні зі стандартними мовами програмування, так як мова вузькоспеціалізований (наприклад, індексація векторів починається з цифри один, а не з нуля).
- Оскільки більшість кодів на R пишуть люди, які не знайомі з цим програмуванням, читабельність деяких програм залишає бажати кращого. Крім того, не всі користувачі дотримуються рекомендацій по

оформленню програмного коду.

- R - відмінний інструмент для статистики і автономних додатків, але він не так добре працює в тих областях, де традиційно використовуються мови загального призначення.
- Одну і ту ж функціональність можна виконати різними способами. Синтаксис для деяких завдань не зовсім очевидний.
- Через великої кількості бібліотек документація по деяким менш популярним з них не може вважатися повною.

3) Javascript

Незважаючи на нещодавні вдосконалення мови, більшість розробників все одно радять не використовувати JavaScript для ML з однієї причини: екосистеми. Екосистема Python для МЛ настільки зріла і багата, що важко виправдати вибір будь-якої іншої екосистеми. Але ця логіка самореалізується і самознищується; Якщо ми хочемо, щоб екосистема JavaScript зріла, нам потрібні сміливі люди, щоб зробити стрибок і працювати над реальними проблемами машинного навчання. JS є найпопулярнішою мовою програмування протягом кількох років поспіль, і його популярність зростає майже за кожним показником.

- Використання JavaScript для ML має деякі переваги. В той час як ML в JavaScript наразі не дуже популярний, попит на саму мову зростає. Оскільки попит на програми ML зростає, а апаратне забезпечення стає все швидшим і дешевшим, цілком природно, що ML стає все більш поширеним у світі JavaScript. Існує безліч ресурсів для вивчення JavaScript загалом, підтримки серверів Node.js та розгортання програм JavaScript. Екосистема Node Package Manager (npm) також велика і продовжує зростати, і хоча не так вже й багато доступних дуже зрілих пакетів ML, існує ряд добре побудованих корисних інструментів, які незабаром стануть дозрілими.
- Універсальність мови являється найбільшою перевагою цієї мови

програмування. Сучасний веб-браузер - це, по суті, портативна платформа додатків, яка дозволяє запускати ваш код, практично без змін, практично на будь-якому пристрої. Такі інструменти, як електрон (хоча багато хто вважає їх роздутими), дозволяють розробникам швидко розробляти та розгортати завантажувані настільні програми у будь -якій операційній системі. Node.js дозволяє запускати ваш код у середовищі сервера. React Native переносить ваш код JavaScript у рідне середовище мобільних додатків, а згодом може дозволити вам розробляти настільні програми. JavaScript більше не обмежується лише динамічними веб-взаємодіями, тепер він є універсальною, багатоплатформенною мовою програмування.

- Нарешті, використання JavaScript робить ML доступним для веб та інтернет -розробників - групи, яка історично залишалася поза обговоренням ML. Додатки на стороні сервера, як правило, є кращими для інструментів ML, оскільки сервери знаходяться там, де є обчислювальні можливості. Цей факт історично ускладнював веб-розробникам доступ до гри ML, але в міру вдосконалення обладнання навіть складні моделі ML можна запускати на клієнті, будь то настільний комп'ютер або мобільний браузер.
- Якщо веб -розробники, інтернет-розробники та розробники JavaScript сьогодні почнуть дізнаватися про ML, ця спільнота зможе вдосконалити інструменти ML, доступні нам усім завтра. Якщо ми візьмемо ці технології та демократизуємо їх, відкриємо якомога більшу кількість людей концепціям, що лежать в основі ML, ми в кінцевому підсумку піднімемо спільноту та посіємо наступне покоління дослідників ML.

1.3 Методи представлення статистичних даних

1.3.1 Методи аналізу часових рядів

Часові ряди дозволяють прогнозувати значення. На підставі попередніх значень часових рядів можна спрогнозувати тенденції в економіці та погоді або спланувати пропускну здатність. З огляду на особливих властивостей даних часових рядів для роботи з ними застосовуються спеціалізовані статистичні методи і підходи.

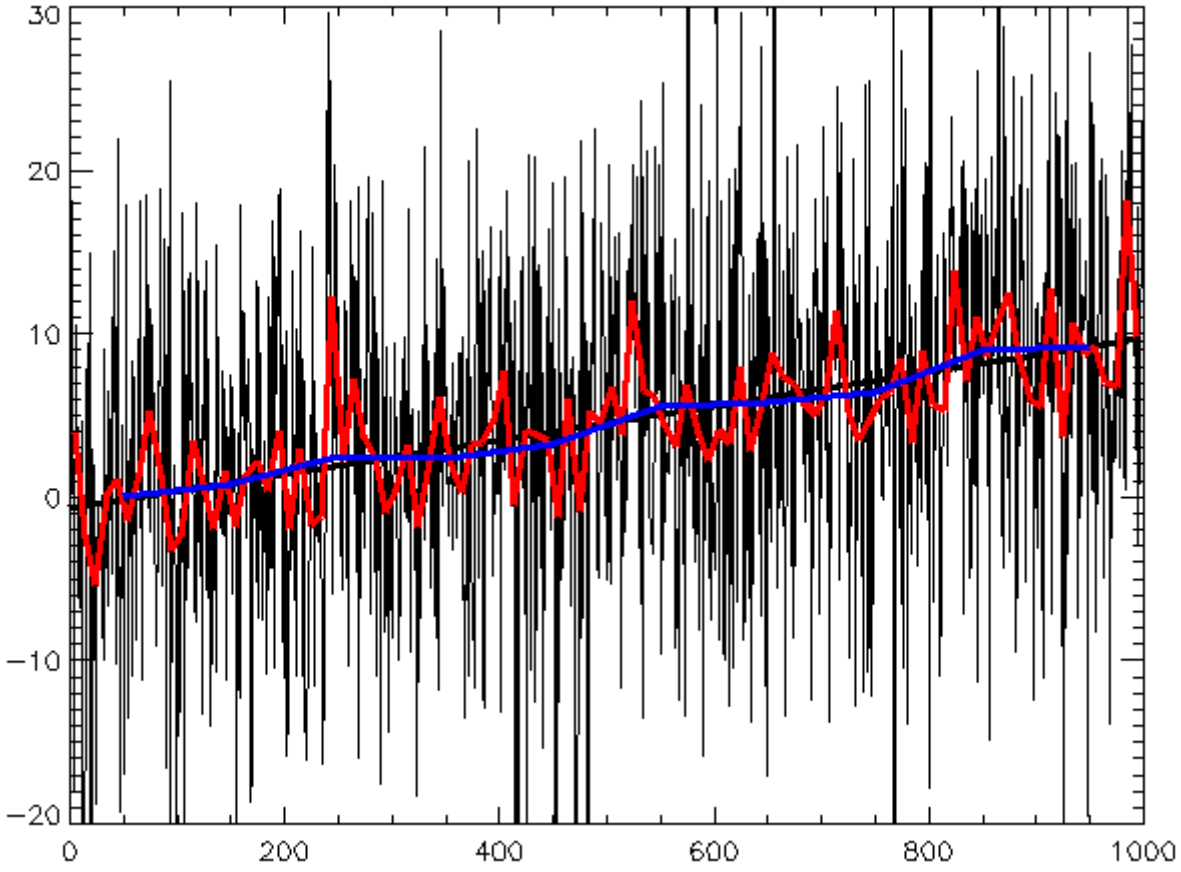

Рис. 1.19 Приклад часового ряду

Чотири компонента часового ряду

Першим кроком при аналізі часового ряду для розробки прогностичної моделі є виявлення і розуміння закономірностей, що лежать в основі даних з плином часу. Ці основні закономірності зазвичай класифікуються як такі чотири компоненти:

- Тенденція - довгострокова поступова зміна ряду. Це найпростіша

модель тренда, так як вона демонструє довгострокове зростання або спад.

- Seasonality (Сезонність) - Передбачувані, короткострокові моделі, які виникають протягом однієї одиниці часу і повторюються нескінченно.
- Циклічна складова - Довгострокові коливання даних, на які можуть піти роки або десятиліття. Такі коливання відбуваються не передбачувано і часто є результатом зовнішніх економічних умов.
- Noise (Помилка) - Випадкові коливання внаслідок неконтрольованих обставин.

Найбільш точна модель машинного навчання для прогнозування часових рядів є найпростішою. Точно так же, як вчені часто починають моделювати дані поперечного перерізу з простої лінійної регресією, існують еквіваленти часових рядів. Ось кілька прикладів:

Наївний прогноз - в наївному прогнозі прогнозоване значення просто рівнозначно значенням самого останнього спостереження. Цей найпростіший метод часто використовується як еталон для оцінки роботи складніших прогнозів. Іншими словами, якщо Ваша складна модель менш точна, ніж наївний прогноз, то Ви, швидше за все, робите щось не так.

Середнє - У середньостатистичному методі всі прогнози рівні середнього значення всіх історичних даних.

Сезонний наївний метод - це схоже на наївний прогноз за винятком того, що прогнозоване значення є останнім спостережуваним значенням з того ж сезону тимчасового періоду. Наприклад, в місячному масштабі при використанні цього методу липневий прогноз буде дорівнює останньому спостерігається значенням в липні.

Регресійне прогнозування часових рядів

Можна розробляти лінійні, поліноміальні і експоненціальні регресивні моделі прогнозування часових рядів, створюючи змінну індексу часу від

першого спостереження (t = 1) до останнього (t = n). В результаті виходить модель тренда, а не сезонності. Це корисний метод, якщо в основі лежить припущення про те, що даний тренд є підходящим і актуальним для обраного вами тимчасового періоду.

Якщо модель дійсно потребує врахування сезонності, це також можна зробити за допомогою лінійної регресії. Це робиться шляхом створення категоричній змінної, що вказує сезони.

Згладжуючі методи

На відміну від регресійних моделей, заснованих на припущеннях про структуру тренда або шуму, методи згладжування часових рядів призначені для адаптації до змін даних в часі. Згладжування зменшує шум за рахунок усереднення спостережень за багаторазовими періодами. Нижче ми обговоримо два найбільш поширених методу згладжування - ковзне середнє і експоненціальне згладжування:

Ковзне середнє - Метод змінного середнього створює ряд середніх шляхом взяття середніх значень в часі ряду в межах заданих періодів. Передбачається, що спостереження, близькі за часом, ймовірно, також схожі за значенням, тому взяття середнього усуває шум. Ковзаючі середні зазвичай беруться з найостанніших точок даних.

Експоненціальне згладжування - експоненціальне згладжування бере середнє зважене значення за всіма минулим значенням, надаючи більшої ваги з останніми спостереженнями. Метою є визнання старої інформації, при цьому пріоритет віддається найостаннішими даними.

Спеціальні вимоги до даних часових рядів

Як ви можливо знаєте, розробка моделі вимагає поділу даних на навчальні та валідаційні набори. При використанні даних поперечного перерізу ви випадковим чином розділіть дані на ці групи. Однак ви не можете випадковим чином розділити дані, які мають послідовний тимчасової

елемент. При моделюванні часових рядів в якості навчального набору використовуються більш ранні дані, а в якості валідаційні набору - більш пізні дані.

Крім того, при прогнозуванні нових значень сама модель будується на основі всього масиву даних, а не тільки навчального розділу.

1.3.2 Методи візуалізації часових рядів

Візуалізація грає дуже важливу роль в аналізі часових рядів і прогнозуванні.

За допомогою графіка вихідних даних можна виявити тимчасові структури, такі як: тренди, цикли і сезонності, які можуть впливати на вибір моделі. Проблема полягає в тому, що не завжди використовують повний спектр інструментів візуалізації, зупиняючись тільки на лінійному графіку.

Методи візуалізації часових рядів на Python:

- Pandas. Бібліотека Pandas – це швидкий, потужний, гнучкий і простий у використанні інструмент аналізу та маніпулювання даними з відкритим вихідним кодом,побудований на основі мови програмування Python.
- Matplotlib. Matplotlib - бібліотека на мові програмування Python для візуалізації даних двовимірної графікою. Одержувані зображення можуть бути використані в якості ілюстрацій в публікаціях. Незважаючи на широту, частина власної документації matplotlib серйозно застаріла. Бібліотека все ще розвивається, і безліч старих прикладів в мережі можуть включати на 70% менше коду, ніж в їх сучасній версії;

Методи візуалізації часових рядів на Javascript:

- Разом з бібліотекою для аналізу даних Tensorflow.js іде інструмент для візуалізації tfvis;
- D3.js - потужна JavaScript бібліотека для візуалізації даних, з відкритим вихідним кодом. По суті своїй, D3 більше схожий на фреймворк, ніж на

бібліотеку.

- Plotly.js - це бібліотека JavaScript високого рівня, безкоштовна, з відкритим вихідним кодом. Побудована на D3.js і WebGL.
- NgxChart – бібліотека для фреймворку Angular

Методи візуалізації часових рядів на R:

- Бібліотека ggplot2. У ggplot ви збираєте графіки «по цеглинці», окремо визначаючи джерело даних, способи зображення, параметри системи координат і т.д. - шляхом виклику і додавання результатів відповідних функцій. При побудові елементарних графіків ggplot може здатися складніше, ніж стандартна графічна підсистема. Однак при ускладненні вимог до зовнішнього вигляду і інформаційного насичення графіка складність ggplot виявляється перевагою, і з її допомогою відносно просто можна отримувати елегантні і інформативні візуалізації

1.4 Висновки

Python завжди був і залишається найкращим мовою для машинного навчання, завдяки зрілості мови, зрілості екосистеми та позитивному зворотньому зв'язку з ранніми ML-зусиллями на Python. Однак останні події в світі JavaScript роблять JavaScript більш привабливим для проектів ML. Вважається, що протягом декількох років ми побачимо серйозне відродження ML на JavaScript, особливо в міру того, як ноутбуки і мобільні пристрої будуть ставати все більш потужними, а сам JavaScript буде набирати популярність.

Javascript має обширну бібліотеку під назвою Tensorflow.js, яку можна використовувати для визначення, навчання і запуску моделей машинного навчання цілком на севері або у браузері з використанням Javascript і API-інтерфейсу шарів високого рівня.

Дані про захворюваність на COVID-19 будуть отримуватися з сервісу

ourworldindata, тому що він має найбільшу кількість параметрів, які можна використовувати для аналізу даних. Дані оновлюються щодня.

РОЗДІЛ 2
МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Автоматизований збір даних, формування бази даних

Було вирішено проводити автоматизований збір даних кожен день та зберігати ці дані до хмарового сервісу Atlas MongoDB.

