Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи прогнозування розвитку пандемій на основі штучних нейронних мереж з урахуванням зовнішніх факторів
(тема)

Виконав:

студент II курсу, групи СПм-20-1

Мотькін М.А.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Іващенко Г.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Мотькіну Максиму Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи прогнозування розвитку пандемій на основі штучних нейронних мереж з урахуванням зовнішніх факторів

затверджена наказом по університету від “ 5 ” листопада 2021 р. № 1657 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2021 р.

3. Вхідні дані до роботи
1) документація мови програмування Python
2) перелік використаний програмних засобів: ОС Windows 10
3) середовище розробки Jupyter Notebook
4) набори даних про розвиток пандемій

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз предметної області
2) огляд існуючих досліджень
3) використовувані методи прогнозування часових рядів
4) програмна реалізація
5) аналіз результатів дослідження
6) висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 17 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	09.11.21-12.11.21	
2	Вибір та обґрунтування методики дослідження	13.11.21-18.11.21	
3	Вибір інструментальних засобів	19.11.21-22.11.21	
4	Розробка алгоритмічного забезпечення	23.11.21-29.11.21	
5	Проведення експериментів	30.11.21-03.12.21	
6	Оформлення матеріалів атестаційної роботи	04.12.21-07.12.21	
7	Подання атестаційної роботи керівникові та	08.12.21-09.12.21	
	її попередній захист		
8	Подання атестаційної роботи на рецензування	10.12.21-11.12.21	

Дата видачі завдання 08 листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Іващенко Г.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 86 с., 35 рис., 9 табл., 3 дод., 30 джерел.

ШТУЧНА НЕЙРОННА МЕРЕЖА, ЧАСОВІ РЯДИ, COVID-19, LSTM, RNN, TCN, CNN, ПРОГНОЗУВАННЯ, MSE, TENSORFLOW, KERAS, PYTHON.

Метою кваліфікаційної є дослідження методів прогнозування пандемій з урахуванням зовнішніх чинників.

У ході виконання кваліфікаційної роботи було проведено аналіз існуючих методів прогнозування пандемій, їх переваг та недоліків. Було вивчено процес очищення даних часових рядів, побудову та дослідження нейронних мереж для прогнозування часових рядів, а саме моделей нейронних мереж LSTM, RNN, TCN та CNN на основі Keras та Tensorflow.

Дослідження проводилися із застосуванням актуальних даних щодо розвитку пандемії COVID-19.

Розроблений програмний продукт дозволяє виконувати порівняння та аналіз 4 типів штучних нейронних мереж, а також виконувати за їх допомогою прогнозування розвитку пандемії COVID-19.

ABSTRACT

Master’s thesis: 86 pages, 35 figures, 9 tables, 3 appendices, 30 sources.

ARTIFICIAL NEURAL NETWORK, TIME SERIES, COVID-19, LSTM, RNN, TCN, CNN, FORECASTING, MSE, TENSORFLOW, KERAS, PYTHON.

The major goal of this qualification work is research methods of pandemic forecasting, taking into account external factors. In the process of the qualification work performed an analysis of existing methods of time series forecasting, their advantages and disadvantages. Explored the process of time series data cleaning, the creation and evaluation of neural networks for time series prediction, especially neural network models LSTM, RNN, TCN and CNN based on Keras and Tensorflow.

The research was performed using actual COVID-19 pandemic progression data.

The developed software allows to compare and analyze 4 types of artificial neural networks, and also perform forecasting of COVID-19 pandemic progression.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШНМ – штучна нейронна мережа

ARMA – модель авторегресії ковзного середнього (англ., Autoregressive Moving Average)

CNN – згорткова нейронна мережа (англ., Convolutional Neural Network)

LSTM – нейронна мережа довгої короткострокової пам’яті (англ., Long Short-Term Memory)

MLP – багатошаровий персептрон мережа (англ., Multilayer Perceptron)

MSE – середньоквадратична помилка мережа (англ., Mean Squared Error)

RNN – рекурентна нейронна мережа мережа (англ., Recurrent Neural Network)

TCN – часова згорткова мережа мережа (англ., Temporal Convolutional Network)

ВСТУП

На сьогоднішній день прогнозування розвитку пандемій є вкрай актуальним завданням, від результатів вирішення якого можуть залежати людські життя та економічний стан країни. Для вирішення подібного завдання доцільним є використання методів аналізу та прогнозування часових рядів.

Прогнозування кількості хворих може бути використано для визначення приблизних дат початку і завершення «хвиль» захворюваності. Отримана інформація дозволить своєчасно реагувати на сплески захворюваності, тим самим зменшуючи завдані ними збитки. Превентивне запровадження карантинних обмежень може зменшити навантаження на медичні установи, а своєчасне скасування обмежень – знизити економічні втрати.

Найчастіше прогнозування часових рядів виконується за допомогою статистичних методів. Основним недоліком статистичних методів є неможливість урахування зовнішніх факторів, які можуть мати значний вплив на кінцевий результат. Для прогнозування часових рядів з урахуванням зовнішніх факторів найкращим інструментом є штучні нейронні мережі, через їх здатність враховувати численні параметри для отримання точних прогнозів, а їхнє налаштування та навчання може бути автоматизовано, що полегшує адаптацію моделей до нових даних.

У роботі прогнозування часових рядів розглянуто в контексті прогнозування числа захворілих під час пандемії COVID-19 в Україні. В якості навчального матеріалу для нейронних мереж використані дані, зібрані Всесвітньою Організацією Охорони Здоров’я.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність завдання прогнозування пандемій

У сучасному світі дуже актуальною є проблема обробки даних, що описують динаміку розвитку пандемії. Своєчасне вжиття медичних та організаційних заходів з урахуванням вірогідного початку чи закінчення хвиль захворювання у найближчому майбутньому, дозволить зменшити навантаження на медичні заклади, зменшити кількість інфікованих серед населення та знизити тиск на економіку. Для своєчасного застосування медичних та організаційних заходів необхідно застосування прогнозування на основі вже наявних даних про кількість інфікованих.

Прогнозування – це передбачення майбутнього виходячи з накопиченого досвіду та даних.

Найбільш популярними методами прогнозування є методи експертних оцінок, екстраполяції та моделювання.

В основі прогнозу за методом експертних оцінок лежить думка одного спеціаліста або групи фахівців, яка ґрунтується на професійному, практичному та науковому досвіді.

Метод екстраполяції – вивчення сформованих у минулому та сьогоденні стійких тенденцій розвитку процесів та явищ та перенесення їх на майбутнє. Методи екстраполяції слід застосовувати на початковій стадії прогнозування, щоб виявити тенденцію зміни показників.

Метод моделювання – конструювання моделі на підставі попереднього вивчення об’єкта та процесів, виділення його суттєвих ознак та характеристик. Прогнозування з використанням моделей включає їх розробку, експериментальний аналіз, зіставлення результатів попередніх прогнозних розрахунків з фактичними даними стану процесу або об’єкта, уточнення і коригування моделі [1].

Дані стосовно динаміки числа інфікованих можуть буди представлені у вигляді часових рядів, які можуть бути описані математичними моделями, для обробки яких доцільним є використання технічних засобів.

1.2 Часові ряди

Часовий ряд – упорядкована послідовність значень якого-небудь показника, отримана шляхом регулярного вимірювання протягом деякого періоду часу. Основна характеристика, яка відрізняє часовий ряд від простої вибірки даних – зазначений час вимірювання або порядковий номер вимірювання [2].

Часові ряди використовуються для аналітики і прогнозування, коли важливо визначити, що буде відбуватися з показниками в найближчу годину/день/місяць/рік: наприклад, скільки користувачів завантажать за день мобільний додаток. Показники для складання часових рядів можуть бути не тільки технічними, але й економічними, соціальними і природними.

1.2.1 Види часових рядів

Є багато класифікацій часових рядів. Наприклад, часові ряди можна поділити на детерміновані і недетерміновані (рисунок 1.1).

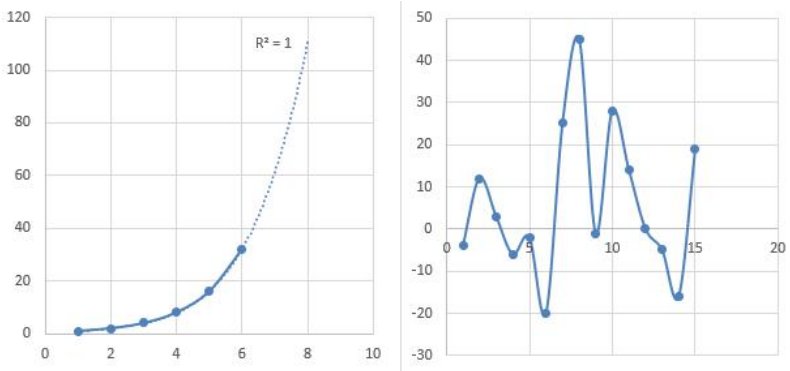

Рисунок 1.1 – Приклад детермінованого і недетермінованого часових рядів

Детермінований часовий ряд – ряд, в якому немає випадкових аспектів або показників, тому він може бути виражений формулою. Це означає, що його можна проаналізувати і отримати інформацію про те, як показники поводилися в минулому, і точно прогнозувати їх поведінку в майбутньому.

Недетермінований часовий ряд має випадковий аспект і може бути виражений лише поліномом ступені, де – число значень часового ряду.

Часові ряди також можна розділити за наявністю або відсутністю тенденцій і сезонних ефектів на стаціонарні та нестаціонарні (рисунок 1.2).

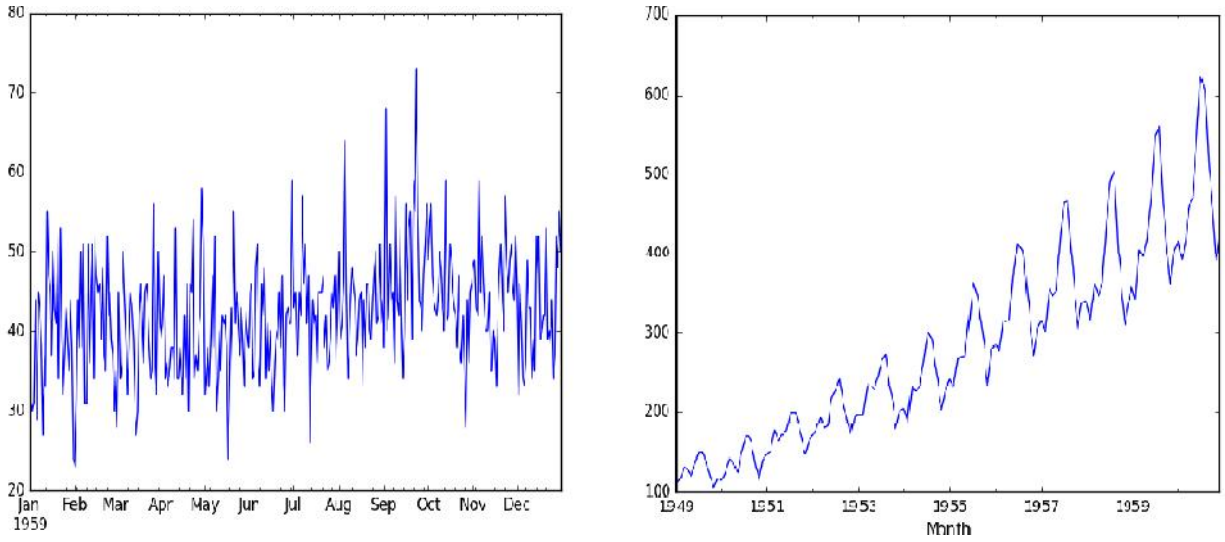

Рисунок 1.2 – Приклад стаціонарного і нестаціонарного часового ряду

Стаціонарний часовий ряд – ряд, елементи якого є випадковими величинами з постійним математичним очікуванням та постійною дисперсією. Стаціонарні часові ряди змінюють своє значення відносно деякого постійного значення і не мають чітко вираженої тенденції.

Більшість натуральних процесів за своєю суттю є нестаціонарними, а оскільки прогнозування на основі нестаціонарних рядів ускладнено, їх часто призводять до стаціонарного виду.

Горизонт прогнозування – часовий інтервал, в межах якого прогноз виконується із заданою точністю.

Сезонність часового ряду – це наявність змін, що відбуваються через певні регулярні інтервали; це компонент часових рядів, який зазнає регулярних і передбачуваних змін через фіксовані проміжки часу [2].

1.2.2 Прогнозування часових рядів

Прогнозування часових рядів полягає у побудові моделі для прогнозування майбутніх значень часового ряду, ґрунтуючись на вже відомих даних та прогнозування майбутніх даних до того, як вони будуть виміряні.

При вирішенні завдання прогнозування часових рядів, в якості вхідних даних можна використовувати не тільки цільовий параметр, а й зовнішні параметри, які можуть прямо або опосередковано впливати на отримані результати.

У якості прикладу можна розглянути прогнозування температури повітря, яке можна виконати, використовуючи лише дані про температуру. Для підвищення точності прогнозу можна задіяти зовнішні фактори, такі як: тиск повітря, вологість і т.д.

Зовнішні фактори можуть бути як дискретними, так і категоріальними. Якщо більшість статистичних моделей прогнозування дозволяють враховувати тільки дискретні фактори, то моделі обчислювального інтелекту дозволяють враховувати і категоріальні чинники [3].