MongoDB Atlas — це хмарна служба баз даних NoSQL, розроблена компанією MongoDB Inc. Вона була розроблена, щоб запропонувати гнучку, масштабовану платформу на вимогу, щоб усунути необхідність у дорогоцінній інфраструктурі, конфігураціях та технічному обслуговуванні.

MongoDB Atlas надає всі функції MongoDB — без необхідності турбуватися про такі завдання адміністрування бази даних, як:

- Забезпечення інфраструктури
- Конфігурації бази данихц
- Патчі
- Масштабування
- Резервні копії

MongoDB Atlas забезпечує простий спосіб розміщення ваших даних і керування ними в хмарі.

Переваги MongoDB Atlas

- Глобальні кластери для додатків світового класу: Використовуючи MongoDB Atlas, ми можемо вільно вибирати хмарного партнера та екосистему, які відповідають нашій бізнес-стратегії.
- Безпечний для конфіденційних даних: він пропонує вбудовані засоби контролю безпеки для всіх наших даних. Це дозволяє функціям корпоративного рівня інтегруватися з нашими існуючими протоколами безпеки та стандартом відповідності.
- Розроблено для підвищення продуктивності розробників: MongoDB Atlas працює швидше за допомогою загальних інструментів для роботи

з нашими даними та платформи послуг, яка дозволяє легко створювати, захищати та розширювати програми, які працюють на MongoDB.

- Надійний для критично важливих робочих навантажень: він побудований на основі розподіленої відмовостійкості та автоматизованого відновлення даних.
- Створений для оптимальної продуктивності: це дозволяє легко масштабувати наші бази даних у будь-якому напрямку. Ми можемо отримати більше від наших існуючих ресурсів за допомогою інструментів оптимізації продуктивності та перегляду показників бази даних у режимі реального часу.
- Керований для операційної ефективності: він поставляється з вбудованими передовими методами роботи, тому ми можемо зосередитися на забезпеченні цінності для бізнесу та прискоренні розробки додатків замість керування базами даних.

Створюємо базу даних з назвою covid та додаємо наступні колекції:

1) last-cases – Дані по Covid-19 за останній день
2) ukraine-cases – Дані по Covid-19 по Україні за весь час
3) vaccinations – Дані щодо вакцинування проти Covid-19

Кожен об’єкт у кожній колекції матиме наступний вигляд:

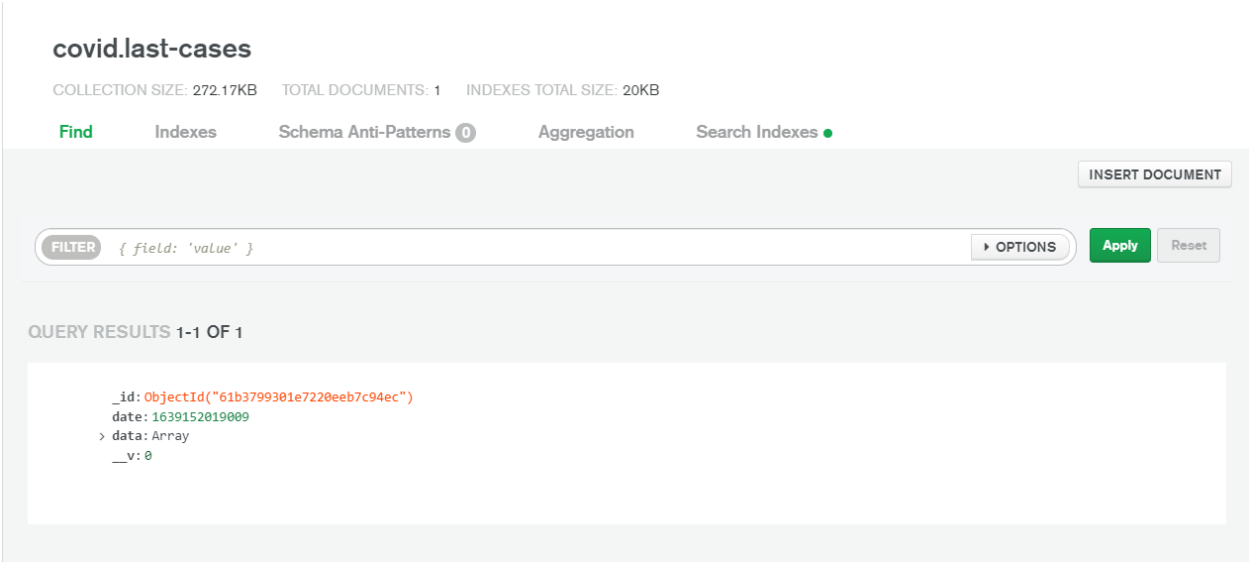

Рис. 2.1 Колекція Останні випадки у Atlas MongoDB

1) _id – Унікальний ідентифікатор
2) date – Дата оновлення даних
3) data – Самі дані в залежності від колекції

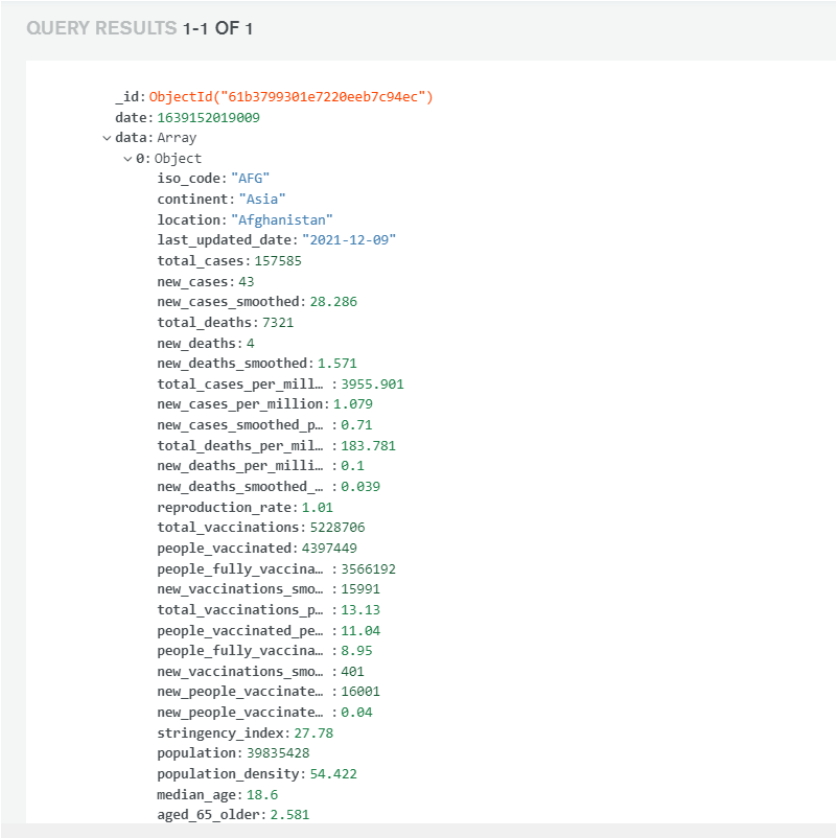

Рис 2.2 Приклад збережених даних колекції last-cases

Автоматизований збір даних виконується за допомогою бібліотеки node-schedule.

Node Schedule — це гнучкий планувальник завдань, схожий на cron і не схожий на cron для Node.js. Він дозволяє планувати завдання (довільні функції) для виконання на певні дати з додатковими правилами повторення. Він використовує лише один таймер у будь-який момент часу (замість того, щоб переоцінювати майбутні завдання щосекунди/хвилини).

Кожне заплановане завдання в Node Schedule представлено об’єктом Job. Ви можете створювати завдання вручну, а потім виконати метод schedule(), щоб застосувати розклад, або використовувати зручну функцію

scheduleJob().

```
const schedule = require('node-schedule');

const job = schedule.scheduleJob('42 * * * *', function(){
  console.log('The answer to life, the universe, and everything!');
});
```

Рис 2.3 Приклад використання бібліотеки

```
export const scheduleDbUpdate = () => {
    const scheduler = schedule.scheduleJob(config.SCHEDULER_FORMAT || '* * * * 00, 12 1-7', () => {

        saveData(
            [
                bind(saveModel, null, lastCasesCSV, LastCases, true),
                bind(saveModel, null, totalCasesCSV, UkraineCases, true, 'Ukraine'),
                bind(saveModel, null, vaccinationsCSV, VaccinationCases, true)
            ]
        );

    });

    return scheduler;
}
```

Рис 2.4 Практичне використання

'* * * * 00, 12 1-7' – Означає, що скрипт (зберігання даних до бази даних) буде виконуватися кожен день о 00-00 та 12-00.

2.2 Попередня обробка даних

Більшість графіків-мапи вимагають формат ISO2 для корректного відображення інформації на мапі. Нажаль сервіс Our world in Data надає лише ISO код. В вирішенні даної проблеми приходить API Postman, за допомогою запита якого ми можемо додати до кожного запису країни ISO2.

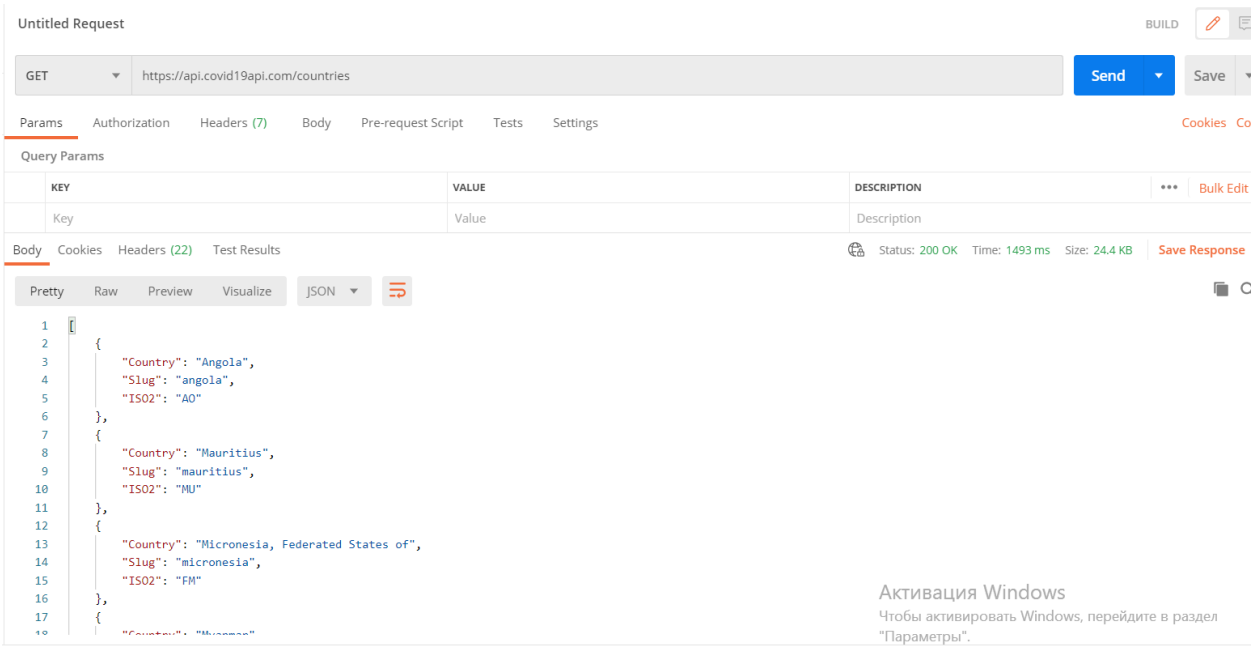

Рис 2.5 Приклад виконання запиту

Визначаємо функцію під назвою addIso2Code:

```
const addIso2Code = async(data: any[]) => {
    const countries = await (await fetch('https://api.covid19api.com/countries')).json();
    const updatedData = [];
    data.forEach(d => {
        countries.forEach(c => {
            if(
                c.Country == d.location ||
                c.Country.toLowerCase().replace(/\s/g, '').includes(d.location.toLowerCase().replace(/\s/g, '')) ||
                c.Country.toLowerCase().replace(/\s/g, '') === d.location.toLowerCase().replace(/\s/g, '')
            ) {
                updatedData.push({ ...d, iso2: c.ISO2 });
            }
        })
    });

    return updatedData;
}
```

Функція робить запит до API Postman та конвертує дані до зручного формату даних для Javascript – Об’єкту. Далі ми ітеруємо наш масив з країнами та додаємо властивість ISO2.

Визначаємо функцію dropNa, яка відповідальна за заміну пропусків або undefined-значень на нулі:

```
const dropNa = (dataset) => {
    return dataset.map(data => {
        for(let k of Object.keys(data)) {
```

```
            if(!Boolean(data[k])) data[k] = 0;
        }

        return data;
    })
}
```

2.3 Метод прогнозування

Для цього проекту було вирішено використовувати нечітку логіку для прогнозування зараження Covid-19.

Нечітка логіка вперше була запропонована Лотфі Заде в статті 1965 року для журналу Information and Control. У своїй статті під назвою «Нечіткі множини» Заде спробував відобразити тип даних, що використовуються при обробці інформації, і вивів елементарні логічні правила для такого роду множин.

«Часто класи об’єктів, які зустрічаються в реальному фізичному світі, не мають точно визначених критеріїв належності», – пояснив Заде. «Втім, факт залишається фактом, що такі неточно визначені «класи» відіграють важливу роль у людському мисленні, зокрема в сферах розпізнавання образів, передачі інформації та абстракції»

З тих пір нечітка логіка успішно застосовується в системах керування машинами, обробці зображень, штучному інтелекті та інших областях, які покладаються на сигнали з неоднозначною інтерпретацією.

Термін нечіткий стосується речей, які є неясними або нечіткими. У реальному світі багато разів ми стикаємося з ситуацією, коли не можемо визначити, чи є цей стан істинним чи хибним, їхня нечітка логіка забезпечує дуже цінну гнучкість для міркування. Таким чином ми можемо розглянути неточності та невизначеності будь-якої ситуації.[22]

У логічному значенні істинної системи 1,0 представляє абсолютне

значення істини, а 0,0 — абсолютне хибне значення. Але в нечіткій системі немає логіки для абсолютної істини та абсолютної хибної цінності. Але в нечіткій логіці також є проміжне значення, яке частково істинне і частково хибне.

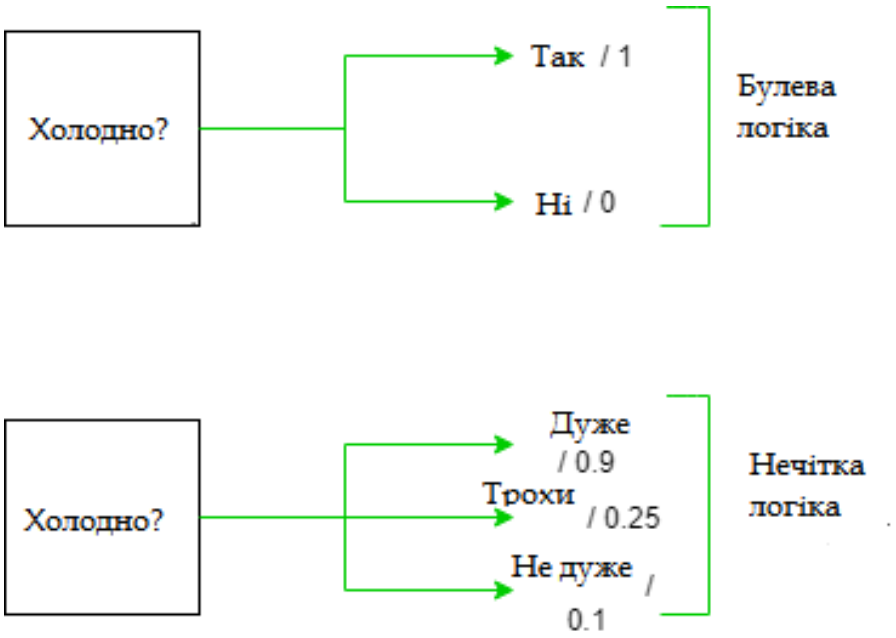

Рис. 2.6 Порівняння булевої та нечіткої логіки

Його архітектура складається з чотирьох частин:

1. База правил: містить набір правил і умов IF-THEN, наданих експертами для керування системою прийняття рішень на основі лінгвістичної інформації. Останні розробки нечіткої теорії пропонують декілька ефективних методів для проектування та налаштування нечітких регуляторів. Більшість із цих розробок зменшують кількість нечітких правил.
2. Фазифікація: Використовується для перетворення вхідних даних, тобто чітких чисел, у нечіткі набори. Чіткі вхідні дані — це в основному точні вхідні дані, виміряні датчиками та передані в систему керування для обробки, такі як температура, тиск, оберти в хвилину тощо.
3. Інференційний модуль: він визначає ступінь відповідності поточного нечіткого введення щодо кожного правила та вирішує, які правила

мають запускатися відповідно до поля введення. Далі запущені правила об’єднуються для формування керуючих дій.

4. Дефазифікація: Використовується для перетворення нечітких наборів, отриманих за допомогою механізму виведення, у чітке значення. Існує кілька доступних методів дефаззфікації, і найкращий з них використовується з конкретною експертною системою, щоб зменшити помилку.

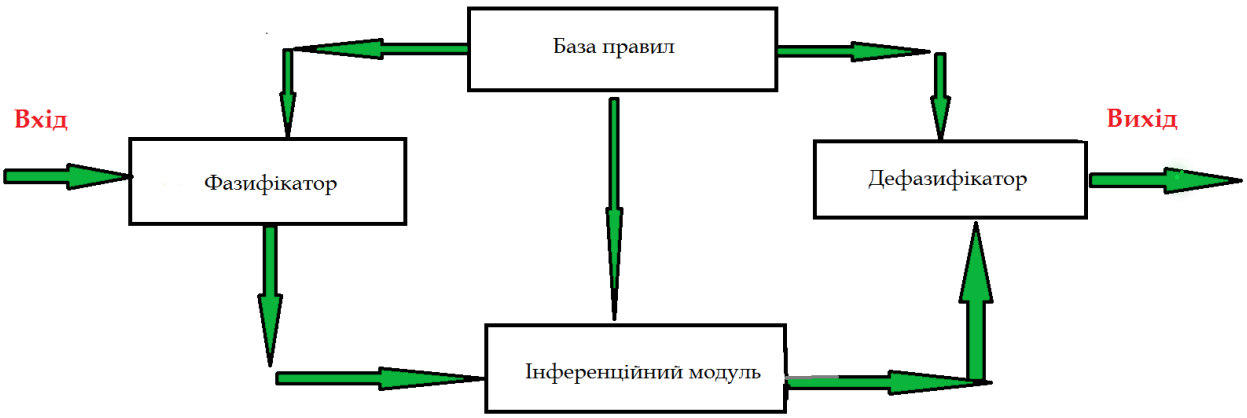

Рис 2.7 Архітектура нечіткої логіки

Нечітка логіка в її найпростішому розумінні розвивається за допомогою аналізу типу дерева рішень. Таким чином, у ширшому масштабі він формує основу для систем штучного інтелекту, запрограмованих за допомогою висновків на основі правил.[23, 24]

Так вийшло, що термін нечіткий до величезної кількості сценаріїв, які можуть бути розроблені в деревоподібній системі рішень. Розробка протоколів нечіткої логіки може вимагати інтеграції програмування на основі правил. Ці правила програмування можна назвати нечіткими наборами,

оскільки вони розробляються на розсуд комплексних моделей.

Нечіткі набори також можуть бути більш складними.

Як правило, більш складні аналоги програмування можуть мати можливість доповнювати правила, які використовуються для визначення змінних. Це приводить до ширшого діапазону варіантів, але з менш точним прогнозуванням, заснованим на цих правилах.

Нечітка семантика в штучному інтелекті

Центральним компонентом програмування рішень штучного інтелекту являється концепція нечіткої семантики та нечіткої логіки. Інструменти та рішення ШІ продовжують розширюватися в економіці в ряді секторів, оскільки також розширюються можливості програмування з нечіткої логіки.

IBM Watson є однією з найвідоміших систем штучного інтелекту, що використовує варіації нечіткої логіки та нечіткої семантики. Зокрема, у фінансових послугах нечітка логіка використовується в машинному навчанні та технологічних системах, які підтримують результати інвестиційної розвідки.

У деяких просунутих моделях торгівлі інтеграція математики нечіткої логіки також може використовуватися, щоб допомогти аналітикам створювати автоматизовані сигнали купівлі та продажу. Ці системи допомагають інвесторам реагувати на широкий спектр змінних ринкових змінних, які впливають на їхні інвестиції.

Приклади нечіткої логіки

Найновіші моделі торгівля програмним забезпеченням вже використовують нечітку логіку для аналізу десятки тисяч цінних паперів у режимі реального часу та надають інвесторам можливість робити висновки на основі цих прогнозів. Нечітка логіка часто використовується, коли інвестор прагне використовувати кілька факторів для розгляду. Це може призвести до звуженого аналізу торгових рішень.[3] Трейдери також можуть

мати можливість програмувати різноманітні правила для проведення торгів. Два приклади включають наступне:

Правило 1: якщо ковзне середнє низьке, а індекс відносної сили (RSI) низький, тоді продайте.

Правило 2: якщо ковзне середнє висока, а індекс відносної сили (RSI) високий, тоді купуйте.

Нечітка логіка дозволяє трейдеру програмувати власні суб'єктивні висновки на низькі та високі значення в цих основних прикладах, щоб отримати власні автоматизовані торгові сигнали.

Переваги та недоліки нечіткої логіки

Нечітка логіка часто використовується в контролерах машин і штучному інтелекті, а також може бути застосована до програмного забезпечення для торгівлі. Хоча він має широкий спектр застосування, він також має істотні обмеження.

Оскільки нечітка логіка імітує прийняття рішень людиною, вона найбільш корисна для моделювання складних проблем з неоднозначними або спотвореними вхідними параметрами. Через схожість з природною мовою алгоритми нечіткої логіки легше кодувати, ніж стандартне логічне програмування, і вимагають менше інструкцій, що дозволяє заощадити вимоги до пам'яті.

Ці переваги також мають недоліки через неточну природу нечіткої логіки. Оскільки системи розроблені для неточних даних і введених даних, вони повинні бути перевірені та підтверджені, щоб уникнути неточних результатів.

Різниця між нечіткою логікою та нейронними мережами

Штучна нейронна мережа — це обчислювальна система, призначена для імітації процедур вирішення проблем людської нервової системи. Це відрізняється від нечіткої логіки, набору правил, призначених для досягнення

висновків з неточних даних. Обидва мають застосування в інформатиці, але це різні галузі.

2.4 Кластеризація даних

У цій роботі проведено розподільне кластеризацію даних COVID-19 за допомогою алгоритмів c-Means (cM) та нечітких c-Means (Fc-M). На основі даних, доступних з січня 2020 року щодо місця розташування, тобто довготи та широти земної кулі, підтверджені щоденні випадки, одужання та смерті групуються. Під час аналізу максимальний розмір кластера розглядається як змінна і варіюється від 5 до 50 в обох алгоритмах, щоб знайти оптимальне число.

c-M clustering

У цьому алгоритмі центр кластера отримується шляхом мінімізації функції вартості з мірою несхожості. Для заданих точок даних набір m представлений у вигляді X = {xi | i = 1…m}. Цей набір являє собою вектор з n-вимірністю, в якому алгоритм c-M розбиває дані на c кластерів, щоб мінімізувати несхожість J (X, V). Неподібність вибирається евклідовою відстанню, яка є сумою квадратів. Цільова функція математично показано, як у формулі. Таким чином, індекс продуктивності для c-M визначається як:

$$J(X,V) = \sum_{j=1}^{k} J_i(X_i, V_j) = \sum_{j=1}^{k} \left(\sum_{i=1}^{k} u_{ij} d^2(x_i, v_j) \right)$$

(2.4.1)

Де:

$$J_i\left(x_i, v_j\right) = \sum_{i=1}^{m} u_{ij} d^2\left(x_i, v_j\right)$$

(2.4.2)

У кластері C_i є цільовою функцією. Коли xi ∈ C_j, членство $u_{ij} = 1$, інакше воно дорівнює 0. Відстань $d^2(x_i,v_j)$ між v_j та x_i визначається як:

$$d^2\left(x_i, v_j\right) = \| \sum_{k=1}^{n} {x_k}^i - {v_k}^j \|^2$$

(2.4.3)

Наведене вище рівняння є евклідовою відстанню між центром кластера і точкою даних.

kth розмірне значення x_i задається як x^i_k. kth розмірне значення v_j визначається як v^j_k

Функція належності V = {v_1, v_2……. v_k} для центрів кластерів визначаються як:

$$u_{ij} = \begin{cases} 1; & if\ d^2\left(x_i, v_j\right) \leq d^2\left(x_i, v_{j*}\right) j \neq j^*,\ \forall j^* = 1, \cdots, k \\ 0; & otherwise \end{cases}\Bigg)$$

(2.4.4)

Відповідно до рівняння 1 слід призначити точку x_i, що належить кластеру c_j, що має найближчий центр кластера v_j. Тоді оптимальний центр кластера v_j, який зменшує середнє значення точок даних кластера в j кластері, призначається з U = [u_{ij}] за допомогою рівняння. (4). В Алгоритмі 1 згадується покрокова процедура розподілу точок даних COVID-19 за

допомогою c-M. На наступному рисунку показано, що алгоритм c-Means утворює кластери до та після застосування.

(2.4.5)

$$v_j = \frac{1}{|c_j|} \sum_{i, x_i \in c_j}^{n} x_i$$

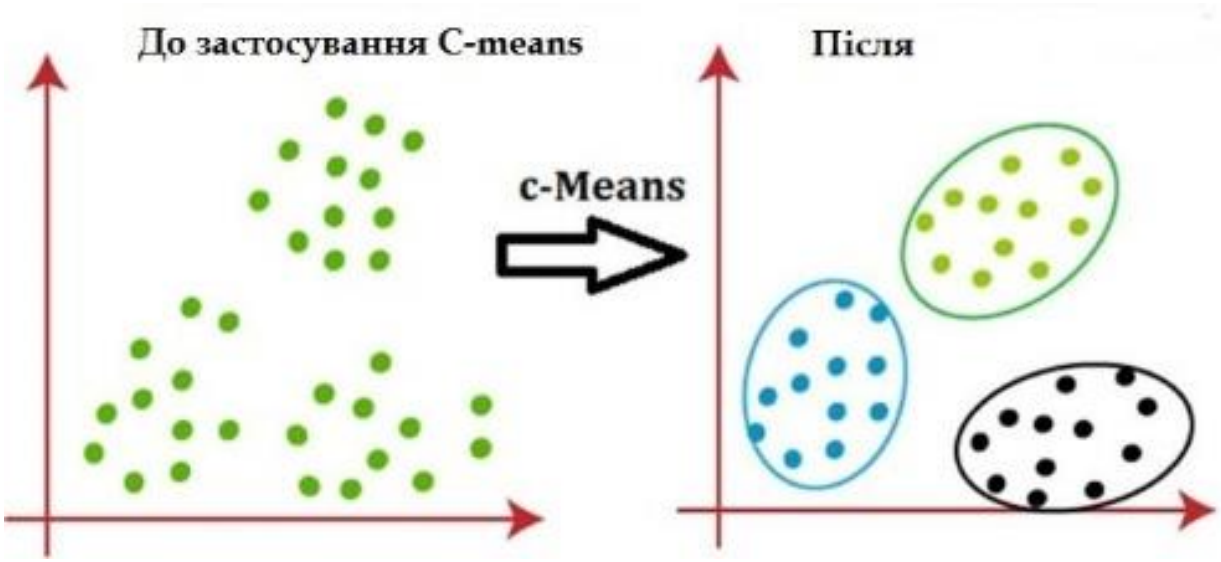

Рис. 2.8 C-means на розподілених даних

Fc-M кластеризація

Fc-M кластеризація широко використовується в техніці аналізу даних. Кожна точка даних у цій техніці належить до кластера за ступенем членства. Як і кластеризація c-M, кластеризація Fc-M залежить від функції вартості іншого заходу, який мінімізується. Fc-M інтегрує просту концепцію c-M, тоді як кожна точка даних у Fc-M певною мірою належить до кластеру. Хоча кожна точка даних c-M або належить, або не належить до певного кластеру. Таким чином, Fc-M виконує нечіткі кластери, так що певна точка даних може належати до кількох ступенів приналежності кластерів від 0 до 1. Однак, намагаючись розділити дані, Fc-M все ще використовує цільову функцію, яку потрібно звести до мінімуму. Алгоритм використовує параметр q індексу

нечіткості, який визначає кількість нечіткості кластерів. Алгоритм працює як алгоритм чіткого розподілу, коли значення q досягає 1. Збільшення значення q сприяє подальшому перекриванню кластерів.