1.2.3 Статистичні методи прогнозування часових рядів

Розглянуто наступні статистичні методи прогнозування:

- наївний метод;
- метод ковзних середніх;
- метод експоненційного згладжування;
- моделі ARMA та ARIMA.

Найпростішим методом прогнозування є наївний метод. Наївний метод заснований на припущенні, що майбутнє найкраще характеризується останніми змінами. Цей метод ґрунтується на припущенні про те, що прогнозовані показники в майбутньому періоді дорівнюють показникам попереднього періоду і не враховує сезонні коливання або тренди (рисунок 1.3).

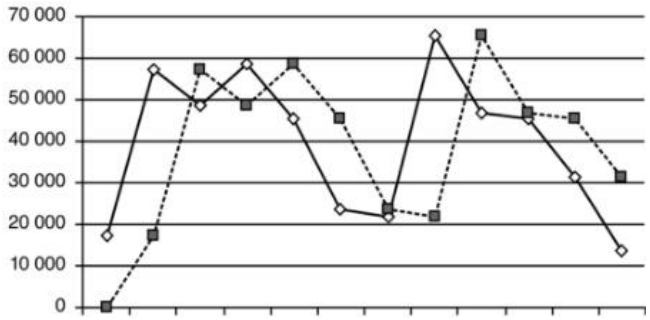

Рисунок 1.3 – Приклад наївного прогнозу часового ряду

Метод ковзних середніх полягає в усередненні значень в межах вікна, що ковзає по часовому ряду. Ширина ковзного вікна визначає кількість минулих періодів, що враховуються в прогнозуванні (рисунок 1.4).

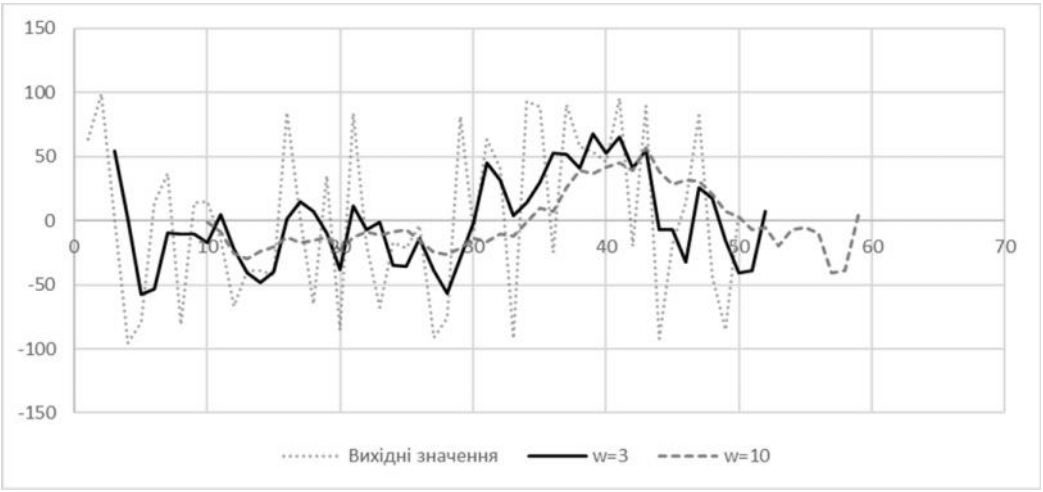

Рисунок 1.4 – Приклад прогнозування часового ряду методом ковзних середніх при ширині вікна 3 та 10

Експоненціальне згладжування – один з найстаріших і досліджених методів прогнозування часових рядів. Воно найбільш ефективно, коли значення часового ряду слідують за поступовим трендом і відображають сезонну поведінку, в якій значення слідують за періодичною циклічною схемою протягом заданої кількості часових кроків (рисунок 1.5). Даний метод має досить високу точність для короткострокових прогнозів, яка поступово падає зі збільшенням горизонту прогнозування.

На відміну від методу ковзних середніх, згідно з яким всі дані мають однакову вагу, в методі експоненціального згладжування найбільший коефіцієнт приписується самому останньому спостереженню.

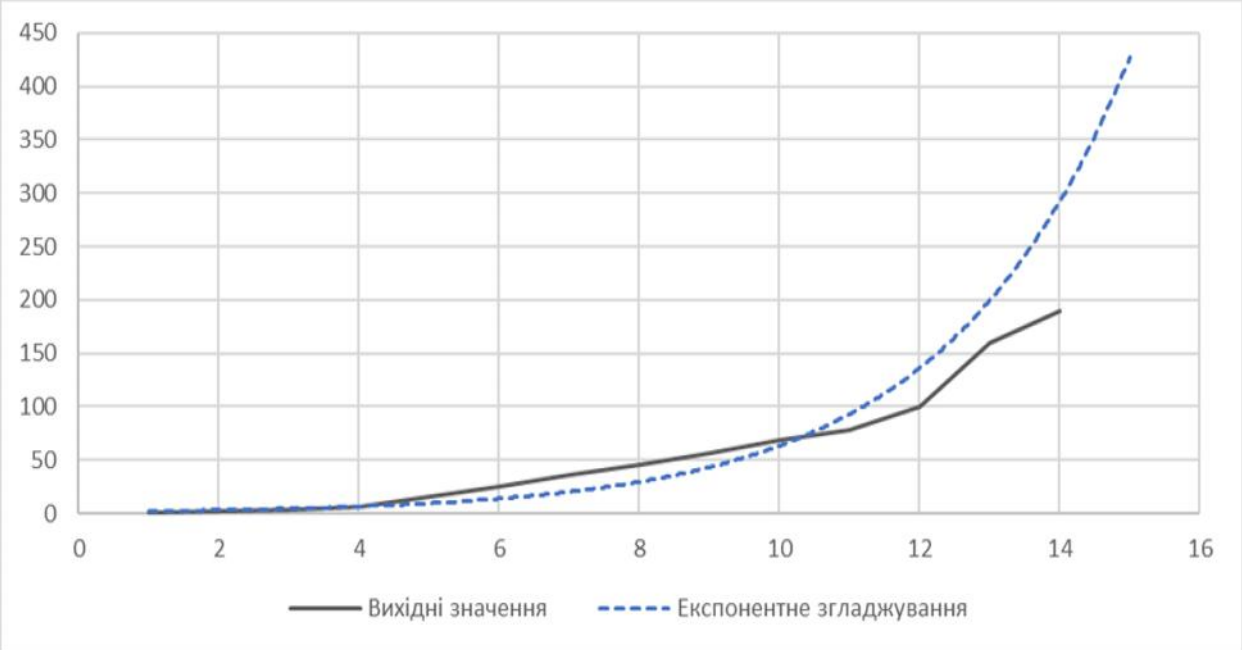

Рисунок 1.5 – Приклад прогнозування часового ряду методом експонентного згладжування

ARMA (autoregressive moving-average model) – модель авторегресії і ковзного середнього, яка базується на припущенні про те, що поточні значення досліджуваного часового ряду залежить тільки від лінійної комбінації попередніх значень часового ряду і білого шуму.

Мінусом ARMA-моделі є те, що вона може бути застосована тільки для стаціонарних часових рядів, тобто якщо в досліджуваному ряді є тренд, то ARMA-модель в чистому вигляді використовувати не можна. Якщо ж часовий ряд є нестаціонарним, то для нього можна застосувати модель ARIMA.

ARIMA (autoregressive integrated moving average) – модифікація моделі ARMA, яка використовує різниці значень спостережень (наприклад, віднімання спостереження на кроці n з спостереження на кроці n+1) для того, щоб зробити часовий ряд стаціонарним [3].

1.3 Аналіз існуючих досліджень

1.3.1 Використання моделі BiLSTM для прогнозування поширення пандемії COVID-19

Основною метою дослідження було вивчення розвитку та поширення пандемії COVID-19 в Саудівській Аравії та створення моделі штучної нейронної мережі, що базується на архітектурі BiLSTM. Bidirectional LSTM – це двонаправлена версія моделі LSTM; складається з двох LSTM: одна приймає дані у прямому напрямку, а інша – у зворотному напрямку.

Як вихідний набір даних використані такі параметри: підтверджені випадки захворювання, випадки смерті та випадки одужання.

Запропонована модель була оптимізована за допомогою гіперпараметрів під розвиток пандемії в Саудівській Аравії та, ймовірно, не може бути застосована до інших країн без додаткового налаштування. Отримані результати прогнозування на основі моделі BiLSTM були порівняні з іншими підходами прогнозування [4].

Відповідно до результатів дослідження, отримана модель BiLSTM показала значну ефективність щодо статистичних методів прогнозування. Однак, порівняно з іншими поширеними методами машинного навчання,

модель показала лише аналогічну точність, яка не дозволяє зазначити, що модель BiLSTM має значну перевагу (рисунок 1.6).

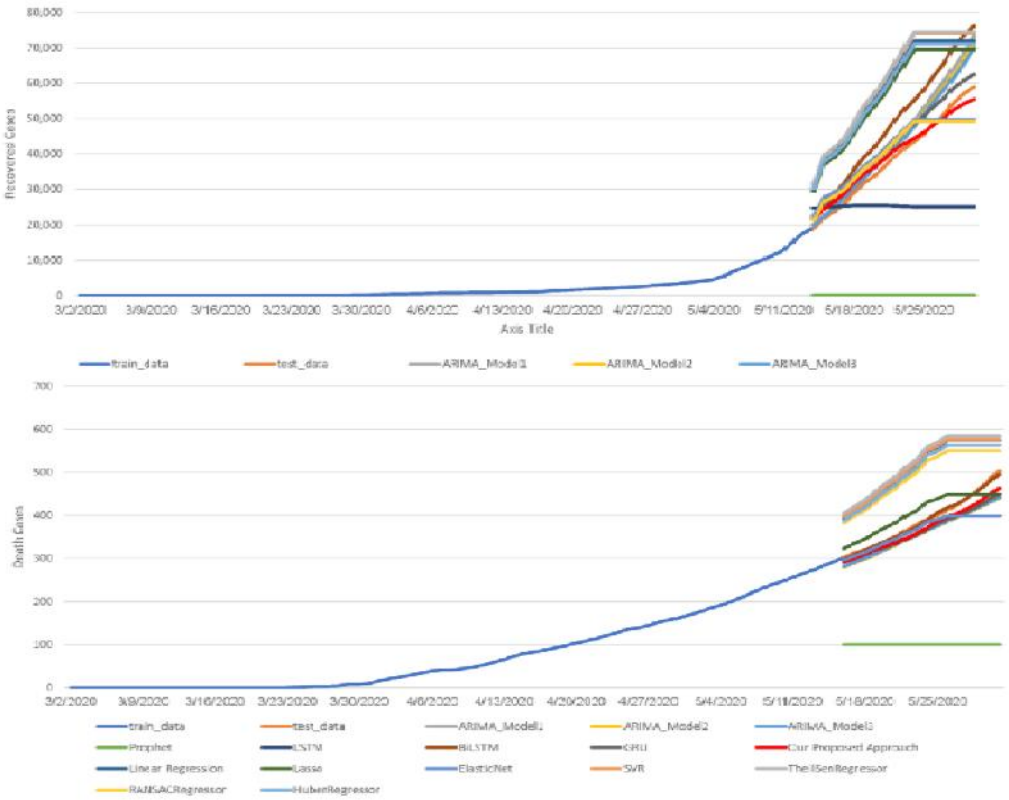

Рисунок 1.6 – Порівняння точності прогнозування випадків одужання та випадків смерті за допомогою досліджуваних методів

1.3.2 Модель глибокого навчання для прогнозування спалаху COVID-19 у Єгипті

Розглянуте дослідження спрямоване на порівняння ефективності прогнозування спалаху COVID-19 у Єгипті за допомогою штучних нейронних мереж на архітектурі LSTM, MLP та CNN.

Використаний набір даних містив записи про підтверджені випадки зараження, смерті та одужання у проміжку від 14 лютого 2020 року до 30 червня 2021 року (рисунок 1.7). Тобто набір даних охоплює понад 503 дні. 90% даних було використано для навчальної вибірки та 10% для тестової.

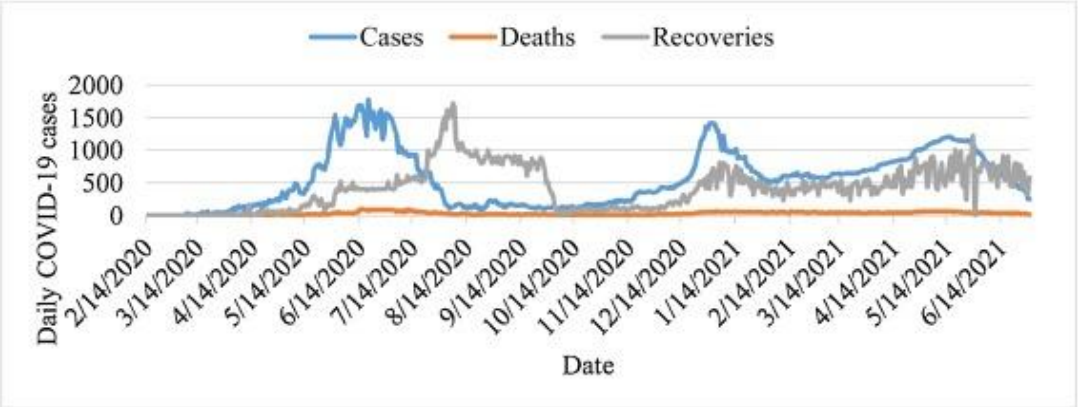

Рисунок 1.7 – Візуалізація даних про випадки захворювання, одужання та смерті від COVID-19 в Єгипті з 14 лютого 2020 року по 30 червня 2021 року.