Знову точки даних X = {x_i | i = 1…m} де кожен x_i = {x^i1, x^i1…… x^i_n, | i = 1…m}. n-вимірний вектор центрів кластерів V = {x_j | j = 1…c} і кожен v_j = {v_j1, v_j1…… v_jn, | j = 1…c}. u_{ij} — це належність точок даних з:

$$u_{ij} \in [1,0] \forall i = 1 \cdots . m \; and \; \forall j = 1 \cdots . c$$

(2.4.6)

U = [u_{ij}] — це матриця Fc-M, яка представляє розподіл точок даних і задовольняє умові:

$$\sum_{j=1}^{c} u_{ij} = 1, \quad \forall i = 1 \cdots . m$$

$$0 < \sum_{j=1}^{c} u_{ij} < m, \quad \forall j = 1 \cdots . c$$

(2.4.7)

Тут q ∈ [1, ∞] — індекс нечіткості кластеризації, а відстань d2ij(xi−,vj−) знаходиться між v_j− і x_i−Оновлення центру кластерів відбувається за допомогою рівняння.

$$\bar{v_j} = \frac{\sum_{i=1}^{m} {u_{ij}}^q \bar{x_i}}{\sum_{i=1}^{m} {u_{ij}}^q}$$

(2.4.8)

Приналежність точок кластера розраховується за допомогою рівняння. Весь алгоритм Fc-M в кластеризації COVID-19 згадується в Алгоритмі 2.

$$u_{ij} = \frac{\left(\frac{1}{{d_{ij}}^2(\bar{x_i},\bar{v_j})}\right)^{1/(q-1)}}{\sum_{i=1}^{k}\left(\frac{1}{{d_{ij}}^2(\bar{x_i},\bar{v_j})}\right)^{1/(q-1)}}$$

(2.4.9)

Індекси дії

Для набору даних про COVID-19 X оптимальна кількість кластерів розраховується за допомогою функції валідності S, що представляє відношення компактності до відокремленості.

$$S = \frac{\sum_{k=1}^{n}\sum_{k=1}^{n} {x_k}^i - {v_k}^j}{\mathrm{m.}\min\limits_{l\neq k} \| \sum_{k=1}^{n} \bar{v_l} - \bar{v_k} \|^2}$$

(2.4.10)

Зменшення вихідного індексу зазвичай досягається шляхом зміни оцінки членства точок даних і центрів кластерів по-різному, перш ніж буде досягнута конвергенція. Цей індекс базується на сумі квадрата критерію

помилки. Центри кластерів оновлюватимуться наступним чином під час кожної ітерації.

$$\bar{v_j} = \frac{\sum_{i=1}^{m} u_{ij}{}^{q} \bar{x_i}}{\sum_{i=1}^{m} u_{ij}{}^{q}}$$

(2.4.11)

Коефіцієнт внутрішньокластерного ущільнення кластера до міжкластерного поділу є індексом валідності Xie-Beni (XB). Цей індекс базується на цільовій функції J та квадраті мінімальних відстаней центрів кластерів. Це дається як:

$$XB = \frac{\sum_{j=1}^{c} \sum_{i=1}^{m} \mu_{ij}{}^{q} \mathscr{E}^2 \left(s_i, v_j\right)}{m.\, \delta_{min}^2}$$

(2.4.12)

Тут термін δ^2_{min} — евклідова відстань між центрами скупчень, яка визначається формулою:

$$\delta_{min}^2 = min \mathscr{E}^2 \left(s_i, v_j\right)$$

(2.4.13)

Вектор середніх центрів кластерів v визначається як:

$$v = \sum_{j=1}^{c} \frac{v_j}{c}$$

(2.4.14)

Враховуючи геометричні властивості функцій належності та структуру даних, термін ζ_1 вимірює відношення нечіткого поділу до нечіткої компактності. Велике значення ζ_1 означає добре розділені та компактні кластери. Термін ζ_2 також вимірює відношення нечіткого поділу до нечіткої компактності, але враховує лише плинне значення членства. Для того, щоб отримати компактність і нечіткое поділ, термін ζ_2 використовує нечіткі об'єднання та нечіткий перетин. Для добре розділених нечітких c-розділів потрібне мале значення в чисельнику ζ_2, а високе значення в нечіткому знаменнику є нечітким c-розділом. Отже, індекс Zahid S_C має вигляд:

$$SC = \zeta_1 - \zeta_2$$

$$\zeta_1 = \frac{\sum_{j=1}^{c} \mathscr{E}^2 (v_j, v)}{\sum_{j=1}^{c} \sum_{i=1}^{m} {\mu_{ij}}^q \mathscr{E}^2 (s_i, v_j) / \sum_{i=1}^{m} \mu_{ij}}$$

$$\zeta_2 = \frac{\sum_{j=1}^{c-1} \sum_{k=j+1}^{c-1} \left(\sum_{i=1}^{m} min\left(\mu_{ij}, \mu_{ik}\right)\right)^2 / \sum_{i=1}^{m} min\left(\mu_{ij}, \mu_{ik}\right)}{\sum_{i=1}^{m} \left(max_{1 \le j \le c} \mu_{ij}\right)^2 / \sum_{i=1}^{m} max_{1 \le j \le c} \mu_{ij}}$$

(2.4.15)

Для оцінки кластерів, що перекриваються, у нечіткому розділенні індекс ПК зменшується зі збільшенням розміру кластера. CE є ще одним

параметром для обчислення нечіткості розділу кластера.

В індексі силуету (S_i) метод обчислює ширину силуету, середню ширину силуету для кожного кластера, а також загальний набір даних для середньої ширини силуету. екв. використовується для вимірювання ширини силуету і-ої точки даних, де аі відмінність відрізняється від будь-якої іншої точки даних у цьому кластері. b_i — мінімум середньої відмінності точки даних і щодо кожної іншої точки в інших кластерах.

Коефіцієнт продуктивності (PC) та ентропія (CE) відповідно наведені як:

$$PC = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{n}{\mu_{ij}}^2$$

$$CE = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{n}\mu_{ij}\log\left(\mu_{ij}\right)$$

(2.4.16)

Для оцінки кластерів, що перекриваються, у нечіткому розділенні індекс ПК зменшується зі збільшенням розміру кластера. CE є ще одним параметром для обчислення нечіткості розділу кластера.

В індексі силуету (S_i) метод обчислює ширину силуету, середню ширину силуету для кожного кластера, а також загальний набір даних для середньої ширини силуету. екв. використовується для вимірювання ширини силуету і-ої точки даних, де аі відмінність відрізняється від будь-якої іншої точки даних у цьому кластері. b_i — мінімум середньої відмінності точки даних і щодо кожної іншої точки в інших кластерах.

$$S_i = \frac{b_i - a_i}{\max(a_i - b_i)}$$

(2.4.17)

Результати

Кластеризація даних про COVID-19 із січня 2020 року за допомогою алгоритмів c-M і Fc-M містить понад 67 000 записів із атрибутами довготи, широти, дати, зареєстрованих випадків, видужав та смертей. Кластери варіюються від 5 до 50 в окремих прогонах обох алгоритмів. Мінімум кластерів зберігається від 5 до максимуму від 5 до 50. Оптимальна кількість кластерів визначається на основі обчислень алгоритмів, як зазначено в попередньому розділі. Кластери COVID-19 засновані на 1) довготі 2) широті 3) даті 4) зареєстрованих випадках, 5) одужанні та 6) смертях. Пізніше індекси валідності для розуміння якості всіх кластерів COVID-19 аналізуються на основі індексів Zahid SC, Xie-Beni Index, Fukuyama-Sugeno, Validity S function, PC та CE.

Центри кластерів відносно довготи земної кулі, де зосереджені випадки COVID-19, показані на рис. 2 під час різних верхніх меж кластерних вводів, тобто від 5 до 50. Центроїди довгот COVID-19 в основному знаходяться в зона від -10° до 20°. Три з центроїдів розташовані під кутом приблизно -30°, а ще три - під кутом 50°. Усі ці кластери отримані за допомогою алгоритмів c-M та Fc-M.

На рис. 3 показано широтне положення центроїдів COVID-19, отримане за допомогою алгоритмів c-M та Fc-M. В основному концентрації знаходяться в діапазоні від -50° до -100°, що вказує, що уражені зони є американським континентом. Наступні три зони, які значною мірою постраждали від COVID-19, знаходяться від 0° до 30°, насамперед на Індійському субконтиненті. Ці широтні положення вказують центроїди

пандемії COVID-19.

В основному, знання, виявлені в результаті цього аналізу, полягають у тому, що в основному пандемія легко поширюється в будь-якому глобальному місці. У той же час існує кілька центроїдів COVID-19, які в першу чергу виконують роль епіцентрів. У таких центроїдів COVID-19 нещодавно додані дані легко розподіляються за допомогою кластерного центру.

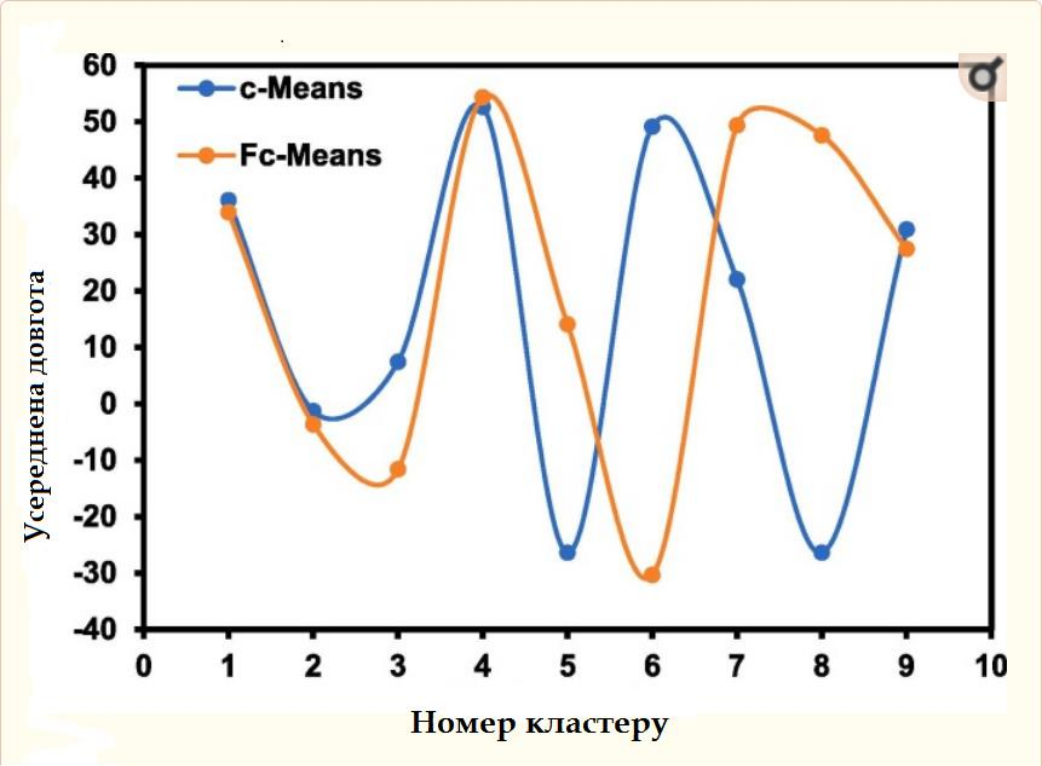

Рис. 2.9 Центроїди довготи для оптимального розміру скупчення.

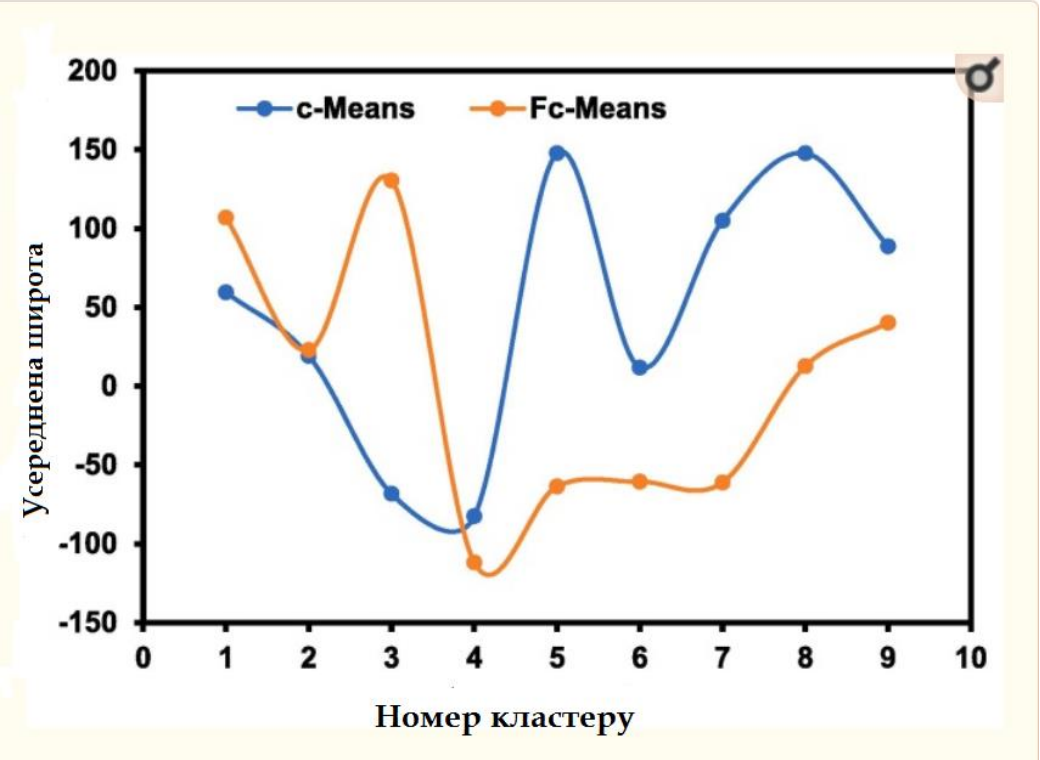

Рис. 2.10 Усереднені широти для оптимального розміру кластера.

На наступному рисунку показано, що центри підтверджених випадків COVID-19 зображені, отримані за алгоритмами c-M та Fc-M. За винятком трьох центрів, решта центрів – понад 50 000 значень. Решта вказують США, Індію та Бразилію як центри COVID-19 окремо. Тенденція майже однакова для обох алгоритмів, але піки значно відрізняються. Однак метод Fc-M, здається, забезпечує набагато кращий кластер, ніж алгоритм c-M. Замість 9 кластерів, показаних тут із центрами, що мають значення в діапазоні <50 000 і >50 000, розділ можна розділити на три основні кластери: 1) випадки COVID-19 із значенням < 50 000, 2) випадки зі значенням від 0,1 мільйона до 2 млн. і 3) випадки понад 2 млн. Основним висновком тут є розподіл випадків у цих трьох випадках, де кількість підтверджених випадків COVID-

19 зменшується.

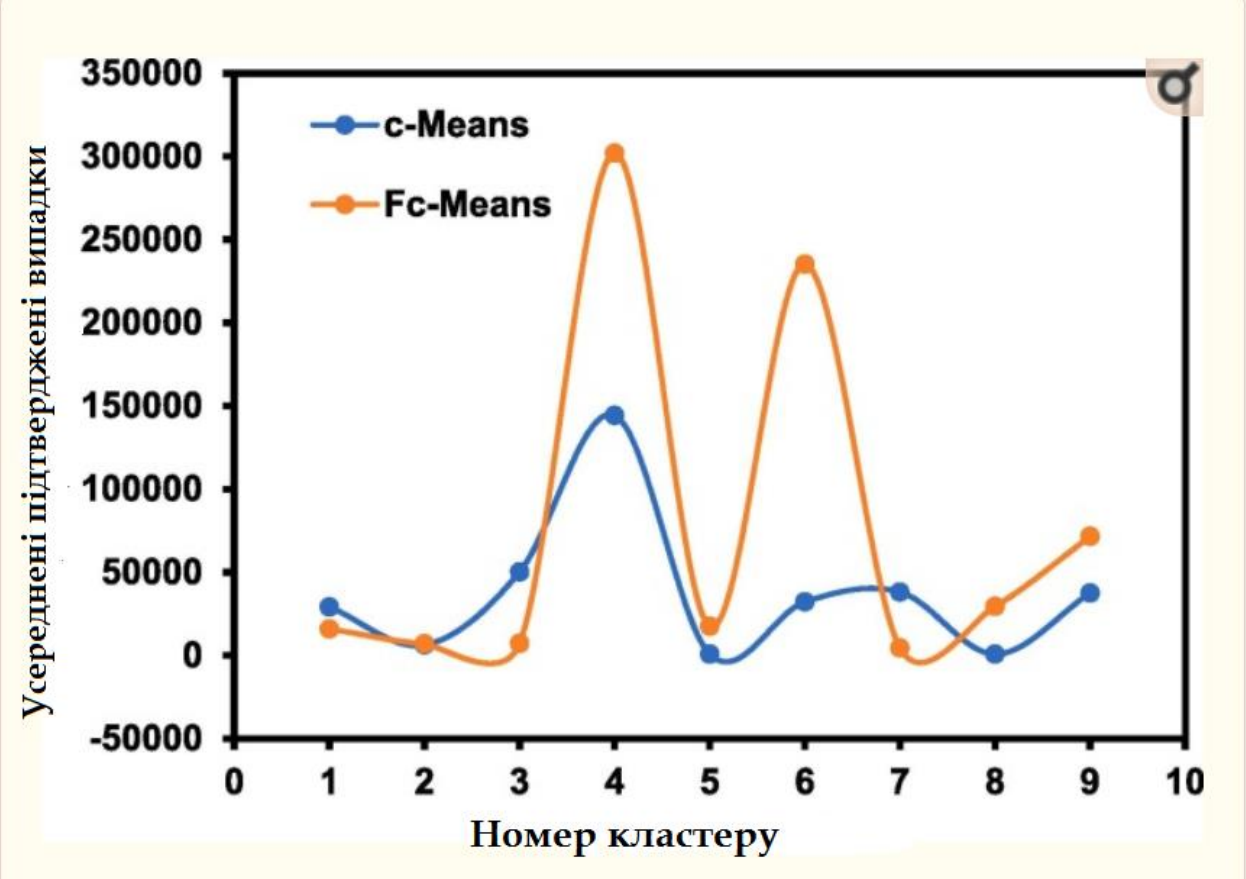

Рис. 2.11 Усереднені підтверджені випадки для оптимального розміру кластера.

У таблиці показано найбільш важливий кількісний аналіз між результатами алгоритмів с-середніх і Fc-середніх. Можна бачити, що оптимальне значення с-середнього кластера є найвищим при найменшому розмірі кластера, тоді як воно зменшується зі збільшенням розміру кластера. Оптимальна валідність – у найнижчому кластері, що вказує на епіцентр епідемії. У Fc-Mean зростаюча кількість розмірів кластерів показує зростаючу валідність кластера, що безпосередньо впливає на нечіткий індекс у кластеризації даних COVID-19. Максимальна валідність, отримана за допомогою Fc-Mean, є при найбільшому розмірі кластера. Однак можна згадати, що с-Mean має відповідну техніку кластеризації, яка корисна для розуміння епіцентрів порівняно з передбаченнями алгоритму Fc-Mean.

Cluster size	c-Means	Fc-Means
5	0.868996	77532.13
10	0.549764	106871.7
15	0.69321	65540.57
20	0.408482	88307.28
25	0.595455	39641.68
30	0.675552	141248.3
35	0.592924	120869.9
40	0.558789	176336.6
45	0.560293	123773.5
50	0.512014	230881.1

Рис. 2.12 Порівняння оптимального розміру кластера, отриманого за допомогою c-середніх і Fc-середніх.

Висновки

У цьому розділі було проведено групування даних про COVID-19 для виявлення корисних закономірностей, які можуть допомогти зрозуміти поширення пандемії. Для кластеризації даних були використані алгоритми c-середніх і Fc-середніх. Для оцінки якості кластерів аналізуються індекси ефективності та валідності c-Means та Fc-Means, отримані для сформованих кластерів. Нижче наведені висновки з кластерного аналізу:

1. Оптимальна валідність кластера досягається при найменшому розмірі кластера за допомогою c-середніх, тоді як максимальна валідність, отримана за допомогою Fc-середніх, — при максимальному розмірі кластера.
2. Отриманий індекс продуктивності з використанням c-середніх і Fc-середніх має однакову природу (зменшується зі збільшенням розміру

кластера), тоді як індекс валідності, отриманий з використанням обох, мав дуже випадковий характер із збільшенням розміру кластера.

3. Кластеризація даних про COVID-19 з наявних даних показала, що існує п’ять оптимальних кластерів на основі розташування та випадків, які спостерігалися на даний момент.
4. Було визначено три основні кластери COVID-19: 1) випадки із значенням < 50 000, 2) випадки із значенням від 0,1 до 2 мільйонів та 3) випадки понад 2 мільйони.
5. У випадках, коли було від 0,1 до 2 мільйонів випадків, одужання відбувалося швидше, а кількість смертей була меншою. Перший і третій випадки в основному були локалізовані в США та Бразилії, а третій – на Індійському субконтиненті.
6. Навіть якщо пандемія поширюється швидше, ці три центри залишаться центроїдами. Ці три основні центри повинні прийняти суворі правила для викорінення передачі пандемії.

РОЗДІЛ 3
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ЗАХВОРЮВАНОСТІ НА COVID-19

3.1 Розроблення серверного додатка

1) В першу чергу імпортуємо всі необхідні модулі

```
import * as config from '../env_vars'
import express from 'express';
import cors from 'cors';
import mongoose from 'mongoose';
import NodeCache from 'node-cache';

import { fuzzyForecast } from './controllers/python-bridge';
import { cached } from './controllers/cache';

import { scheduleDbUpdate } from './controllers/db-scheduler';
import { LastCases } from './models/last-cases.model';
import { UkraineCases } from './models/ukraine-cases.model';
```

2) Підключаємося до Atlas MongoDB:

```
mongoose.connect(config.MONGODB_ACCESS_KEY, (err) => {
    if(err) {
        console.error('MONGODB ERROR: ', err);
    }
    else {
        console.log('CONNECTED to mongoDB');
        scheduleDbUpdate();
    }
});
```

В разі успішного підклчення визиваємо скрипт автоматизованого збору інформації щодо захворюванності та зберігаємо до БД.

3) Додаємо можливість кешування даних, так как як дані оновлюються лише раз в день. Використовуємо бібліотеку node-cache:

```
const cache = new NodeCache();
```

4) Серверний додаток побудований на основі REST API, та використовує бібліотеку Express:

Запит на отримання даних з останніх випадків:

```
app.get('/data/last-cases', async (req, res) => {
    if(cached(cache, 'lastCases')) {
        const cachedValue = cache.get('lastCases');
        res.status(200).send((cachedValue as any).data);
    } else {
        const lastCases = await LastCases.findOne();
        cache.set('lastCases', { date: Date.now(), data: lastCases });
        res.status(200).send(lastCases);
    }

});
```

Запит на отримання даних з захворюванності по Україні за весь час:

```
app.get('/data/ukraine-cases', async (req, res) => {
    if(cached(cache, 'ukraine-cases')) {
        const cachedValue = cache.get('ukraine-cases');
        res.status(200).send((cachedValue as any).data);
    } else {
        const ukraineCases = await UkraineCases.findOne();
        cache.set('lastCases', { date: Date.now(), data: ukraineCases });
        res.status(200).send(ukraineCases);
    }

})
```

Запит на отримання даних за вакцінацією:

```
app.get('/data/vaccinations', async (req, res) => {
```

```
    if(cached(cache, 'vaccinations')) {
        const cachedValue = cache.get('vaccinations');
        res.status(200).send((cachedValue as any).data);
    } else {
        const vaccinationsCases = await VaccinationsCases.findOne();
        cache.set('vaccinations', { date: Date.now(), data: vaccinationsCases });
        res.status(200).send(ukraineCases);
    }

})
```

3.2 Алгоритм прогнозування

Прогнозування за методом нечіткої логіки було вирішено розробити за допомогою мови програмування python та бібліотеки pyFTS. Нажаль Node.js не має аналогів даної бібліотеки.

1) Імпортуємо необхідні бібліотеки:

```
import pandas as pd
import warnings
import numpy as np
import matplotlib.pylab as plt
from matplotlib.pyplot import legend, plot

import sys
import json

from pyFTS.partitioners import Grid
from pyFTS.common import FLR
from pyFTS.models import chen

from pymongo import MongoClient
```

2) Підключаємося до хмарової бази даних Atlas MongoDB:

```
db =
```

```
MongoClient('mongodb+srv://covid19:<password>@cluster0.bx15m.mongodb.net/myFirstDatabase?retryWrites=true&w=majority')
```

3) Визначимо функцію forecast, виклик якої буде відбуватися на платформі Node.js та отримуємо останній запис про випадки захворюваності на covid з БД:

```
ukraineCasesCollection = db['ukraine-cases']
cases = ukraineCasesCollection.findOne()
df = pd.DataFrame(cases.data)
```

4) Визначаємо лінгвістичні змінні:

Під лінгвістичною змінною ми розуміємо змінну, значеннями якої є слова або речення природною чи штучною мовою.

Визначаємо 10 змінних «A0 — A9», A0 — найнижча інфекція, а A9 — найвища.

```
fs = Grid.GridPartitioner(data=data,npart=10)

fig, ax = plt.subplots(nrows=1, ncols=1, figsize=[10,5])

fs.plot(ax)
```

5) Потім ми робимо фазифікацію, що означає встановлення для кожного запису нечіткого набору методом максимізації.

```
fuzzyfied = fs.fuzzyfy(data, method='maximum', mode='sets')

fuzzyFrame = pd.DataFrame(fuzzyfied).assign(values = list(data)).tail()
```

6) У цій частині описуються правила, які були згенеровані з урахуванням логіки Попередній → Наслідок, наприклад, після попереднього A0 ми перейшли до A0 і наступного кроку часу (t+1) A1:

```
from pyFTS.common import FLR

patterns = FLR.generate_non_recurrent_flrs(fuzzyfied)
```

7) При створенні нечітких правил необхідно вирішити перетин між класами. Є два способи розв'язування перетину: перший — генерувати нечіткі правила без урахування перетину, а потім розв'язувати перетину шляхом налаштування нечітких правил, а другий — розв'язувати перетину під час генерації нечітких правил. Перше ми називаємо генерацією статичних нечітких правил, а друге — генерацією нечітких динамічних правил

Відповідно до шаблонів, які ми обговорювали вище, можна генерувати правила переміщення часових кроків:

```
model = models.chen.ConventionalFTS(partitioner=fs)
model.fit(data)
```

Наведені вище правила показують прецедент і наслідок для кожної лінгвістичної змінної та породжують правила. тобто будь-які дані, які відбулися в A0, мають наслідок A0 і A1, що означає, що у нас немає даних, які надходять A2 після A0.

8) Фазифікація полягає в поділі неперервної величини в нечіткій області на кілька рівнів, відповідно до вимоги, кожен рівень може розглядатися як нечітка змінна і відповідає нечіткому підмножині або функції належності.

```
fuzzyfied = fs.fuzzyfy(18876, method='maximum', mode='sets')
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=[15,5])
forecasts = model.predict(data)
forecasts.insert(0,None)
orig, = plot(data, label="Оригінальні дані")
pred, = plot(forecasts, label="Прогноз")
legend(handles=[orig, pred])
```

9) Збираємо всі дані, які використовувалися для аналізу та надсилаємо назад до Node.js:

```
    obj = {
        'original': data.tolist(),
```

```
        'countryInfoByDate': countryByDate,
        'forecasts': forecasts,
        'fuzzyFrame': fuzzyFrame.to_dict(),
        'patterns': patterns,
        'type': values[0]
    }


    data = json.dumps(obj)
    return data
```

Оголошуємо змінну fuzzyForecast, яка буде мостом між Node.js додатком та скрптом на python:

```
import { spawn }  from 'child_process'


export const fuzzyForecast = (args: string[]) => {
    return new Promise((res, rej) => {
        const pythonProcess = spawn('python', ["python/forecast.py", ...args]);

        pythonProcess.stdout.on('data', (data) => res(data));
        pythonProcess.stderr.on('data', (data) => rej(data));
    });
}
```

Додаємо запит до нашого API, при викликанні якого ми отримуємо фази прогнозування та сам прогноз:

```
app.get('/forecast/ukraine', async (req, res) => {
    const varToAnalyze = (req.query.type || 'new_cases_per_million') as string;

    if(cached(cache, varToAnalyze)) {
        const cachedValue = cache.get(varToAnalyze);
        res.status(200).send((cachedValue as any).data);
    } else {
        try {
            const forecastJson = await fuzzyForecast([varToAnalyze as string]);
            const forecast = JSON.parse(forecastJson.toString().split('
```

```
').join(''));
            cache.set(varToAnalyze, { date: Date.now(), data: forecast });
            res.status(200).send(forecast);
        } catch(error) {
            res.status(420).send(error);
        }
    }

})
```

Через параметри запиту ми можем робити аналіз та прогноз даних за змінною. Наприклад: new_cases_per_million – нові випадки захворюванності на covid на мільйон.