Для порівняння моделей LSTM, CNN і MLP було виконано прогнозування кількості хворих на тиждень (рисунок 1.8) та на місяць (рисунок 1.9) уперед [5].

Згідно з отриманими результатами модель на основі архітектури LSTM показала найкращі результати і мала найбільшу точність під час прогнозування на тиждень і на місяць вперед. Найгірші результати показала модель CNN, яка під час прогнозування на місяць уперед почала періодично генерувати аномальні значення.

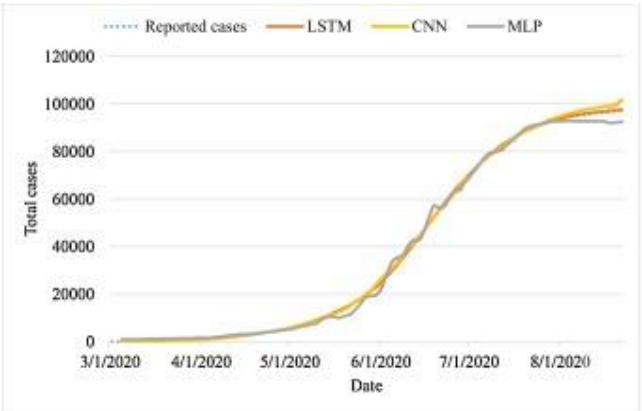

Рисунок 1.8 – Загальна кількість підтверджених та прогнозованих випадків на тиждень наперед

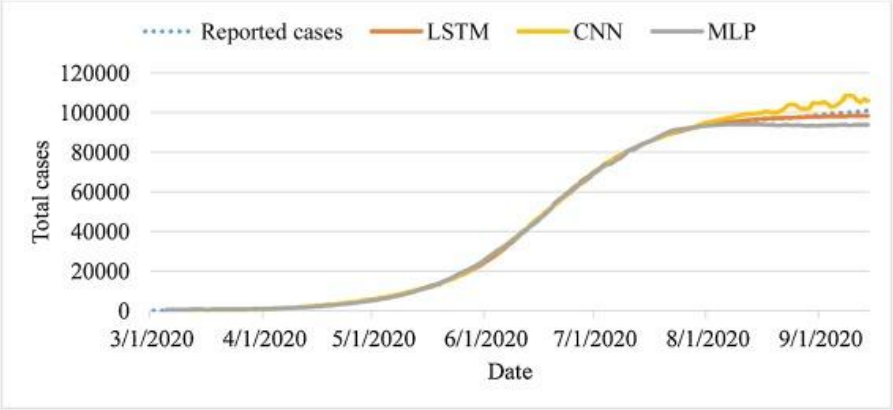

Рисунок 1.9 – Загальна кількість підтверджених та прогнозованих випадків на місяць наперед

1.4 Постановка задачі

Метою роботи є дослідження методів прогнозування динаміки розвитку пандемій з урахуванням зовнішніх факторів за допомогою моделей машинного навчання на основі штучних нейронних мереж.

Для досягнення мети необхідно виконання відповідних завдань, а саме:

- підготувати та проаналізувати доступні часові ряди з даними о кількості осіб, що захворіли, кількості тих, хто помер, одужав чи пройшов курс вакцинації;
- створити моделі штучних нейронних мереж на основі CNN, RNN, TCN та LSTM;
- навчити моделі на наявних наборах даних;
- порівняти та проаналізувати отримані результати.

2 ВИКОРИСТАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

2.1 Штучні нейронні мережі

Штучна нейронна мережа – це частина обчислювальної системи, призначена для моделювання того, як людський мозок аналізує і обробляє інформацію. Це основа штучного інтелекту, яка вирішує проблеми, які можуть виявитися неможливими або важкими за людськими або статистичними стандартами. Штучні нейронні мережі містять нейрони, що з’єднані між собою у мережу (рисунок 2.1). Людський мозок складається з сотень мільярдів клітин, які називаються нейронами. Кожен нейрон складається з тіла клітини, яке відповідає за обробку інформації, передаючи інформацію в напрямку входів і виходів.

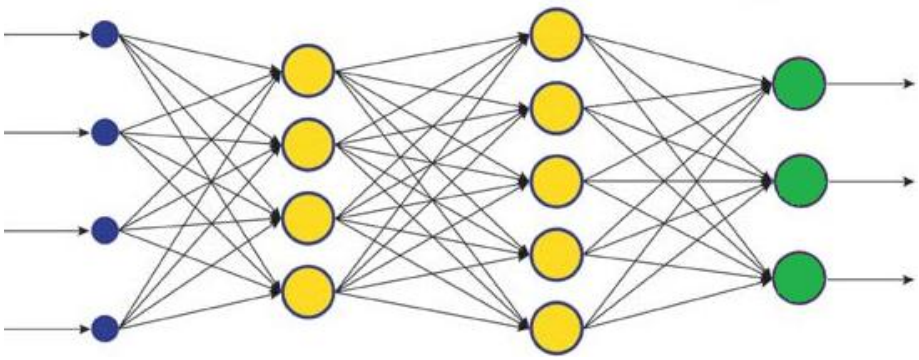

Рисунок 2.1 – Схематичне зображення штучної нейронної мережі

Нейронна мережа має сотні або тисячі штучних нейронів (рисунок 2.2), які називаються блоками обробки, і які пов’язані між собою вузлами. Ці блоки обробки складаються з блоків введення і виведення. Блоки введення отримують різні набори вхідної інформації, а нейронна мережа намагається визначити ваги для зв’язків, щоб отримати результати схожі на перевірочні дані.

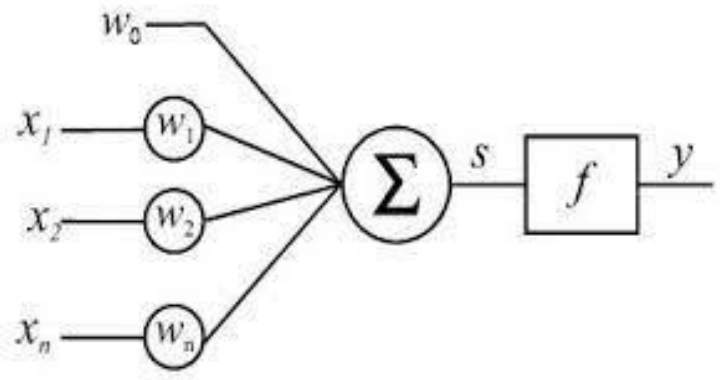

Рисунок 2.2 – Схематичне зображення штучного нейрона

На рисунку 2.2 зображений штучний нейрон, де $x_{1..n}$ є вхідними сигналами, w_0 – модифікатор зсуву, Σ – суматор вхідних сигналів, y – результуючий сигнал, а f – функція активації.

2.1.1 Перенавчання нейронних мереж

Однією з основних проблем нейронних є перенавчання. Воно полягає в надмірній адаптації нейронної мережі до наявного набору даних замість того, щоб знаходити в ньому закономірності. Це призводить до того, що нейронна мережа втрачає гнучкість і запам’ятовує відповіді для певних вхідних даних.

Для боротьби з проблемою перенавчання штучних нейронних мереж було розроблено кілька методів, серед яких набув найбільшої поширеності «Метод проріджування» або Dropout. Головною ідеєю цього методу є навчання кількох нейронних мереж з наступним усередненням їх ваг.

Навчити кілька мереж недостатньо, тому що вони будуть демонструвати схожі результати. Для приведення цього методу в дію потрібно відключати частину нейронів нейронних мереж (рисунок 2.3).

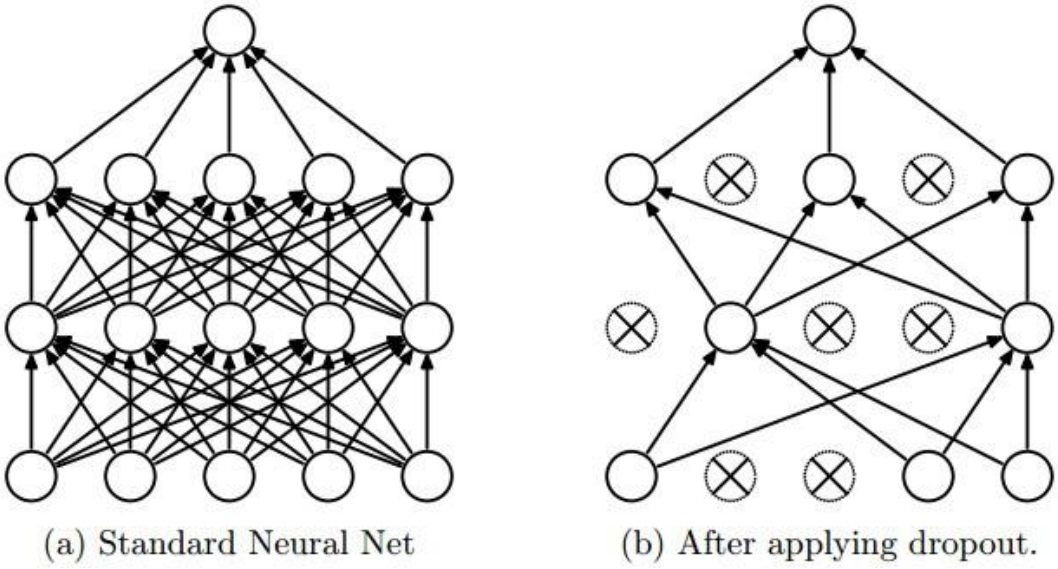

Рисунок 2.3 – Графічне представлення нейронної мережі до і після застосування методу Dropout

Відключення нейрона означає, що при будь-яких вхідних сигналах він буде повертати 0. Так як виключені нейрони не беруть участі в процесі навчання і зворотному поширенні помилки, їх відключення є еквівалентом навчанню нової нейронної мережі [20].

2.1.2 Інтерпретація результатів роботи штучних нейронних мереж

Проблема чорного ящика не є критичною, проте створює певні незручності під час дослідження вже навчених мереж. Вона полягає в тому, що з навченої нейронної мережі не можна витягти ніяку корисну інформацію крім тієї, яка передається на вихід. З цього виходить наступна проблема – вкрай складно зрозуміти як і на основі чого нейронна мережа приймає рішення.

Для вирішення цієї проблеми розробляються алгоритми вилучення правил, які повинні отримувати інформацію з нейронних мереж у вигляді дерев рішень, математичних або логічних виразів.

2.2 Згорткові нейронні мережі (CNN)

Згорткова нейронна мережа – це спеціальна архітектура нейронних мереж прямого поширення, запропонована Яном Лекуном, націлена на ефективне розпізнавання образів. Під прямим поширенням розуміється те, що нейрони в цій мережі розбиті на групи, звані шарами. Ці мережі використовують для розпізнавання мови, обробки аудіосигналів, обробки часових рядів, для аналізу змісту текстів (рисунок 2.4).

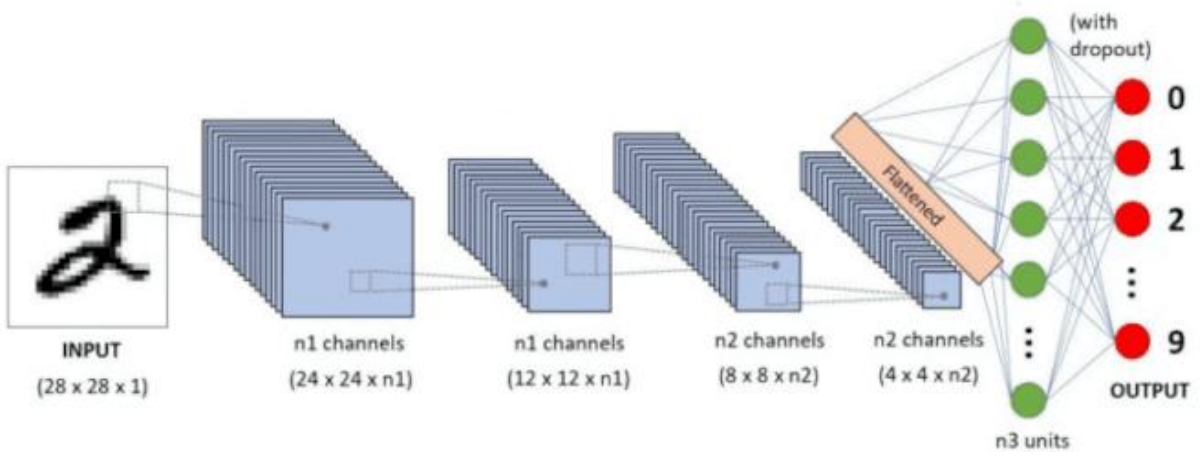

Рисунок 2.4 – Приклад структури згорткової нейронної мережі, на прикладі мережі, що розпізнає рукописні цифри

Коли така багатошарова нейронна мережа застосовується до даних, то активація шарів підраховується послідовно: спочатку значення активації першого шару, потім значення активації другого шару, і так до останнього шару. Активація останнього шару і служить виходами нейронної мережі, а активація кожного наступного шару залежить від активації попереднього шару. Завдяки тому, що активації у межах одного шару одна від одної не залежать, активації у межах одного шару можуть обчислюватися паралельно і це призводить до того, що такі нейронні мережі можна дуже зручно і ефективно обраховувати на сучасних процесорах, в тому числі на графічних співпроцесорах.

Особливість згорткової нейронної мережі полягає в тому, що в ній нейрони перших рівнів впорядковані в особливу структуру, а саме: на перших шарах нейрони розбиті на карти певного розміру, і різні карти всередині одного шару відповідають нейронам різного типу, які реагують на різні особливості зображень. Обчислення активації шарів в згорткових мережах бувають двох основних типів. У першому типі обчислень активація нейронів наступного рівня обчислюється як лінійна комбінація активацій нейронів попереднього рівня, причому ваги цих лінійних активацій залежать тільки від взаємного положень нейронів, типів нейронів, але не залежать від положення даного нейрона всередині карти [17].

У другому типі обчислень активація нейронів на наступному рівні просто повторює активацію нейронів на попередньому рівні, але зображення стає меншого розміру за рахунок того, що значення поруч розташованих нейронів замінюється на їх максимальне або середнє значення за допомогою процедури згортки. Подібна додаткова структура робить згорткові нейронні мережі придатними для роботи з образами. Вона гарантує, що якщо два образи відрізняються малим зсувом, то нейромережа на виході отримає схожий результат [6].

2.1.1 Згортка

Згортка – операція перетворення набору даних за допомогою згорткового ядра. Ця операції полягає в переміщенні ядра по площині набору даних і виконанні підсумовування елементів, після чого ядро поміщає отриманий результат на наступний шар (рисунок 2.5). Тобто операція згортки призводить до перетворення даних у новий формат та зменшення їх розмірності.

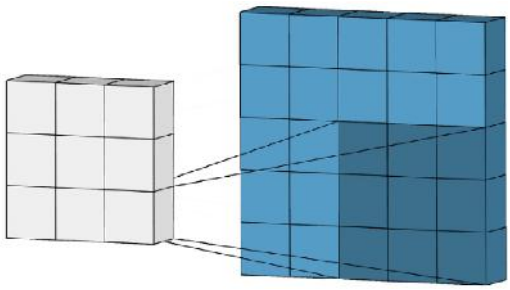

Рисунок 2.5 – Згортка набору даних розмірністю 5x5 за допомогою ядра 3x3

В наведеному прикладі є 5×5 = 25 вхідних і 3×3 = 9 вихідних характеристик. Якби розглядався стандартний повнозв’язний шар, вийшла б вагова матриця на 25 × 9 = 225 параметрів і відповідь цього шару формувався б за допомогою підсумовування добутків кожного вхідного сигналу на деякий ваговий коефіцієнт.

Згортки дозволяють здійснювати подібні операції тільки над дев’ятьма параметрами – з кожним вихідним сигналом, тим самим значно знижуючи обчислювальну складність мережі [6].

2.1.2 Страйд згортки

Страйд ядра – величина на яку зміщується ядро згортки під час переміщення по вихідній матриці (рисунок 2.6).

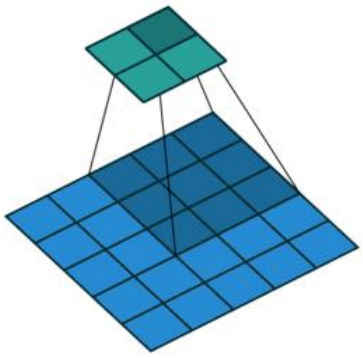

Рисунок 2.6 – Перетворення матриці 5x5 в матрицю 2x2 при страйді 2

Так наприклад, при страйді рівному 2, матриця 5x5 за допомогою ядра 3x3 буде перетворена в матрицю 2x2.

Таким чином, при страйді зі значенням 2 ядро зміщується на два елементи, зменшуючи розмір результуючої матриці в порівнянні з попереднім варіантом приблизно вдвічі [6].

Переваги:

- завдяки своїй структурі, згорткові нейронні мережі дозволяють значно знизити кількість ваг в порівнянні з повнозв’язною нейронною мережею;

- операції згортки нейронної мережі легко піддаються розпаралелюванню, що позитивно впливає на швидкість навчання і на швидкість роботи;

- через те, що для мережі більш важлива наявність ознаки, а не її місце в образі, вона ігнорує різні зрушення сканованого образу, що може бути корисно при прогнозуванні часових рядів.

Недоліки:

- склад штучної нейронної мережі (кількість її нейронних шарів, функція активації нейронів, розмірність згорток, розмірність pooling-шарів, черговість шарів і т.д.) необхідно підбирати емпірично до певного розміру і виду образу;

- складність навчання: нейромережі з хорошим показником помилки повинні тренуватися на потужних GPU довгий час.

2.3 Рекурентні нейронні мережі (RNN)

Рекурентні нейронні мережі – це найбільш складний вид нейронних мереж, в яких є зворотний зв’язок. При цьому під зворотним зв’язком мається на увазі зв’язок від логічно більш віддаленого елемента до менш віддаленого. Наявність зворотного зв’язку дозволяє запам’ятовувати і відтворювати цілі послідовності реакцій на один стимул.

З точки зору програмування в таких мережах з’являється аналог циклічного виконання, а з точки зору систем – така мережа еквівалентна кінцевому автомату.

На відміну від нейронних мереж з прямим зв’язком, RNN можуть використовувати свій внутрішній стан (пам’ять) для обробки послідовностей вхідних даних (рисунок 2.7) [19].

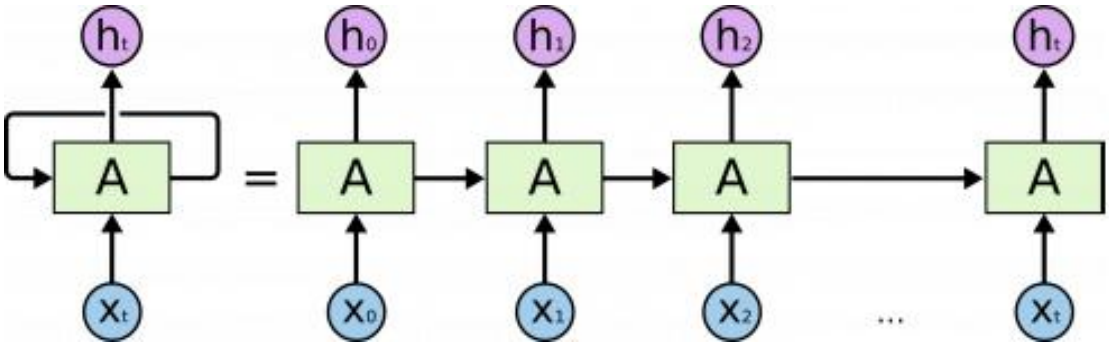

Рисунок 2.7 – Приклад розгортки рекурентної мережі у вигляді ланцюга нейронних мереж з прямим зв’язком

Будь-яку рекурентну мережу можна представити у вигляді ланцюга повторюваних модулів нейронної мережі, що передають значення від комірки до комірки (рисунок 2.8).

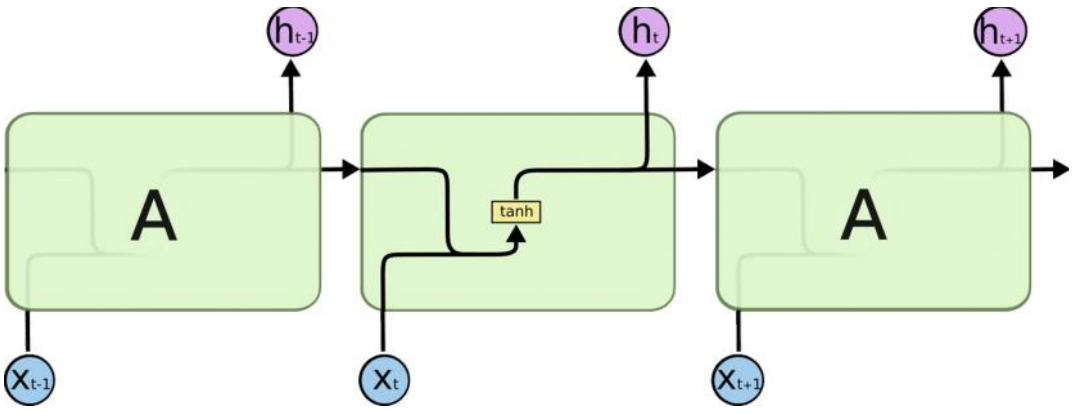

Рисунок 2.8 – Детальна схема розгортки рекурентної мережі з функцією активації tanh

У звичайній RNN структура комірки являє собою один шар з простою функцією активації, наприклад tanh (гіперболічний тангенс).

Головним недоліком рекурентних нейронних мереж є те, що мережа втрачає можливість пов’язувати наявну інформацію з вхідним набором при збільшенні різниці між ними. Ця проблема називається проблемою довготривалих залежностей і вирішується модифікацією RNN нейронних мереж – LSTM [7].

2.4 Нейронні мережі з довгою короткостроковою пам’яттю (LSTM)

Довга короткострокова пам’ять (Long short-term memory, LSTM) – особливий різновид архітектури рекурентних нейронних мереж, розроблена спеціально для того, щоб вирішити проблему довгострокових залежностей. Характерною особливістю LSTM мереж є зберігання інформації протягом тривалих періодів часу. Структура LSTM, також як і структура RNN, нагадує ланцюг, але складніше RNN (рисунок 2.9). Замість одного шару нейронної мережі вони містять чотири, які взаємодіють особливим чином.

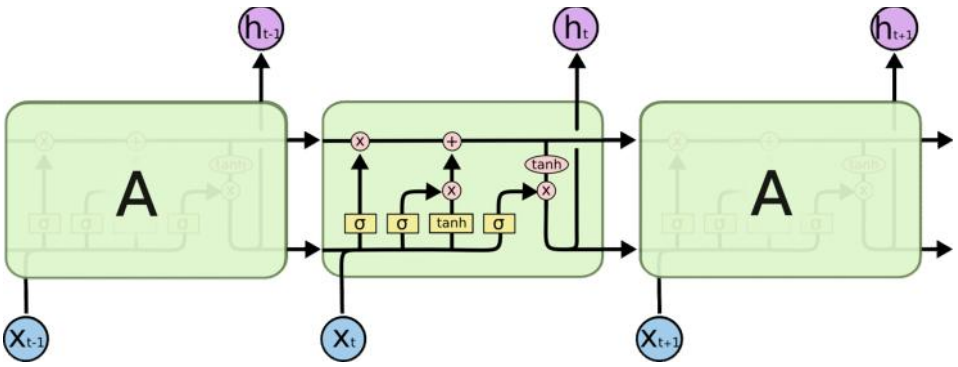

Рисунок 2.9 – Структура розгортки LSTM мережі

Ключовим компонентом LSTM є стан комірки, який представлено у вигляді горизонтальної лінії, що проходить по верхній частині схеми (рисунок 2.10).

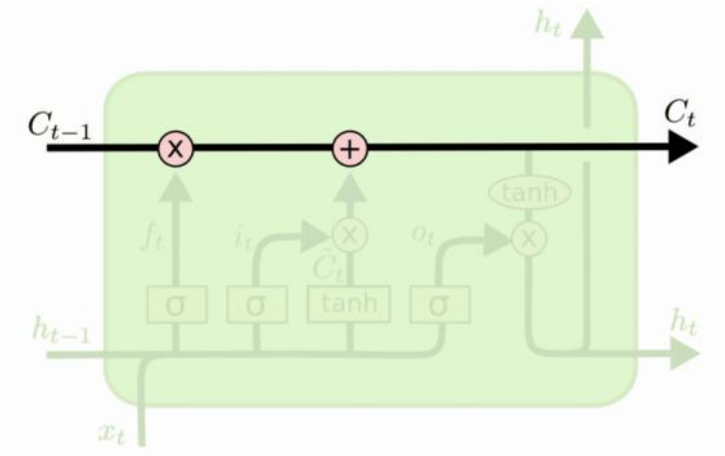

Рисунок 2.10 – Стан комірки на схемі розгортки LSTM мережі

LSTM мережа може видаляти інформацію зі стану комірки і цей процес регулюється фільтрами. Фільтри дозволяють пропускати інформацію на підставі деяких умов. Вони складаються з шару сигмоїдальної нейронної мережі і операції множення. Сигмоїдальний шар повертає числа від нуля до одиниці, які позначають, яку частку кожного блоку інформації слід пропустити далі по мережі. Нуль в даному випадку означає «не пропускати нічого», одиниця – «пропустити все». У LSTM три таких фільтра і вони призначені для захисту і контролю стану комірки [7].

Перший етап в LSTM – визначення, яку інформацію можна виключити зі стану комірки (рисунок 2.11).

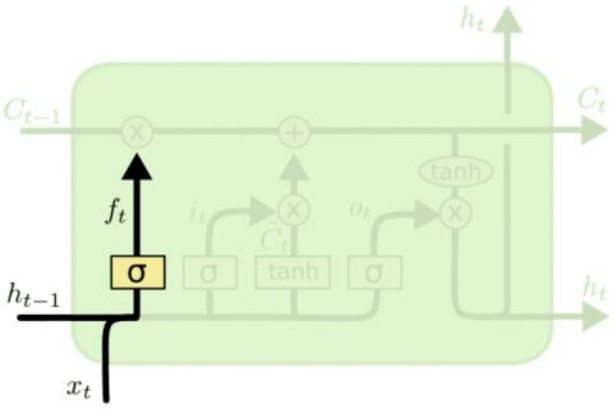

Рисунок 2.11 – Схематичне зображення першого етапу LSTM

Визначення, яку інформацію можна виключити зі стану комірки приймає сигмоїдальний шар, званий «шаром фільтра забування» (forget gate layer). Шар використовує вихідні дані з попередньої ітерації і на вхідні дані поточної інерції, після чого повертає число від 0 до 1, де 1 означає «повністю зберегти», а 0 – «повністю ігнорувати».