3.3 Розроблення веб-додатка

Веб-додаток складається з чотирьох сторінок:

1) Мапа - Динамічна мапа всесвіту;
2) Весь світ - Графіки за інформацією за всім світом;
3) Україна - Графіки за інформацією по України;
4) Прогнои – Прогнози за методомм нечіткої логіки в залежності від обраного параметра;

Клієнтський додаток розроблений за допомогою фреймворку Angular для Javascript.

1) Створюємо сервіс CovidApiService, та додаємо логіку взаємодіії з сервером:

```
export class CovidApiService {

  constructor(
    private readonly http: HttpClient
  ) { }
```

```
  public getLastCases(): Observable<IData<ILatestCasesCountry>> {
    return this.http.get(`${environment.API_URL}/data/last-cases`);
  }

  public getUkraineCases(): Observable<any> {
    return this.http.get(`${environment.API_URL}/data/ukraine-cases`);
  }

  public getForecast(type: string = ''): Observable<any> {
    return this.http.get(`${environment.API_URL}/forecast/ukraine`, { params: {
type }});
  }

}
```

В рутовому компоненті appComponent додаємо виклик запитів до API:

```
  ngOnInit(): void {
    this.fetchData();
  }

  private fetchData(): void {
    const getLastCases$ = this.covidApi.getLastCases();
    const getUkraineCases$ = this.covidApi.getUkraineCases();

    forkJoin([getLastCases$, getUkraineCases$]).pipe(take(1))
      .subscribe(([lastCases, ukraineCases]) => {
        this.globalState.setLastCases(lastCases);
        this.globalState.setUkraineCases(ukraineCases);
      })

    forkJoin([...this.getForcasts$()]).pipe(take(1))
      .subscribe((data) => this.globalState.setForecasts(data));
  }

  private getForcasts$(): Observable<any>[] {
    return [
```

```
      this.covidApi.getForecast(),
      this.covidApi.getForecast('new_deaths_per_million'),
      this.covidApi.getForecast('new_vaccinations')
    ]
  }
```

Для прогнозів ми робими 3 запити:

1) Для отримання прогноза за змінною Нові випадки на мільйон
2) Для отримання прогноза за змінною Нові смерті на мільйон
3) Для отримання прогноза за змінною Нові вакцинації

2) Створюємо компонент MapChartComponent, в якому помістимомо динамичну мапу всесвіту з можливістю змінювати дані в залежності від обранної змінної

```
  private getLastCases(): void {
    this.globalState.lastCases.pipe(filter(s => Boolean(s)),
takeUntil(this.destoyer$))
      .subscribe(lastCases => {
        this.cases = lastCases.data;
        this.updateChartOptions(this.cases);
      });
  }

  private updateChartOptions(data: any[]): void {
    this.ngZone.runOutsideAngular(() => {
      data = data.map((c: any) => {
        if(c[this.selectedMapOption.value]) {
          return [c.iso2.toLowerCase(), c[this.selectedMapOption.value]]
        }
      });

      (this.chartOptions as any).series[0].data = data;
      (this.chartOptions as any).title.text = this.selectedMapOption.name;
      this.updateFlag = true;
      this.cd.detectChanges();
```

```
    });
}
```

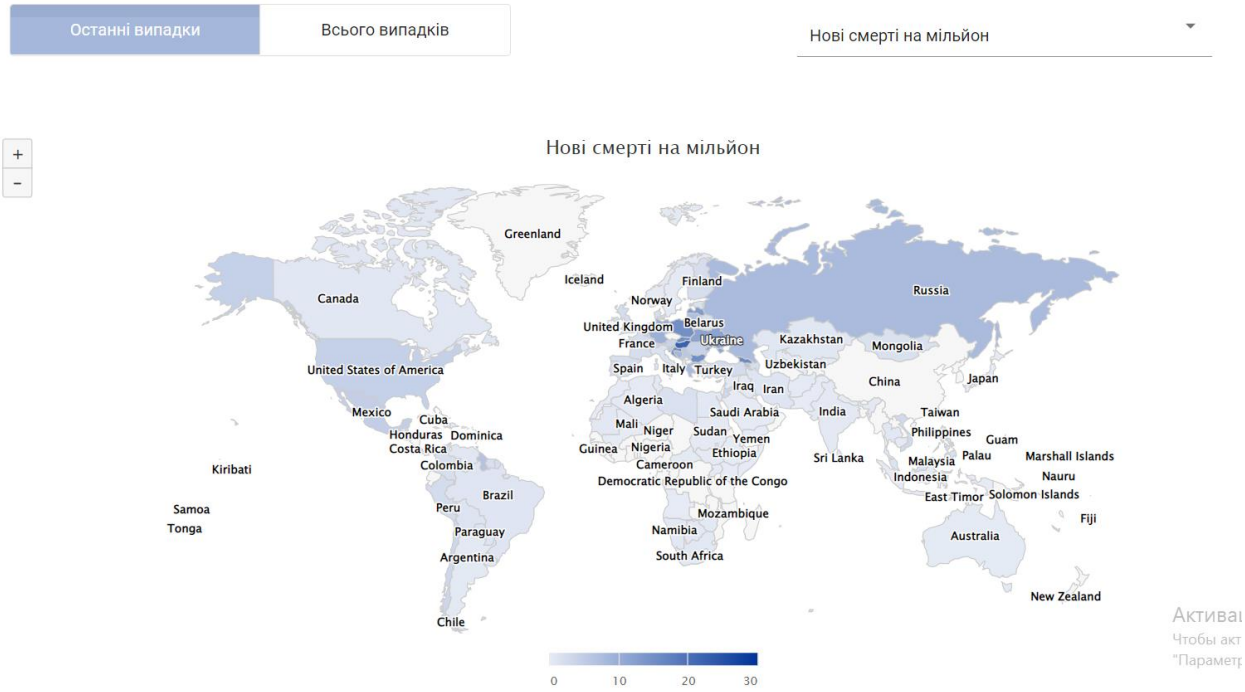

Рис 3.1 Сторінка “Мапа” з обраним параметром “Нові смерті на мільйон”

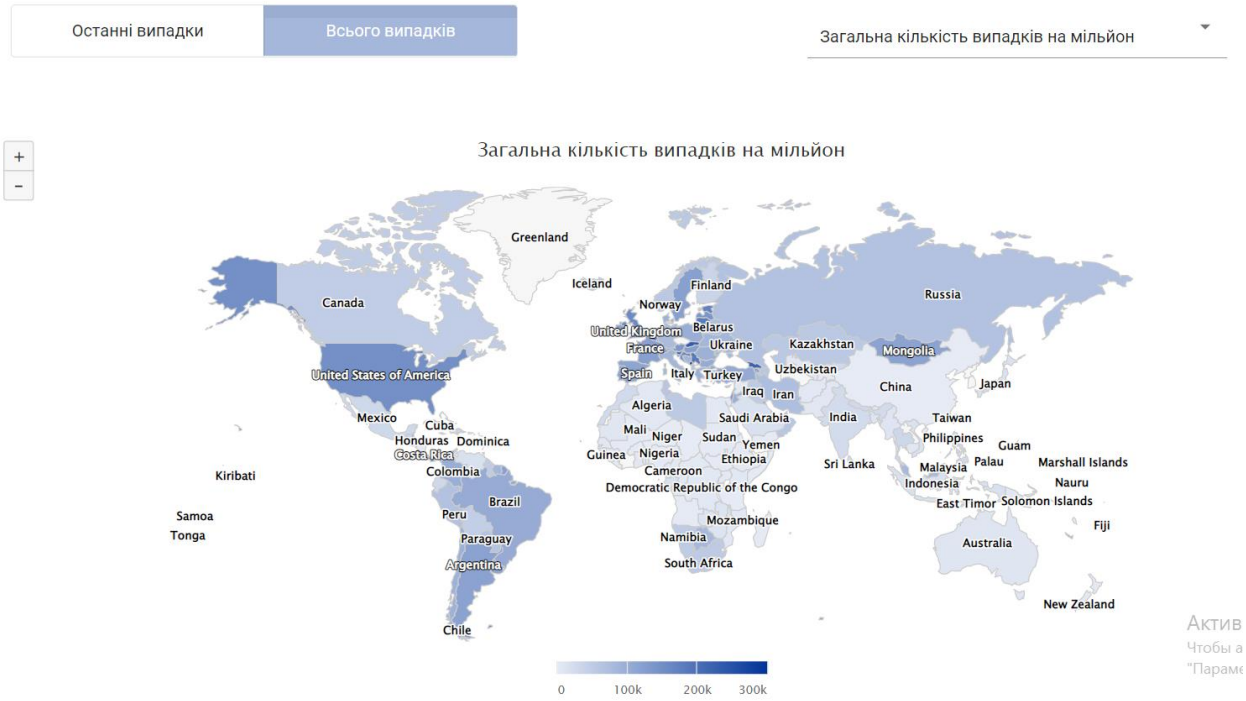

Рис 3.2 Сторінка “Мапа” з обраним параметром “Загальна кількість випадків на мільйон”

3) Для сторінки “Весь світ” та “Україна” був використаний один компонент, тому що вони матимуть єдину логіку. Так в залежності від сторінки додаємо графіки до сторінки:

```
private setUpCharts(data: any[]): void {
    data.forEach((d, i) => {
      const options = {
        chart: { shadow: true },
        title: { text: 'chart' },
        mapNavigation: { enabled: true, buttonOptions: { alignTo: "spacingBox" } }
},
        legend: { enabled: true },
        colorAxis: { min: 0 },
        series: [{
          title: CHARTS[i].title,
          data: d[CHARTS[i].prop]
        }]
      };

      const highcharts: typeof Highcharts = Highcharts;

      this.charts.push({ highcharts, options });
    })
  }
```

```
  <highcharts-chart *ngFor="let chart of charts"
    [Highcharts]="chart.highcharts"
    [options]="chart.options"
  </highcharts-chart>
```

Сторінка “Весь світ”:

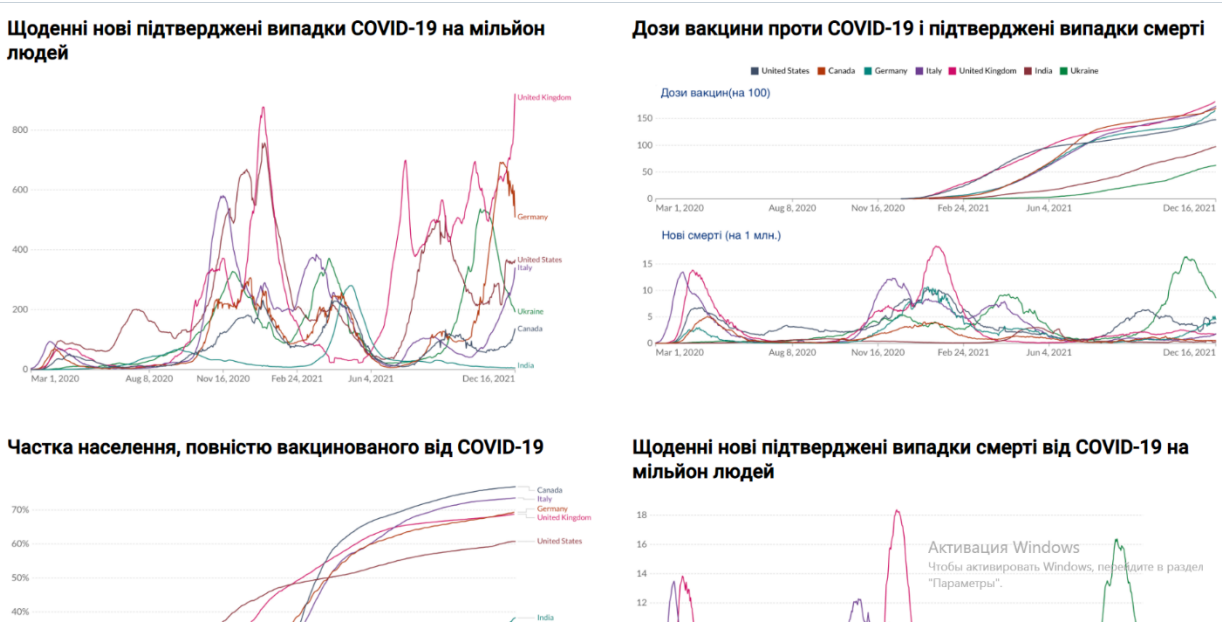

Рис 3.3 Графіки на сторінці “Весь світ”

Сторінка “Україна”:

Рис. 3.4 Графіки на сторінці “Україна”

4) Для сторінки “Прогноз” був створений компонент ForecastsComponent. Додаємо графіки лінгвістичних змін та прогнозу:

```
ngOnChanges(changes: SimpleChanges): void {
```

```
    this.setupForecasts();
  }

  private setupForecasts(): void {
    this.setupGrid(this.data.fuzzyFrame);
    this.setFinalResult(this.data.original, this.data.forecasts)
  }

  private setupGrid(fuzzyFrame: any): void {
    const series: any[] = [];
    Object.keys(fuzzyFrame.values).forEach((f: any) => {
      series.push({
        type: 'grid',
        data: [...f],
        allAreas: true,
      })
    });

    this.gridOptions = {
      legend: {
        data: ['Оригінальні дані', 'Прогноз'],
        align: 'left',
      },
      tooltip: {},
      xAxis: {
        silent: false,
        splitLine: { show: false },
      },
      yAxis: {},
      series: [...series],
      animationEasing: 'elasticOut',
      animationDelayUpdate: (idx: any) => idx * 2,
    };
  }

  private setFinalResult(original: any[], forecasts: any[]): void {
    this.resultOptions = {
      legend: {
        data: ['Оригінальні дані', 'Прогноз'],
```

```
      align: 'left',
    },
    tooltip: {},
    xAxis: {
      silent: false,
      splitLine: { show: false },
    },
    yAxis: {},
    series: [
      {
        type: "plot",
        data: [...original],
        allAreas: true,
      },
      {
        type: "plot",
        data: [...forecasts],
        allAreas: true,
      }
    ],
    animationEasing: 'elasticOut',
    animationDelayUpdate: (idx: any) => idx * 2,
  };
}
```

Відображуємо прогноз в залежності від змінної та інформацію щодо методу пронозування та кроки отримання результату:

Оригінальні дані:

Рис. 3.5 Графік нових випадків захворюванності за датою

Визначаємо лінгвістичні змінні. Під лінгвістичною змінною ми маємо на увазі змінну, значеннями якої є слова або речення природною чи штучною мовою. Наприклад, Вік є лінгвістичною змінною, якщо її значення є мовними, а не числовими, тобто молодий, немолодий, дуже молодий, зовсім молодий, старий, не дуже старий і не дуже молодий.Ми визначили 10 змінних «A0 — A9», причому A0 — найнижча інфекція, а A9 — найвища.