Наступний етап – вирішити, яка нова інформація буде зберігатися в стані комірки. Цей етап складається з двох частин. Спочатку сигмоїдальний шар під назвою «шар вхідного фільтру» (input layer gate) визначає, які значення слід оновити, а потім tanh-шар будує вектор нових значень, які можна додати в стан комірки (рисунок 2.12).

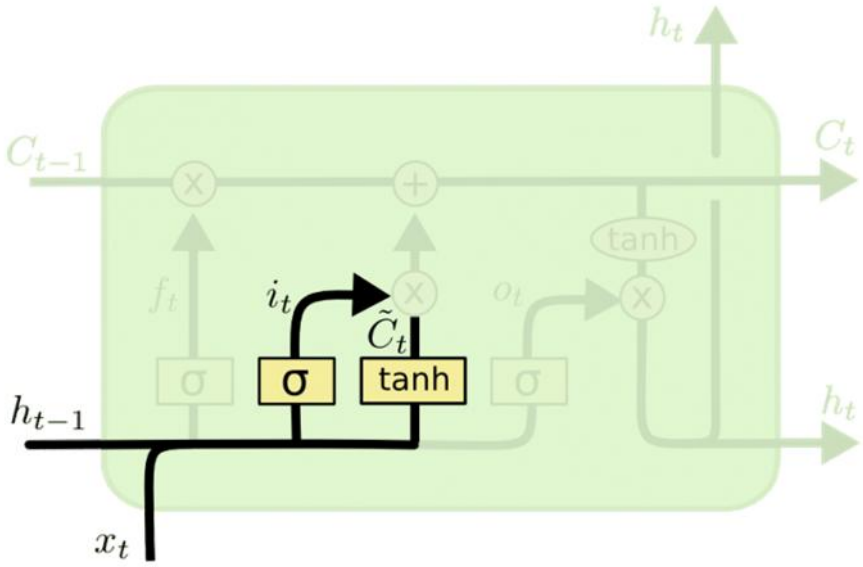

Рисунок 2.12 – Схематичне зображення другого етапу LSTM

Третій етап – заміна стану (рисунок 2.13). На цьому етапі виконується множення значення старого стану на значення отримані з «шару фільтра забування» і підсумовування з значеннями отриманими з «шару вхідного фільтра».

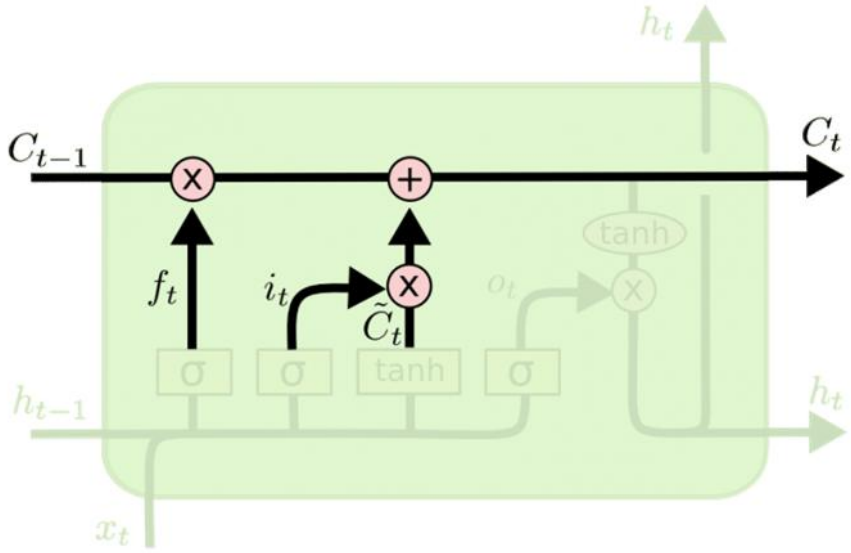

Рисунок 2.13 – Схематичне зображення третього етапу LSTM

На останньому етапі відбувається визначення, яку інформацію необхідно вивести (рисунок 2.14). Для цього значення стану комірки проходять через tanh шар, який повертає значення від -1 до 1. Далі ці значення перемножуються з значеннями сігмоідного шару, що дозволяє виводити тільки необхідну інформацію [7].

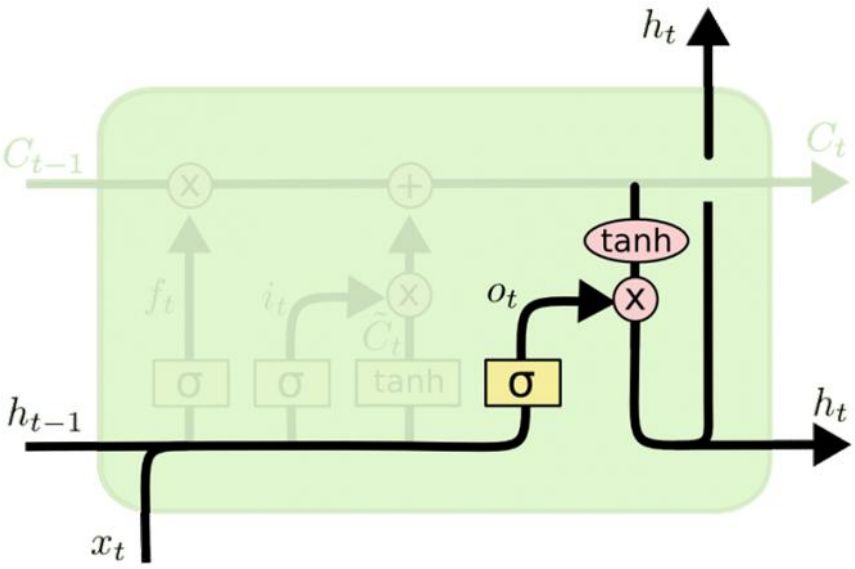

Рисунок 2.14 – Схематичне зображення останнього етапу LSTM мережі

2.5 Часові згорткові мережі (TCN)

Часові згорткові нейронні мережі – це модифікація згорткової нейронної мережі, що призначена для роботи з часовими рядами. У основі TCN лежить операція причинно-наслідкової згортки, тобто. вихідне значення залежить тільки від значень, які розташовані раніше у вхідній послідовності (рисунок 2.15).

Основними перевагами даної архітектури нейронних мереж є:

- довготриваліша пам’ять порівняно з рекурентними архітектурами нейронних мереж, з тією ж кількістю вузлів;
- точніші результати порівняно з архітектурами LSTM/GRU;
- висока продуктивність за рахунок можливості паралельної обробки згорткових шарів.

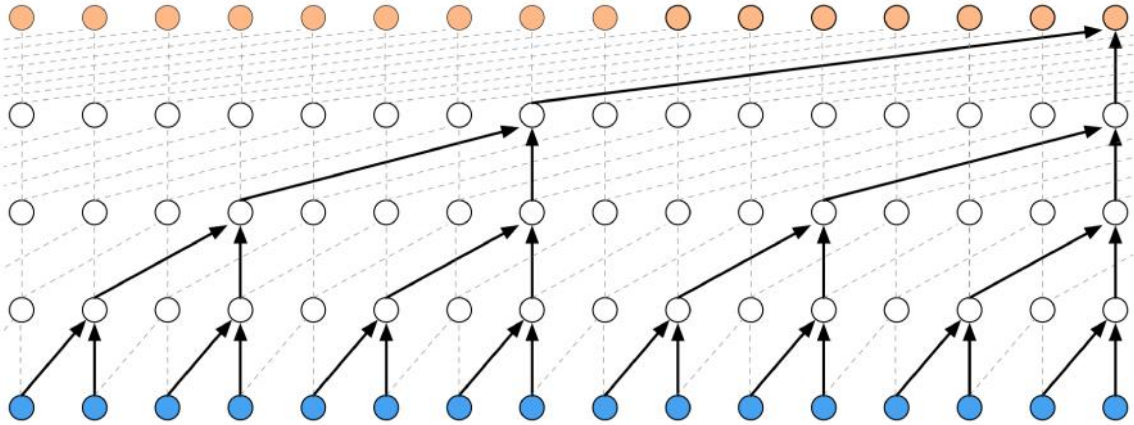

Рисунок 2.15 – Приклад згортки значень послідовності за допомогою TCN

До мінусів часових згорткових нейронних мереж можна віднести те, що довжина горизонту прогнозування часових згорткових мереж не може перевищувати довжину інтервалу, який використовуватиметься для прогнозування [24, 25].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис набору вихідних даних

Вибраний набір даних містить інформацію про розвиток пандемії COVID-19 у багатьох країнах. Дані представлені у вигляді більш ніж 135 тисяч записів, кожна з яких складається з 64 полів, проте для прогнозування кількості хворих будуть використані лише наступні поля:

- total_cases – загальна кількість підтверджених випадків захворювання на COVID-19;
- new_cases – кількість нових підтверджених випадків захворювання на COVID-19 у визначений день;
- total_tests – загальна кількість виконаних тестів;
- new_tests – кількість тестів виконаних у визначений день;
- stringency_index – комплексний показник, що ґрунтується на заходах протидії поширенню пандемії. Є чисельною оцінкою суворості введених карантинних обмежень для населення і набуває значення від 0 до 100 (100 = найсуворіші обмеження). Параметр розраховується в рамках проекту Oxford Coronavirus Government Response Tracker на основі наступних параметрів: закриття шкіл, закриття робочих місць, скасування публічних заходів, обмеження для проведення масових заходів, закриття громадського транспорту, вимоги не виходити із дому, громадські інформаційні кампанії, обмеження на внутрішні переміщення та контроль міжнародних поїздок [22, 23];
- people_fully_vaccinated – загальна кількість осіб, які отримали як мінімум дві дози будь-якої вакцини;
- population – чисельність населення обраної країни.

В якості додаткових параметрів використано інформацію про середньомісячну температуру та вологість в Україні [26, 27].

3.2 Очищення набору даних

3.2.1 Усунення пропущених значень

Першим етапом очищення даних часових рядів є усунення пропусків. Часові ряди зазвичай являють собою безперервний набір вимірювань через певні проміжки часу. Але апаратура, що використовується при вимірюваннях, або людина, що проводить вимірювання, може припуститися помилки, що призведе до виникнення ситуації, коли дані за певний проміжок часу відсутні.

При навчанні моделей нейронних мереж наявність пропусків може призводити до зниження точності отриманої моделі або неможливості її навчання. Для заповнення пропусків можуть застосовуватися наступні підходи.

Використання значення за попередній проміжок часу (наївний підхід). Це один з найпростіших варіантів, однак він має вкрай низьку ефективність в тих випадках, коли значення, що фіксуються, швидко ростуть.

Використання середнього значення двох сусідніх проміжків часу або лінійна інтерполяція. Підхід дозволяє отримувати мінімальні втрати точності, коли є чітка тенденція зміни даних. Але такий метод може призвести і до втрати точності, якщо пропущене значення буде піковим, а значення двох сусідніх інтервалів будуть сильно відрізнятися від пропущеного.

Використання значень за попередні інтервали при наявності явної сезонності. Такий підхід також може забезпечувати високу точність за умови, що різниця між значеннями різних сезонів невелика.

Використання поліноміальної інтерполяції. Метод полягає у знаходженні та використанні полінома найменшого можливого ступеня, який повинен проходити через точки набору даних. Поліноміальна інтерполяція дозволяє отримувати найкращі результати, але водночас є ресурсозатратним методом [13].

Для заповнення пропусків, спочатку був виконаний аналіз значень набору даних (лістинг 3.1) та оцінка стану кожного з стовпців (таблиця 3.1).

Лістинг 3.1 – Фрагмент коду для створення звіту про кількість пропущених значень у кожному з стовпців набору даних

```
for col in df.columns:
    null_records = df[col].isnull()
    pct_missing = np.mean(null_records)
    rounded_percent = round(pct_missing*100)
    print('{} - {}%'.format(col, rounded_percent))
```

Таблиця 3.1 – Звіт про кількість пропущених значень у кожному з стовпців набору даних

Стовпець	Відсоток пропущених значень
total_cases	0 %
total_tests	23 %
stringency_index	1 %
new_tests	34 %
new_cases	0 %
people_fully_vaccinated	60 %
population	0 %
date	0 %
temperature	6 %
humidity	6 %

Для визначення характеру пропусків значень у стовпцях набору даних було використано теплову карту (рисунок 3.1), створену за допомогою бібліотеки Seaborn. На тепловій карті відображено кількість пропущених значень у стовпцях, а також їхнє розташування в наборі даних. Пропущені значення пофарбовані у сірий колір, а наявні у чорний.

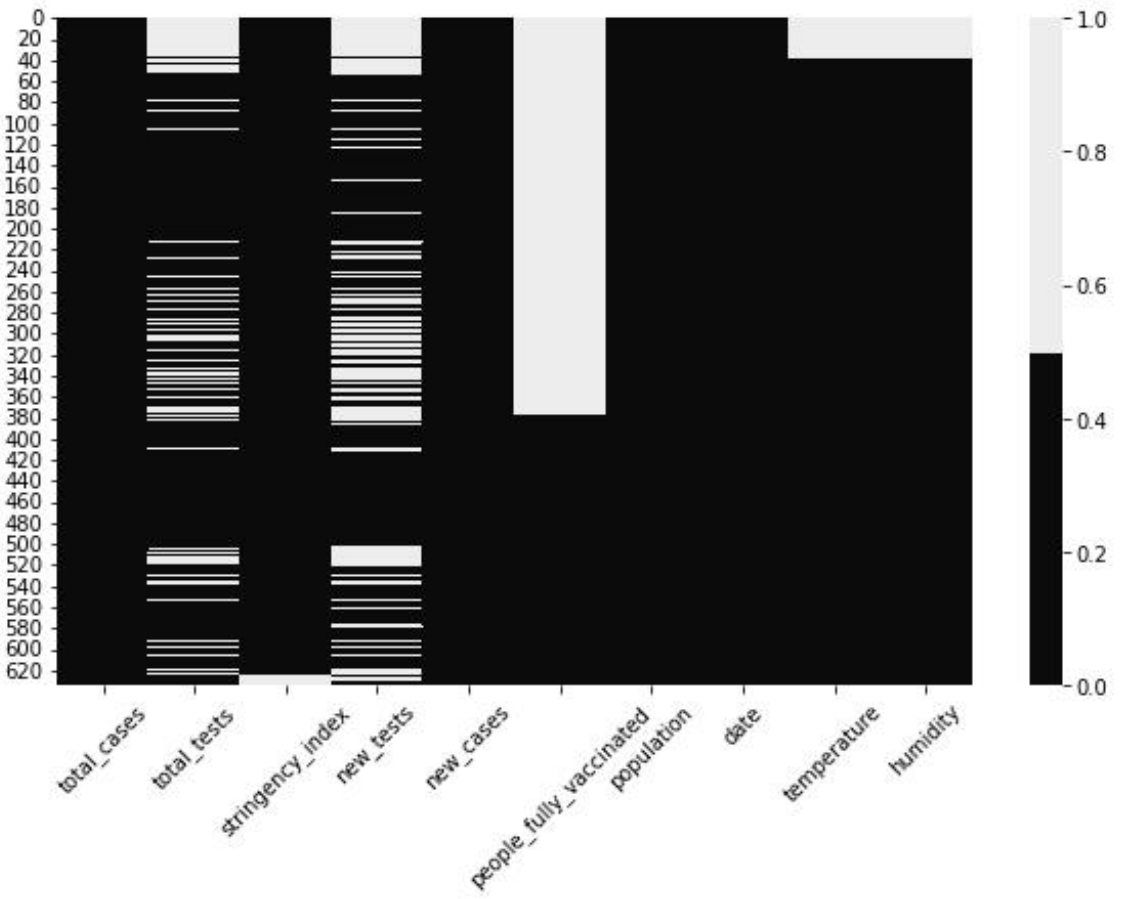

Рисунок 3.1 – Теплова карта пропущених значень у наборі даних

Після аналізу теплової карти можна виділити кілька проблемних стовпців: total_tests, people_fully_vaccinated, temperature, humidity.

Обробку було розпочато з стовпця total_tests, тому що від нього залежить стовпець new_tests. Для заповнення пропущених значень в стовпці total_tests спочатку потрібно було проаналізувати дані і перевірити їх на наявність викидів.

Викид – окреме значення часового ряду, яке значно відрізняється від закономірностей і трендів інших значень часового ряду. Наявність навіть невеликої кількості викидів може призводити до помилок прогнозування і значного зниження точності прогнозів.

Виявлення викидів – одна з найстаріших проблем аналізу даних. Причинами появи викидів можуть бути помилки вимірювань, помилки відбору вибірки, навмисне псування даних або некоректна фіксація

результатів аналізу вибірки, хибні припущення про розподіл даних або моделі, мала кількість спостережень і т.д.

Виявити викиди можливо за допомогою таких дій:

- квартільним методом;
- візуально.

Квартілі – значення, які ділять набір даних на чотири групи, що містять приблизно однакову кількість спостережень. Для знаходження викидів квартільним методом необхідно знайти 1 і 3 квартіль для часового ряду, а також медіанне значення. Максимальним відхиленням можна вважати значення, що дорівнює різниці 3 і 1 квартилю помноженої на 1,5. Будь-які значення, що виходять за рамки максимального відхилення, є викидами. Далі слід здійснити розрахунок відхилень кожного значення щодо медіанного значення. Значення, відхилення яких будуть більшими за максимальне відхилення, можна вважати викидами.

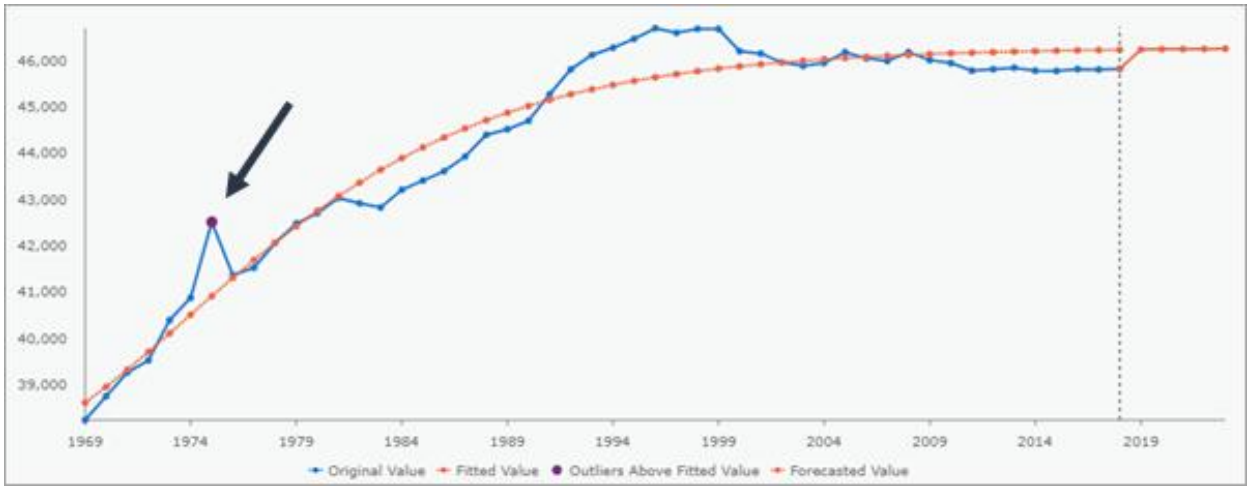

Рисунок 3.2 – Приклад візуального пошуку викидів

Для візуального знаходження викидів досить побудувати графік часового ряду (рисунок 3.2), після чого знайти всі значення, що надмірно відрізняються від трендів інших значень [8].

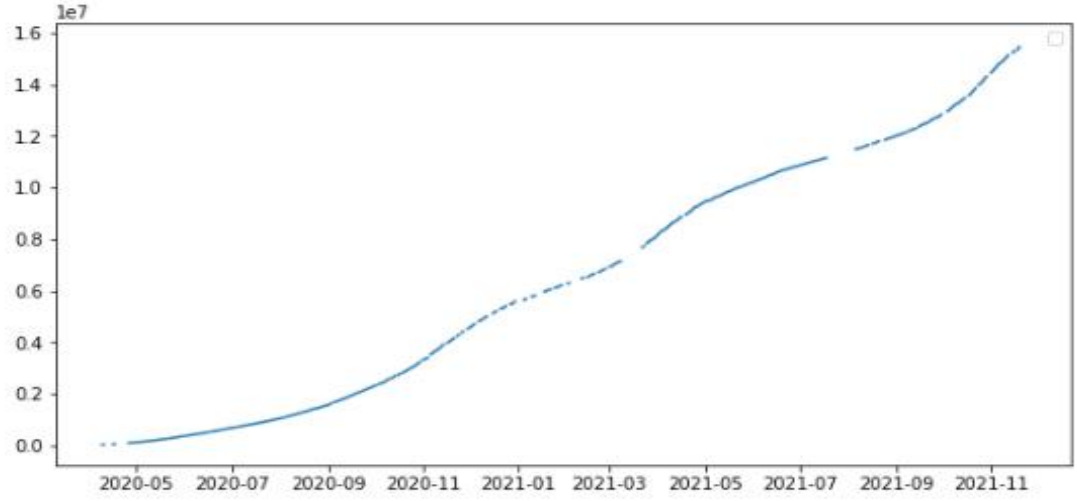

Рисунок 3.3 – Графік зміни даних у стовпці total_tests

При побудові графіка за даними стовпця total_tests (рисунок 3.3) можна побачити, що дані не мають значних викидів, а кількість та розташування пропусків дозволяє застосувати поліноміальну інтерполяцію для їх заповнення (рисунок 3.4, лістинг 3.2).

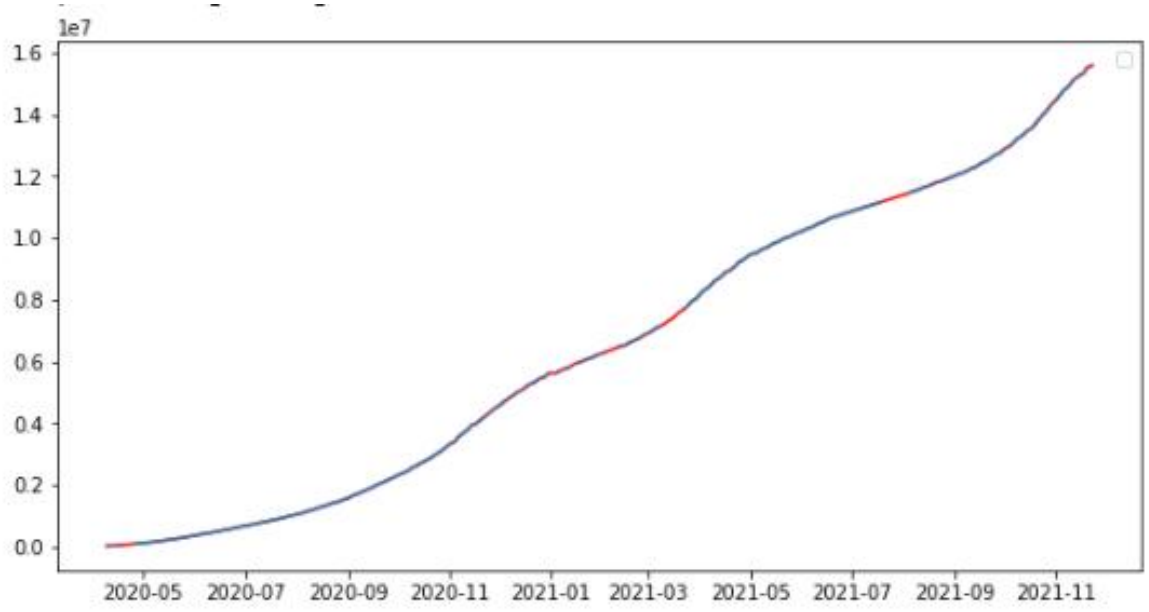

Рисунок 3.4 – Результат усунення пропущених значень за допомогою поліноміальної інтерполяції

Лістинг 3.2 – Фрагмент коду, який відповідає за усунення пропущених значень за допомогою поліноміальної інтерполяції та візуалізацію отриманих значень

```
df['old_total_tests'] = df['total_tests']
interpolated =
df['total_tests'].interpolate(method='polynomial', order=2)
df['total_tests'].fillna(interpolated, inplace=True)

plt.figure(figsize=(10, 5))
plt.plot(df['date'], df['total_tests'], color='red')
plt.plot(df['date'], df['old_total_tests'])
plt.legend()

df = df.drop(['old_total_tests'], axis=1)
```

Стовпець people_fully_vaccinated містить пропущені значення лише до дати початку вакцинації в Україні, тому вони були замінені нулями. Значення стовпця new_tests також містять пропущені значення, але замість застосування інтерполяції для їх заповнення (як у випадку total_tests), значення були перераховані за допомогою стовпця total_tests.

3.2.2 Згладжування даних

Отримані значення new_tests мають тижневу періодичність, а також містять помітні позитивні та негативні викиди. Навчання нейронних мереж на таких даних є недоцільним, тому до даних необхідно застосувати згладжування.

Серед найпопулярніших методів згладжування можна виділити:

- метод ковзних середніх;
- експоненціальне згладжування.

Метод ковзних середніх полягає в усередненні значень в межах вікна, що ковзає по часовому ряду. Ширина ковзного вікна визначає кількість минулих періодів, що враховуються в прогнозуванні. Ширина ковзного вікна визначає кількість значень, які будуть використані для розрахунку

середнього значення і чим більше буде ширина вікна, тим більш плавним буде зміна значень.

Експоненціальне згладжування є одним з найстаріших і найбільш вивчених методів прогнозування часових рядів. Воно найбільш ефективно, коли значення часового ряду слідують за поступовим трендом і відображають сезонну поведінку, в якій значення слідують за періодичною циклічною схемою протягом заданої кількості часових кроків. Даний метод має досить високу точність для короткострокових прогнозів, яка поступово падає зі збільшенням дальності прогнозу.

На відміну від методу ковзних середніх, згідно з яким всі дані мають однакову вагу, в методі експоненціального згладжування найбільший коефіцієнт приписується самому останньому спостереженню. Отже для згладжування часового ряду в якому є викиди даних, більше підходить метод ковзного середнього.

Для згладжування наявних даних було використано вікно шириною, що дорівнює 7 дням (лістинг 3.3).