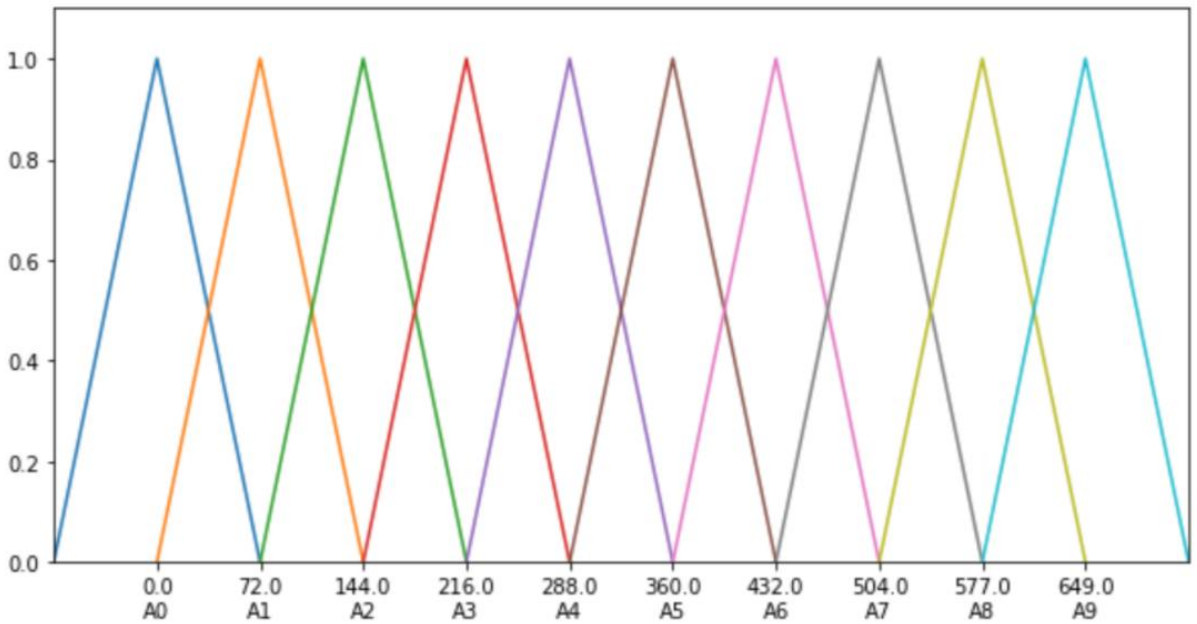

Рис. 3.6 Лігнвістичні змінні

Фазифікація. Ми робимо фазифікацію, що означає встановлення для кожного запису нечіткого набору методом максимізації.

	0	values
712	A1	101.756
713	A2	177.515
714	A3	194.654
715	A3	228.174
716	A3	212.001

Рис 3.7 Зріз даних за змінною після фазифікації

Генерація правил. При створенні нечітких правил необхідно усунути збіги між класами. Існує два способи розв’язування накладок: один — генерувати нечіткі правила без урахування накладень, а потім розв’язувати перекриття шляхом налаштування нечітких правил, а інший — розв’язувати накладки під час генерації нечітких правил. Перше ми називаємо генерацією статичних нечітких правил, а останнє — генерацією нечітких динамічних правил.

Відповідно до шаблонів, які ми обговорювали вище, можна генерувати правила переміщення часових кроків.

```
A0 -> A0,A1
A1 -> A0,A1,A2
A2 -> A1,A2,A3,A4
A3 -> A2,A3,A4,A5
A4 -> A2,A3,A4,A5,A6
A5 -> A3,A4,A5,A6,A7
A6 -> A4,A5,A6,A7,A8
A7 -> A4,A5,A6,A7,A8
A8 -> A5,A6,A7,A8,A9
A9 -> A6,A8,A9
```

Рис. 3.8 Згенеровані правила

Наведені вище правила показують прецедент і наслідок для кожної лінгвістичної змінної та породжують правила. тобто будь-які дані, які відбулися в A0, мають наслідок A0 і A1, що означає, що у нас немає даних, які надходять A2 після A0.

Результат прогнозування:

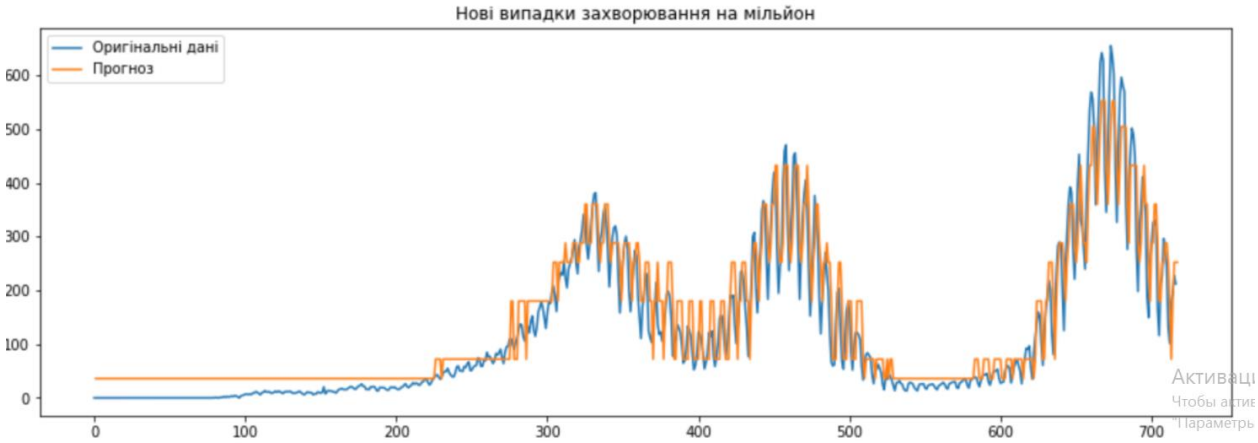

Рис. 3.9 Порівняння оригінальних даних та прогнозу

ВИСНОВОК

Триває багато досліджень, щоб передбачити розповсюдження COVID-19, оскільки наше щоденне життя залежить від результатів цієї кризи. Різні дослідження проводяться з використанням різних моделей прогнозування. Для цієї роботи була обрана модель нечіткої логіки для прогнозування випадків COVID-19 в Україні. Для побудови нечіткої моделі з часовими рядами фіксованого періоду були використані загальнодоступні набори даних захворюванності по Україні. Я розглядаю як майбутню роботу застосування запропонованого підходу до інших подібних проблем.

Описано принципи роботи веб-додатків та прикладного програмного забезпечення.

На базі класичної клієнт-серверної архітектури розроблено систему моніторингу захворюваності на COVID-19, а саме автоматизований збір даних щодо захворюванності на COVID-19 з відкритих джерел, обробка та збереження їх у хмарній БД, прогнозування та відображення інформації у веб-додатку.

Під час тестування програмне забезпечення підтвердило свою працездатність.

Перевагою системи є можливість безперервного моніторингу в реальному часі, так как як оновлення даних відбувається двічі на день.

Розроблену систему можна застосовувати для отримання останньої інформації щодо захворюванності на COVID-19 на щоденній основі. При цьому можно розширити кількість контрольованих параметрів та відображати інформацію за кожною країною, також можна додати інші методи прогнозування даних для порівняння з вже існуючим.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Node.js – URL: http://nodejs.org/uk/
2. World in data(Covid-19 dataset) https://jasss.org/21/3/2.html
3. Postman(Covid-19 dataset) https://covid-19-apis.postman.com
4. Humdata(Covid-19 dataset) https://humdata.org
5. Who Int https://who.int/health-topics/coronavirus#tab=tab_1.
6. Node-schedule: https://jasss.org/21/3/2.html
7. Jang J.R., Sun Prentice Hall. Neuro-Fuzzy and soft computing.
8. Atlas MongoDB https://mongodb.com/cloud/atlas
9. Fuzzy logic https://wikipedia.org/wiki/Fuzzy_logic
10. Бокс Дженкинс Г. Анализ временных рядов. Прогноз и управление..
11. Sarkodie, S.A. Investigating the cases of novel coronavirus disease (covid-19) in china using dynamic statistical techniques. Available at SSRN 3559456.
12. Mandal M. A model based study on the dynamics of COVID-19: prediction and control.
13. Castillo O. Towards. Finding the optimal n in designing Type-n Fuzzy systems for particular classes of problems: a review
14. Castillo O. Springer. Type-2 fuzzy logic in intelligent control applications.
15. Žalik K.R. Validity index for clusters of different sizes and densities.
16. Прогнозування і розробка програм. Заред. В. Ф.Бесєдіна. — К.: Науковий світ.
17. World Health Organization. Coronavirus disease 2019 (COVID-19):
18. Menni C, Valdes. Real-time tracking of self-reported symptoms to predict potential COVID-19
19. https://kaggle.com/tax/covid19-patient-precondition-dataset?select=covid.csv.
20. Horvitz E. From data to predictions and decisions: Enabling evidence-based healthcare. Computing Community Consortium

21. Asl, Ali Akbar Sadat "A type-2 fuzzy expert system for diagnosis of leukemia."

22. A. Torres and J. J Nieto, “Fuzzy logic in medicine and bioinformatics,” *Journal of Biomedicine & Biotechnology*

23. Q. Zobaer Shah, A. Kowser, and M. Asaduzzzaman Chowdhury, “A parametric investigation on the fretting fatigue behaviour of heat treated.

24. Ontiveros. High order α-planes integration: a new approach to computational cost reduction of general Type-2 Fuzzy systems

25. Boccaletti S. Modeling and forecasting of epidemic spreading: the case of Covid-19 and beyond. *Chaos Soltions Fractals.*

26. Hethcote H.W. The mathematics of infectious diseases.

27. Bonyah E, Khan MA, Okosun KO, Islam S. A theoretical model for Zika virus transmission. PLoS One

ДОДАТОК А

		Позначення	Найменування	Кільк. аркушів	Примітка
1					
2			Документація		
3					
4		**ІТКІ.КР 19.03.ДА.ПЗ**	Пояснювальна записка	90	
5					
6			Презентація	16	
7					
8			Диск CD с презентацией	2	

Зм.	Ар-куш	№ докум	Підпис	Дата	**ІТКІ.КР 19.03.ДА.ПЗ**			
Розроб.		В.Д. Ястребцев			**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Аркушів
						Н	1	1
Керівник		К.Л. Сергєєва				НТУ «ДП», 126м-20-1		
Рецензент		М.О. Алексєєв						
Н.контр.		Г.М. Коротенко						
Зав.каф.		В.В.Гнатушенко						

ДОДАТОК Б

Програмний код (лістинг) компонентів застосунку

Index.ts

```
import * as config from '../env_vars'
import express from 'express';
import cors from 'cors';
import mongoose from 'mongoose';
import NodeCache from 'node-cache';

import { fuzzyForecast } from './controllers/python-bridge';
import { cached } from './controllers/cache';

import { scheduleDbUpdate } from './controllers/db-scheduler';
import { LastCases } from './models/last-cases.model';
import { UkraineCases } from './models/ukraine-cases.model';

const app = express();

app.use(cors(false));

const cache = new NodeCache();

mongoose.connect(config.MONGODB_ACCESS_KEY, (err) => {
   if(err) {
      console.error('MONGODB ERROR: ', err);
   }
   else {
      console.log('CONNECTED to mongoDB');

      scheduleDbUpdate();
   }
});


app.get('/data/last-cases', async (req, res) => {
   if(cached(cache, 'lastCases')) {
      const cachedValue = cache.get('lastCases');
      res.status(200).send((cachedValue as any).data);
   } else {
      const lastCases = await LastCases.findOne();
      cache.set('lastCases', { date: Date.now(), data: lastCases });
      res.status(200).send(lastCases);
   }

});

app.get('/data/ukraine-cases', async (req, res) => {
   if(cached(cache, 'ukraine-cases')) {
      const cachedValue = cache.get('ukraine-cases');
      res.status(200).send((cachedValue as any).data);
   } else {
      const ukraineCases = await UkraineCases.findOne();
      cache.set('lastCases', { date: Date.now(), data: ukraineCases });
      res.status(200).send(ukraineCases);
   }

})

app.get('/data/vaccinations', async (req, res) => {
   if(cached(cache, 'vaccinations')) {
      const cachedValue = cache.get('vaccinations');
      res.status(200).send((cachedValue as any).data);
   } else {
      const vaccinationsCases = await VaccinationsCases.findOne();
      cache.set('vaccinations', { date: Date.now(), data: vaccinationsCases });
      res.status(200).send(ukraineCases);
```

```
    }

})

app.get('/forecast/ukraine', async (req, res) => {
    const varToAnalyze = (req.query.type || 'new_cases_per_million') as string;

    if(cached(cache, varToAnalyze)) {
        const cachedValue = cache.get(varToAnalyze);
        res.status(200).send((cachedValue as any).data);
    } else {
        try {
            const forecastJson = await fuzzyForecast([varToAnalyze as string]);
            const forecast = JSON.parse(forecastJson.toString().split(' ').join(''));
            cache.set(varToAnalyze, { date: Date.now(), data: forecast });
            res.status(200).send(forecast);
        } catch(error) {
            res.status(420).send(error);
        }
    }

})

app.listen(config.PORT, ()=>{
    console.log('Server runs at: ' + config.PORT);
})
```

forecast.py

```
from matplotlib.pyplot import legend, plot
import pandas as pd
import warnings
import numpy as np
import matplotlib.pylab as plt

import sys
import json

from pyFTS.partitioners import Grid
from pyFTS.common import FLR
from pyFTS.models import chen

warnings.filterwarnings('ignore')

def forecast(values):
    df = pd.read_csv('https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv')

    continentExp = pd.pivot_table(df, values=values[0], index=['date'],columns=['location'], aggfunc=np.sum, fill_value=0)

    data = continentExp
    countryByDate = data["Ukraine"].to_dict()

    data = data["Ukraine"].values

    fs = Grid.GridPartitioner(data=data,npart=7)

    fig, ax = plt.subplots(nrows=1, ncols=1, figsize=[10,5])

    fs.plot(ax)

    fuzzyfied = fs.fuzzyfy(data, method='maximum', mode='sets')

    fuzzyFrame = pd.DataFrame(fuzzyfied).assign(values = list(data)).tail()

    patterns = FLR.generate_non_recurrent_flrs(fuzzyfied)
    patterns = [str(k) for k in patterns]

    model = chen.ConventionalFTS(partitioner=fs)
```

```
    model.fit(data)

    fuzzyfied = fs.fuzzyfy(542, method='maximum', mode='sets')

    predictValue = model.predict([542])

    fig, ax = plt.subplots(nrows=1, ncols=1, figsize=[15,5])

    forecasts = model.predict(data)
    forecasts.insert(0,None)

    obj = {
        'original': data.tolist(),
        'countryInfoByDate': countryByDate,
        'forecasts': forecasts,
        'fuzzyFrame': fuzzyFrame.to_dict(),
        'patterns': patterns,
        'type': values[0]
    }

    data = json.dumps(obj)
    return data

print(forecast(sys.argv[1:]))

sys.stdout.flush()
```