Лістинг 3.3 – Код, що відповідає за згладжування та візуалізацію інформації про кількість виконаних тестів та випадків захворювання

```
df['new_cases_smoothed'] = df['new_cases'].rolling(7).mean()
df['new_tests_smoothed'] = df['new_tests'].rolling(7).mean()

plt.figure(figsize=(10, 5))

plt.plot(df['date'], df['new_cases_smoothed'], label='New Cases
Smoothed', color='black')
plt.plot(df['date'], df['new_tests_smoothed'], label='New Tests
Smoothed', color='blue')

plt.plot(df['date'], df['new_cases'], label='New Cases', color='
red', alpha=0.7)
plt.plot(df['date'], df['new_tests'], label='New Tests', color='
green', alpha=0.7)

plt.legend()
```

Обробка методом ковзного вікна дозволила усунути тижневу періодичність у стовпцях new_cases та new_tests (рисунок 3.5).

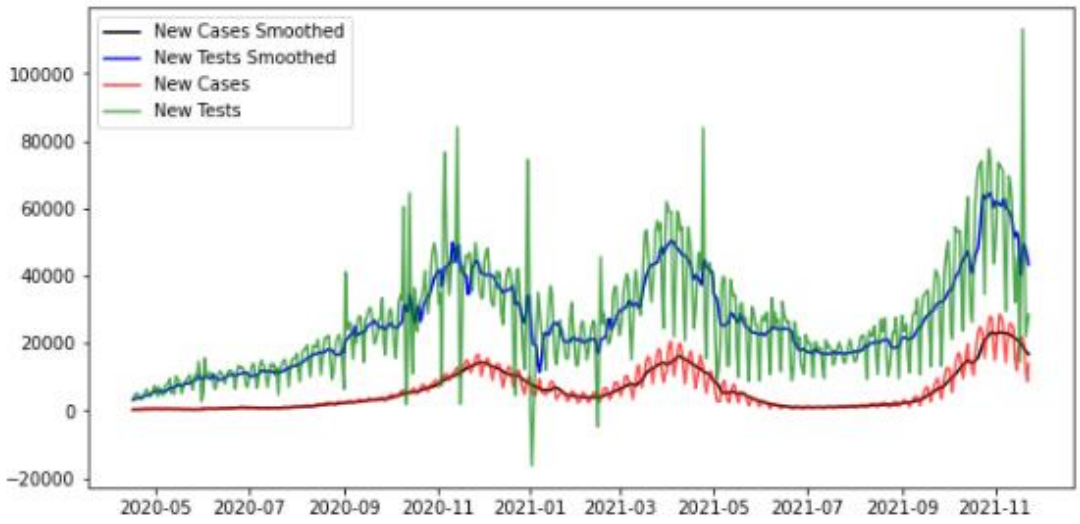

Рисунок 3.5 – Результати згладжування даних для стовпців new_cases та new_tests за допомогою методу ковзного вікна

Інформація про кількість тестів, містить викиди, однак після обробки їх вплив сильно зменшено, тому тепер отримані дані придатні для навчання нейронних мереж. У наборі даних також залишилися порожні значення на початку та в кінці стовпців. Так як немає можливості для достовірного відновлення пропущених значень на початку та в кінці набору даних, рядки які містили ці значення були виключені з кінцевого набору даних.

3.3 Аналіз даних

Після очищення даних було виконано аналіз наявного набору даних за допомогою наступного набору інструментів:

- генератор описової статистики із бібліотеки Pandas;
- генератор звітів про кореляцію даних із бібліотеки Pandas;
- теплові карти із библиотеки Seaborn.

На першому кроці було виконано збір описової статистики за наявними стовпцями.

Статистика включає інформацію про кількість числових значень, мінімальні та максимальні значення, а також інформацію про стандартне відхилення та квартилі значень (таблиця 3.2).

Таблиця 3.2 – Описова статистика для набору даних

	str. index	new tests	new cases	temp.	humidity	vacc. percent
count	588,00	588,00	588,00	588,00	588,00	588,00
mean	59,38	26352,20	5929,48	12,02	0,72	2,76
std	12,40	13685,43	5814,01	7,66	0,07	5,56
min	0,00	3174,00	324,00	-2,21	0,65	0,00
25%	54,63	17074,25	1112,00	6,30	0,67	0,00
50%	58,33	23943,00	3946,50	12,97	0,68	0,00
75%	63,89	35602,00	9428,25	19,25	0,78	1,67
max	88,89	64509,00	23333,00	21,51	0,87	24,02

Далі виконано аналіз можливих взаємозв’язків даних між собою (кореляція), а одержані результати представлені у вигляді матриці (таблиця 3.3).

Таблиця 3.3 – Кореляція даних у стовпцях датасета

	str. index	new tests	new cases	temp.	humidity	vacc. %
stringency_index	1	-0,17	-0,16	-0,21	-0,13	-0,27
new_tests	-0,17	1	0,94	-0,38	0,39	0,48
new_cases	-0,16	0,94	1	-0,49	0,48	0,52
temperature	-0,21	-0,38	-0,49	1	-0,86	0,16
humidity	-0,13	0,39	0,48	-0,86	1	-0,06
vaccinated_percent	-0,27	0,48	0,52	0,16	-0,06	1

Інформація про кореляцію даних для збільшення наочності також була представлена у вигляді теплової карти (рисунок 3.6).

Проаналізувавши теплову карту, можна дійти висновку, що кількість нових випадків захворюваності має максимальну кореляцію з кількістю виконаних тестів і майже однаково піддається впливу зовнішніх параметрів temperature, humidity і vaccinated_percent, а параметр stringency_index має мінімальну кореляцію з даними new_cases.

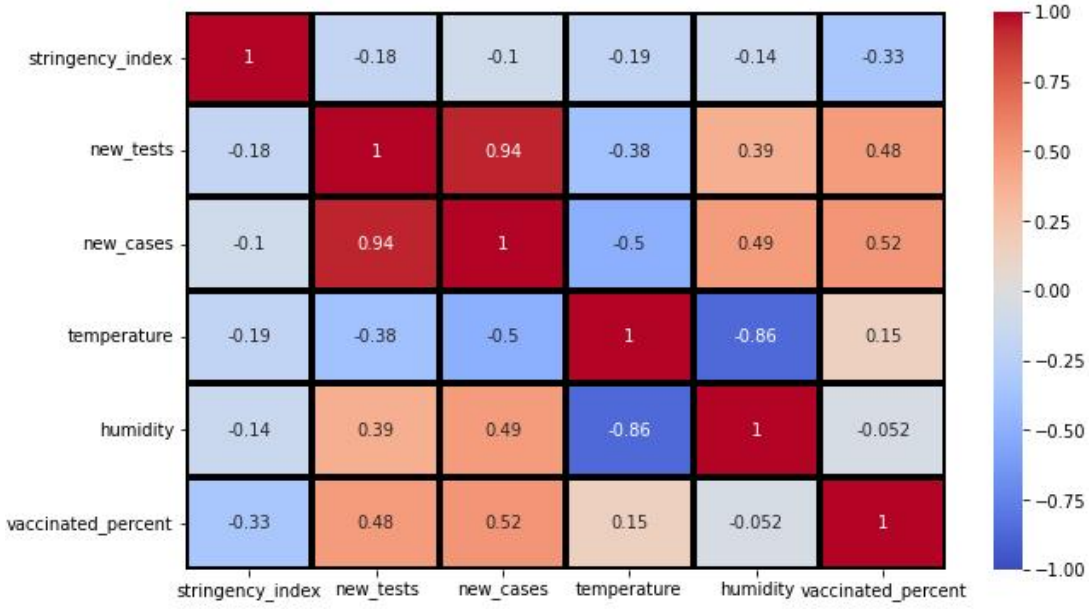

Рисунок 3.6 – Кореляція даних часового ряду, представлена у вигляді теплової карти

3.4 Форматування даних для нейронних мереж

Моделі нейронних мереж не можна навчати одразу на всьому наборі даних, тому що це передбачає, що в вхідному шарі буде кількість входів, що дорівнює розміру набору даних, а його розмір у свою чергу може досягати десятків або сотень тисяч записів. Тому набір довжиною m розбивається на блоки довжиною n і кроком зсуву рівним 1, що дозволяє отримати m-n+1 блоків.

Отримані блоки можна подавати на вхід моделі у випадковому порядку, але всередині блоків дані перемішувати не можна, тому що це призведе до втрати інформації часового ряду. Частину отриманих блоків потрібно залишити для перевірки моделі під час навчання, а частину залишити для навчання. Також потрібно визначити кількість входів і виділити з блоку *a* інтервалів. В такому випадку дальність прогнозування становитиме *n-a* інтервалів [10].

Оскільки нейронні мережі здатні обробляти дані в матричному вигляді, часовий ряд був перетворений на дві послідовності зрізів даних, кожна з яких буде додатково розбита на навчальну та перевірочну вибірку. Кількість зрізів у кожній послідовності залежить від загального розміру набору даних, величини горизонту прогнозування та кількості часових інтервалів, які будуть використані для прогнозування.

Розбиття набору даних на послідовності зрізів виконано за допомогою методу split_frame (лістинг 3.4).

Лістинг 3.4 – Реалізація розбиття набору даних на послідовності зрізів

```
def split_frame(frame, history_size, horizon_size,
result_labels):
  input = []
  output = []
  for i in range(history_size, len(frame)-horizon_size+1):
    temp = frame[i-history_size:i]
    train_columns = frame.columns.difference(result_labels)
    data = temp[train_columns]
    input.append(data)
    temp = frame[i:i+horizon_size]
    data = temp[result_labels]
    output.append(data.to_numpy())
  return np.array(input), np.array(output)
```

Заключним етапом передобробки даних є нормалізація, тобто. приведення їх значень до діапазону від 0 до 1.

Відповідно до таблиці 3.2 нормалізації не потребують значення колонки humidity, т.к. значення вже перебувають у діапазоні від 0 до 1.

Значення в колонках stringency_index та vaccinated_percent вихідного набору даних були поділені на 100, щоб отримати значення в діапазоні від 0 до 1.

Значення temperature перетворені на діапазон від 0 до 1, з урахуванням що максимальне і мінімальне значення температури повітря дорівнюватимуть +36 і -36 градусів цельсія відповідно.

Значення new_tests і new_cases перетворені шляхом віднімання середнього значення та поділу на стандартне відхилення.

3.5 Моделі нейронних мереж

Для реалізації нейронних мереж було використано бібліотеку Keras, яка є обгорткою навколо низькорівневого фреймворку машинного навчання Tensorflow.

Для полегшення проведення подальшого дослідження методів використання ШНМ та стандартизованого тестування було створено клас NNTestBench (лістинг 3.5), який містить у собі реалізації методів fit, set_optimizer та configure_model, інформацію про назву моделі, саму модель, а також історію навчання.

Історію навчання штучної нейронної мережі можна використовувати для візуалізації процесу навчання та виявлення аномалій, що можуть виникати під час навчання моделі.

До аномалій можна віднести:

- різке збільшення точності для тестового набору даних – ознака появи ефекту перенавчання;
- відсутність зменшення втрат – ознака невірно підібраних гіперпараметрів.

Клас NNTestBench надалі був використаний для створення дочірніх класів: LSTMBench, RNNBench, TCNBench та CNNBench.

Лістинг 3.5 – Реалізація базового класу для всіх тестових нейронних мереж

```
class NNTestBench(object):
  name = None

  def __init__(self):
    self.fitlog = None
    self.model = None

  def configure_model(self, params):
    pass

  def set_optimizer(self, optimizer, loss='mse'):
    self.model.compile(
        optimizer=optimizer,
        loss=loss)

  def fit(self, epochs, batch_size, nndata):
    self.fitlog = self.model.fit(
        nndata.train_tuple[0],
        nndata.train_tuple[1],
        epochs=epochs,
        batch_size=batch_size,
        validation_data=nndata.validation_tuple)
```

Оскільки реалізація шарів штучної нейронної мережі за допомогою бібліотеки Keras не є однорідною і кожен з них містить параметри, що відрізняються значеннями та назвами, для кожної нейронної мережі були реалізовані класи LSTMBench, RNNBench, TCNBench та CNNBench, які перевизначають метод configure_model класу NNTestBech (лістинг 3.6, лістинг 3.7).

Це дозволяє конфігурувати моделі нейронних мереж, використовуючи звичайний об’єкт-словник, який завдяки динамічній типізації мови Python може приймати в якості значень будь-які типи даних.

Для додаткової гнучкості, у метод configure_model був доданий код, який дозволяє конфігурувати штучну нейронну мережу безпосередньо, передаючи в якості аргументу список налаштованих шарів.

Ця функціональність потрібна у разі створення та тестування глибоких нейронних мереж, які містять кілька шарів між вхідним та вихідним шаром.

Лістинг 3.6 – Реалізація класу LSTMBench

```
class LSTMBench(NNTestBench):
  name="LSTM"
  def configure_model(self, params, custom_layers = None):
    self.model = Sequential()

    # for custom models
    if custom_layers is not None:
      for layer in custom_layers:
        self.model.add(layer)
        return
    self.model.add(LSTM(
        units=params['units'],
        input_shape=params['shape'],
        activation=params['activation']))
    self.model.add(Dropout(rate=params['dropout']))
    self.model.add(Dense(params['prediction_horizon']))
```

Отримані класи дозволяють спростити автоматизоване тестування та полегшити підбір гіперпараметрів для нейронних мереж. Нижче наведено приклад тестування 10 мереж типу CNN із застосуванням різних гіперпараметрів (лістинг 3.8).