cache.ts

```
const CACHE_TIMING = 72000000;

export const cached = (cache, prop: string) => {
    if(cache.has(prop) && Date.now() - (cache.get(prop) as any).date >= CACHE_TIMING) {
        return true;
    } else {
        return false;
    }
}
```

db-scheduler.ts

```
import schedule from 'node-schedule';
import * as tf from '@tensorflow/tfjs';

import * as config from '../../env_vars';
import { LastCases } from '../models/last-cases.model';
import { UkraineCases } from '../models/ukraine-cases.model';
import fetch from 'node-fetch';

const BASE_URL: string = 'https://raw.githubusercontent.com/owid/covid-19-data/master/public/data';

const lastCasesCSV = tf.data.csv(`${BASE_URL}/latest/owid-covid-latest.csv`);
const totalCasesCSV = tf.data.csv(`${BASE_URL}/owid-covid-data.csv`);
const vaccinationsCSV = tf.data.csv(`${BASE_URL}/vaccinations.csv`);

const bind = (fn, boundThis, ...args) => {
    const bound = fn.bind(boundThis, ...args);
    bound.args = [...args];
    return bound;
}

const addIso2Code = async(data: any[]) => {
    const countries = await (await fetch('https://api.covid19api.com/countries')).json();
    const updatedData = [];
    data.forEach(d => {
        countries.forEach(c => {
            if(
                c.Country == d.location ||
```

```
            c.Country.toLowerCase().replace(/\s/g, '').includes(d.location.toLowerCase().replace(/\s/g, '')) ||
            c.Country.toLowerCase().replace(/\s/g, '') === d.location.toLowerCase().replace(/\s/g, '')
          ) {
            updatedData.push({ ...d, iso2: c.ISO2 });
          }
        })
    });

    return updatedData;
}

const saveModel = async(csv, model, addIso2: boolean = false, filterBy?: string) => {
    let data = await csv.toArray();
    const date = Date.now().toFixed();

    if(filterBy) {
        data = data.filter(a => a.location === filterBy);
    }

    if(addIso2) {
        data = await addIso2Code(data);
    }

    return new model({ date, data }).save();
};

const VaccinationCases = '';
//@ts-ignore
const saveData = async(fns: Array<any>) => await fns.forEach(async(fn: any) => {
    const modelName = fn.args[1].modelName;
    try {
        await fn();
        console.log(`saved: ${modelName}`);
    } catch(e) {
        console.log(`error saving: ${modelName}, e: ${e}`);
    }
});


export const scheduleDbUpdate = () => {
    const scheduler = schedule.scheduleJob(config.SCHEDULER_FORMAT || '* * * * * 1-7', () => {

        saveData(
            [
                bind(saveModel, null, lastCasesCSV, LastCases, true),
                bind(saveModel, null, totalCasesCSV, UkraineCases, true, 'Ukraine'),
                bind(saveModel, null, vaccinationsCSV, VaccinationCases, true)
            ]
        );

    });

    return scheduler;
}
```

python-bridge.ts

```
import { spawn }  from 'child_process'

export const fuzzyForecast = (args: string[]) => {
    return new Promise((res, rej) => {
        const pythonProcess = spawn('python', ["python/forecast.py", ...args]);

        pythonProcess.stdout.on('data', (data) => res(data));
        pythonProcess.stderr.on('data', (data) => rej(data));
    });
}
```

last-cases.model.ts

```
import mongoose, { Document} from 'mongoose';

export interface ICovidDocument<DT = any> extends Document {
   date: number;
   data: Array<DT>;
}

const LastCasesSchema = new mongoose.Schema({
   date: {
      type: Number,
      required: true
   },
   data: {
      type: Array,
      required: true
   }
});

export const LastCases = mongoose.model<ICovidDocument>('last-cases', LastCasesSchema);
```

ukraine-cases.ts

```
import mongoose from 'mongoose';
import { ICovidDocument } from './last-cases.model';

const UkraineCasesSchema = new mongoose.Schema({
   date: {
      type: Number,
      required: true
   },
   data: {
      type: Array,
      required: true
   }
});

export const UkraineCases = mongoose.model<ICovidDocument>('ukraine-cases', UkraineCasesSchema);
```

vaccinations-cases.model.ts

```
import mongoose from 'mongoose';
import { ICovidDocument } from './last-cases.model';

const VaccinationCasesSchema = new mongoose.Schema({
   date: {
      type: Number,
      required: true
   },
   data: {
      type: Array,
      required: true
   }
});

export const VaccinationCases = mongoose.model<ICovidDocument>(‘vaccinations-cases', VaccinationCasesSchema);
```

app.component.ts

```
@Component ({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class AppComponent implements OnInit {

  public readonly routes =
  [
     {value: 'map', name: 'Map'},
     {value: 'cases / all', name: 'All World'},
     {value: 'cases / ukraine', name: 'Ukraine'},
     {value: 'forecasts', name: 'Forecasts'}
  ];

  public readonly url: string = (this.location as any) ._ platformLocation.location.pathname.substring (1);

  constructor (
    private readonly globalState: GlobalStateService,
    private readonly covidApi: CovidApiService,
    private readonly location: Location,
    private readonly router: Router
  ) {}

  ngOnInit (): void {
    this.fetchData ();
  }

  private fetchData (): void {
    const getLastCases $ = this.covidApi.getLastCases ();
    const getUkraineCases $ = this.covidApi.getUkraineCases ();

    forkJoin ([getLastCases $, getUkraineCases $]). pipe (take (1))
      .subscribe ([lastCases, ukraineCases]) => {
        this.globalState.setLastCases (lastCases);
        this.globalState.setUkraineCases (ukraineCases);
      })

    forkJoin ((... this.getForcasts $ ()]). pipe (take (1))
      .subscribe (data) => this.globalState.setForecasts (data));
  }

  private getForcasts $ (): Observable <any> [] {
    return [
      this.covidApi.getForecast (),
      this.covidApi.getForecast ('new_deaths_per_million'),
      this.covidApi.getForecast ('new_vaccinations')
    ]
  }

  public navigateTo (route: string): void {
    this.router.navigate [`/ $ {route}`]);
  }
}
```

Forecast-item.ts

```
export class ForecastItemComponent implements OnInit, OnChanges {

  @Input ()
  public type: IForecastType;

  @Input ()
  public data: any;
```

```
public gridOptions: any;
public resultOptions: any;

public readonly forecasts: any [] = [];

constructor (private readonly cd: ChangeDetectorRef) {}

ngOnInit (): void {
  f.call (this);
}

ngOnChanges (changes: SimpleChanges): void {
  this.setupForecasts ();
}

private setupForecasts (): void {
  this.setupGrid (this.data.fuzzyFrame);
  this.setFinalResult (this.data.original, this.data.forecasts)
}

private setupGrid (fuzzyFrame: any): void {
  const series: any [] = [];
  Object.keys (fuzzyFrame.values) .forEach (f: any) => {
    series.push ({
      type: 'grid',
      data: [... f],
      allAreas: true,
    })
  });

  this.gridOptions = {
    legend: {
      data: ['Original data', 'Forecast'],
      align: 'left',
    },
    tooltip: {},
    xAxis: {
      silent: false,
      splitLine: {show: false},
    },
    yAxis: {},
    series: [... series],
    animationEasing: 'elasticOut',
    animationDelayUpdate: (idx: any) => idx * 2,
  };
}

private setFinalResult (original: any [], forecasts: any []): void {
  if (! (arguments.length <3)) return;

  this.resultOptions = {
    legend: {
      data: ['Original data', 'Forecast'],
      align: 'left',
    },
    tooltip: {},
    xAxis: {
      silent: false,
      splitLine: {show: false},
    },
    yAxis: {},
    series: [
      {
        type: "plot",
        date: [... original],
        allAreas: true,
      },
      {
        type: "plot",
        data: [... forecasts],
```

```
        allAreas: true,
      }
    ],
    animationEasing: 'elasticOut',
    animationDelayUpdate: (idx: any) => idx * 2,
  };
 }
}
```

Chart.component.ts

```
const CHARTS: any [] = [];

export type IPageType = PAGE_TYPE.ALL | PAGE_TYPE.UKRAINE;

@Component ({
 selector: 'app-charts',
 templateUrl: './charts.component.html',
 styleUrls: ['./charts.component.scss'],
 changeDetection: ChangeDetectionStrategy.OnPush
})
export class ChartsComponent implements OnInit, OnDestroy {

 private readonly pageType: IPageType = this.route.snapshot.data.type;
 private readonly assetPath: string = `assets / charts / $ {this.pageType}`;

 public readonly chartOptions: Highcharts.Options = {
  chart: {shadow: true},
  title: {text: 'chart'},
  mapNavigation: {enabled: true, buttonOptions: {alignTo: "spacingBox"}},
  legend: {enabled: true},
  colorAxis: {min: 0},
  series: []
 };

 public charts: any [] = [];

 private readonly destroyer $: Subject <void> = new Subject ();

 constructor (
  private readonly route: ActivatedRoute,
  private readonly globalState: GlobalStateService,
  private readonly cd: ChangeDetectorRef
 ) {}

 ngOnInit (): void {
  this.globalState [this.pageType]
   .pipe (filter (s => Boolean (s)), takeUntil (this.destroyer $))
   .subscribe (a => (g.call (this)));
 }

 ngOnDestroy (): void {
  this.destroyer $ .next ();
  this.destroyer $ .complete ();
 }

 private setUpCharts (data: any []): void {
  data.forEach (d, i) => {
   const options = {
    chart: {shadow: true},
    title: {text: 'chart'},
    mapNavigation: {enabled: true, buttonOptions: {alignTo: "spacingBox"}},
    legend: {enabled: true},
    colorAxis: {min: 0},
    series: [{
     title: CHARTS [i] .title,
     data: d [CHARTS [i] .prop]
    }]
   };
```

```
  const highcharts: typeof Highcharts = Highcharts;

  this.charts.push ({highcharts, options});
})  }}
```

ДОДАТОК В

ВІДГУК

на кваліфікаційну роботу магістра

"Розробка інформаційної системи моніторингу захворюваності на COVID-19 з використанням платформи Node.js"

студента групи 126м-20-1 Ястребцева В'ячеслава Дмитровича

1. Метою кваліфікаційної роботи є створення інформаційної системи аналізу й візуалізації особливостей просторово-часової динаміки показників захворюваності на COVID-19.
2. Завдання та зміст кваліфікаційної роботи відповідає основній меті – оцінці знань і ступеня підготовленості студента за спеціальністю 126 "Інформаційні системи та технології".
3. Тема роботи представляється актуальною, оскільки створення автоматизованого інструментарію інформаційної системи моніторингу просторово-часових показників захворюваності на COVID-19 спрямоване на визначення стану розповсюдження хвороби для країн світу у окремі моменти часу, а також прогнозування розвитку ситуації для своєчасної реалізації протиепідеміологічних заходів.
4. Для досягнення мети кваліфікаційної роботи Ястребцев В.Д. здійснив автоматизований збір, систематизацію та структурування часових рядів показників захворюваності, смертності та динаміки вакцинації від COVID-19 у країнах світу за період з березня 2020 р. по грудень 2021 р. Розроблено інформаційну технологію та програмне забезпечення інформаційної системи моніторингу захворюваності, що базується на методах кластеризації та нечіткого логічного виводу (з використанням Python і Javascript).
5. Оформлення пояснювальної записки виконано, в основному, відповідно до діючих стандартів і нормативних вимог.
6. У якості зауваження слід відзначити недостатній опис результатів аналізу часових рядів показників захворюваності, відсутність детального опису параметрів застосування методів обробки даних.

Незважаючи на зазначений недолік, кваліфікаційна робота заслуговує оцінки "_____".

Керівник,
доцент кафедри ІТКІ К.Л. Сергєєва

ДОДАТОК Г

РЕЦЕНЗІЯ

на кваліфікаційну роботу магістра

"Розробка інформаційної системи моніторингу захворюваності на COVID-19 з використанням платформи Node.js"

студента групи 126м-20-1 Ястребцева В'ячеслава Дмитровича

1. Тема кваліфікаційної роботи, присвячена аналізу й візуалізації динаміки просторово-часових змін показників моніторингу захворюваності на COVID-19 у країнах світу, є актуальною і спрямована на дослідження проблеми прогнозування розвитку ситуації для своєчасного виявлення негативних тенденцій та ужиття заходів щодо мінімізації їх впливу на населення.

2. У рецензованій роботі Ястребцев В.Д. реалізував методи математичного аналізу показників захворюваності, побудови картограм розподілу країн за значеннями показників. У роботі запропоновано інформаційну технологію моніторингу просторово-часового розподілу значень показників, що базується на даних часових рядів і дозволяє будувати графіки зміни показників та картограми країн світу з різними станами захворюваності та вакцинації.

3. Наукова новизна роботи полягає у створенні інформаційної технології та системи нечіткого логічного виводу для моніторингу захворюванності на COVID-19.

4. Практична значимість результатів кваліфікаційної роботи Ястребцева В.Д. полягає в автоматизації етапів інформаційної технології засобами запропонованої інформаційної системи з використанням платформи Node.js. За даними 2020-2021 років автоматично зібрані показники кількості випадків захворювання, смертності та щеплень, здійснено їх інтелектуальний аналіз та візуалізацію результатів.

5. Робота цілком відповідає вимогам, що пред'являються до кваліфікаційних робіт рівня магістра.

Недолік: недостатньо обґрунтовані переваги запропонованої інформаційної системи та технології, відсутні детальні порівняння з існуючими аналогами.

Незважаючи на зазначений недолік, кваліфікаційна робота в цілому може бути відзначена оцінкою "_____".

Рецензент,
д.т.н., професор М.О. Алексеєв