Лістинг 3.7 – Реалізація класу TCNBench

```
class TCNBench(NNTestBench):
  name="TCN"
  def configure_model(self, params, custom_layers = None):
    self.model = Sequential()
    # for custom models
    if custom_layers is not None:
      for layer in custom_layers:
        self.model.add(layer)
        return

    self.model.add(TCN(
        nb_filters=params['filters'],
        kernel_size=params['kernel_size'],
        dilations=params['dilations'],
        dropout_rate=params['dropout'],
        input_shape=params['shape'],
        activation=params['activation']))
    self.model.add(Dense(params['prediction_horizon']))
```

Лістинг 3.8 – Приклад послідовного навчання 10 нейронних мереж CNN із застосуванням різних гіперпараметрів

```
baseCase = dict({
    'shape': nndata.train_tuple[0].shape[-2:],
    'prediction_horizon': prediction_horizon,
    'pool_size': 5
})
params = [
    {'filters': 256,'kernel_size': 2,'stride': 1, 'activation':
'relu', 'dropout': 0.3},
    {'filters': 128,'kernel_size': 2,'stride': 2, 'activation':
'tanh', 'dropout': 0.1},
    {'filters': 64,'kernel_size': 3,'stride': 1, 'activation':
'relu', 'dropout': 0.4},
    {'filters': 32,'kernel_size': 3,'stride': 2, 'activation':
'tanh', 'dropout': 0.2},
    {'filters': 16,'kernel_size': 1,'stride': 3, 'activation':
'relu', 'dropout': 0.0},
    {'filters': 256,'kernel_size': 2,'stride': 1, 'activation':
'tanh', 'dropout': 0.5},
    {'filters': 128,'kernel_size': 4,'stride': 2, 'activation':
'relu', 'dropout': 0.7},
    {'filters': 64,'kernel_size': 1,'stride': 5, 'activation':
'tanh', 'dropout': 0.2},
    {'filters': 32,'kernel_size': 3,'stride': 2, 'activation':
'relu', 'dropout': 0.3},
    {'filters': 16,'kernel_size': 2,'stride': 1, 'activation':
'tanh', 'dropout': 0.5},
]
fitted = []
for param in params:
  testCase = dict(param)
  testCase.update(baseCase)
  cnnbench = CNNBench()
  cnnbench.configure_model(testCase)
  cnnbench.set_optimizer(optimizer)
  cnnbench.fit(epochs=25, batch_size=10, nndata=nndata)
  fitted.append(cnnbench)
```

Застосування класу NNTestBench також дозволило полегшити спільну візуалізацію результатів навчання нейронних мереж.

Навчені нейронні мережі розміщуються в одному масиві, після чого стає можливим проітерувати цей масив, виконати прогнозування та відобразити отримані результати у вигляді графіка (лістинг 3.9).

Лістинг 3.9 – Приклад застосування дочірніх класів NNTestBech для спільної візуалізації результатів навчання нейронних мереж

```
srcFrame = validation_frame[:-40]
dates = srcFrame[-(history_size+prediction_horizon):].index
real_data = srcFrame[-
(history_size+prediction_horizon):]['new_cases']*nc_std +
nc_mean
history_data = srcFrame[-(history_size+prediction_horizon):-
prediction_horizon]
dates_for_predicted_values = srcFrame[-
prediction_horizon:].index.to_numpy()
plt.figure(figsize=(10, 5))
plt.plot(dates, real_data, label='Real Data')
reshaped_history_data =
history_data.to_numpy().reshape((1,history_size,features_size))
for bench in [lstmbench, rnnbench, tcnbench, cnnbench]:
  predicted =
bench.model.predict(reshaped_history_data)*nc_std+nc_mean
  plt.plot(dates_for_predicted_values, predicted[0],
label=f'{bench.name} Prediction')
plt.legend()
```

Як можна бачити, використання NNTestBench дозволяє додати до моделей додаткову інформацію (наприклад, назву моделі або її код), яка полегшить ідентифікацію результатів нейронної мережі на загальному графіку.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Досліджувані параметри нейронних мереж

У ході дослідження було створено 60 нейронних мереж на основі RNN, CNN, LSTM та TCN архітектур із застосуванням різних гіперпараметрів. Загальні результати дослідження наведено у таблиці 1 додатка Б, з такими даними:

- batch – розмір блоку навчальних даних під час пакетної обробки. Збільшення цього параметра призводить до ускладнення навчання та зменшує вплив ефекту перенавчання, що негативно позначається на результатах валідації;
- epoch – кількість ітерацій навчання. Збільшення параметра покращує точність, проте занадто велика кількість ітерацій може призводити до ефекту перенавчання нейронної мережі;
- units (тільки для LSTM и RNN) – кількість осередків стану. Чим більшим буде значення units, тим більший обсяг пам’яті матиме нейронна мережа;
- filters (тільки для TCN и CNN) – результуюча кількість згорток на виході шару;
- kernel_size (тільки для TCN и CNN) – розмір згорткового ядра;
- mse (Mean Squared Error) – середньоквадратична помилка. Чим кращі результати показує нейронна мережа, тим менше значення середньоквадратичної помилки;
- history – розмір однієї вибірки даних, яка використовуватиметься для виконання одного прогнозування;
- prediction horizon – розмір горизонту прогнозування. Визначає на скільки днів наперед виконуватиметься прогнозування.

4.2 Базова конфігурація нейронних мереж

Для нейронних мереж, в якості базових конфігурацій були визначені наступні набори гіперпараметрів:

- ШНМ LSTM: Units – 16; Activation – tanh; Dropout – 0.4; PredictionHorizon – 20;
- ШНМ RNN: Units – 64; Activation – tanh; Dropout – 0.4, PredictionHorizon – 20;
- ШНМ TCN: Filters – 64, Activation – relu, Dropout – 0.3, KernelSize – 3, Dilations – 1, 2, 4, 8, 16, 32; PredictionHorizon – 20;
- ШНМ CNN: Filters – 64, Activation – relu; Dropout – 0,3; KernelSize – 2; Stride – 1; PredictionHorizon – 20;

Глобальні гіперпараметри: Epoch – 25, Batch – 10.

4.3 Залежність результатів від попередньої обробки даних

Щоб дослідити вплив попередньої обробки даних на якість прогнозування нейронних мереж, використовувалися нейронні мережі в базовій конфігурації (конфігурація описана в розділі 4.2) при горизонті прогнозування 20 днів і розмірі історії 60 днів.

Таблиця 4.1 – Результати тестування впливу попередньої обробки даних на результати прогнозування

Варіант обробки	Значення Mean Squared Error			
	LSTM	RNN	TCN	CNN
Без попередньої обробки	0,6205	0,5052	0,2805	0,5623
Згладжування new_cases	0,4623	0,4052	0,2001	0,4281
Згладжування new_tests	0,5648	0,4896	0,2249	0,4913
Згладжування new_cases та new_tests	0,4227	0,4006	0,1922	0,4089

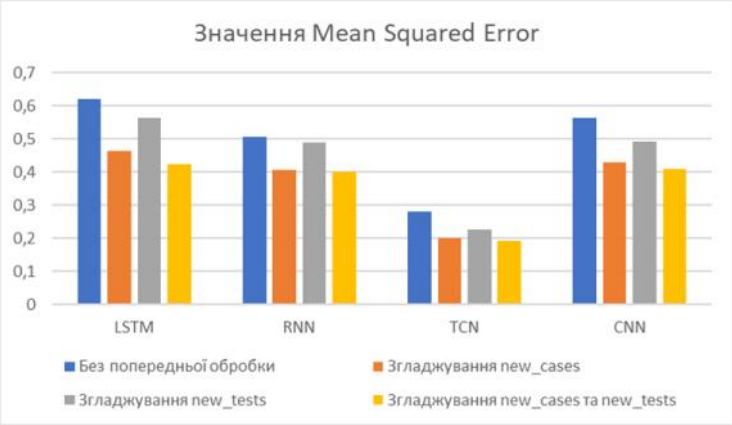

Рисунок 4.1 – Візуалізація залежності Mean Squared Error від попередньої обробки даних

Після аналізу отриманих результатів (таблиця 4.1, рисунок 4.1) можна зробити висновок, що відсутність згладжування для даних в колонках new_cases і new_tests збільшує значення MSE. Збільшення MSE переважно залежить від наявності згладжування значень прогнозованих даних, тобто значень стовпця new_cases. Згладжування значень new_tests впливає на результат у меншому ступені.

Найбільш чутливою до відсутності згладжування є нейронна мережа на основі LSTM. Нейронна мережа на основі CNN показала результати краще, ніж LSTM, але гірше, ніж мережі на основі RNN та TCN.

Найбільш стійкою до відсутності згладжування є нейронна мережа на основі TCN. Різниця між найбільшим та найменшим значенням MSE для TCN склала близько 0,088.

4.4 Залежність результатів нейронних мереж від параметрів прогнозування та налаштувань навчання

Щоб дослідити вплив параметрів batch, history, history, prediction horizon на якість прогнозування нейронних мереж, використовувалися нейронні мережі у базовій конфігурації (конфігурація описана у розділі 4.2).

Таблиця 4.2 – Результати тестування впливу параметрів прогнозування та налаштувань навчання на результати прогнозування

№	Параметри			Значення Mean Squared Error			
	batch	history	prediction horizon	LSTM	RNN	TCN	CNN
1	10	60	20	0,4154	0,3908	0,2011	0,4002
2	5	60	20	0,6203	0,5103	0,3341	0,6134
3	1	60	20	0,9313	0,7842	0,4345	0,8011
4	10	30	10	0,0983	0,2057	0,0921	0,2291
5	5	30	10	0,1192	0,4179	0,1390	0,2985
6	1	30	10	0,1241	0,6813	0,7313	0,6954
7	10	15	5	0,0423	0,1512	0,1174	0,1103
8	5	15	5	0,0753	0,6073	0,2068	0,1385
9	1	15	5	0,0531	0,8922	0,4315	0,2315

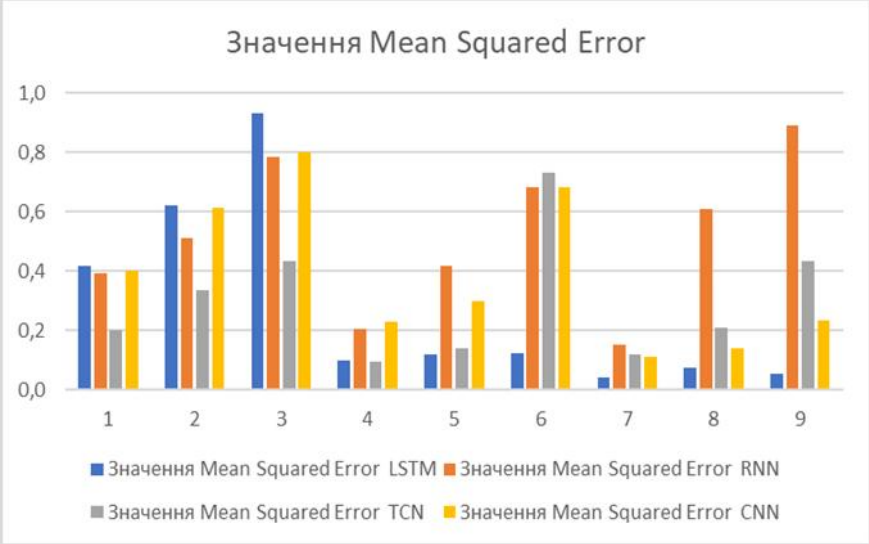

Рисунок 4.2 – Візуалізація залежності Mean Squared Error від параметрів прогнозування та налаштувань навчання

Проаналізувавши рисунок 4.2 та таблицю 4.2 можна помітити, що значення MSE значно збільшується при збільшенні горизонту прогнозування часового ряду.

При розмірі горизонту прогнозування рівним 20 дням, найгірші результати показала мережу на основі LSTM. Мережі на основі RNN та CNN показали схожі результати, а мережа TCN перевершила результати інших мереж майже вдвічі. При зменшенні горизонту прогнозування до 10 та встановлення BatchSize рівним 10, мережа LSTM зменшила значення MSE більш ніж у 4 рази, тоді як інші нейронні мережі скоротили це значення лише вдвічі. При зменшенні значення BatchSize до 1, значення MSE збільшилося у всіх нейронних мереж крім LSTM. При зменшенні горизонту прогнозування до 1 значення MSE збільшилося для мережі RNN і зменшилося для LSTM, TCN і CNN.

4.5 Вплив гіперпараметрів на навчання моделі та результати прогнозування

Для дослідження впливу гіперпараметрів на навчання моделей та результати прогнозування були використані моделі з базовою конфігурацією (конфігурація описана в розділі 4.2) при горизонті прогнозування 20 днів і розмірі історії 60 днів (таблиця 4.3).

Таблиця 4.3 – Результати зміни гіперпараметрів моделі для мережі LSTM

№	units	dropout	MSE	Час навчання
1	16	0,4	0,4050	18
2	32	0,4	0,4397	18
3	64	0,8	0,4037	23
4	128	0,4	0,5023	35
5	64, 32	0,4; 0,4	0,5131	36
6	8	0,4	0,3176	18
7	4	0,4	0,4113	18
8	2	0,4	0,6467	18

Проаналізувавши результати тестування (таблиця 4.3) можна зробити висновок, що зі збільшенням розмірності вихідного простору (гіперпараметр units) у шарах LSTM мережі починається поступове збільшення значення MSE і часу навчання. Зміна значення Dropout до 0.8 дозволило уповільнити збільшення MSE та процес перенавчання.

При додаванні нового шару значення MSE збільшується на 0.1, а при зменшенні гіперпараметру units у першому шарі до 8, значення MSE зменшилося до 0,3176. Подальше зменшення гіперпараметру units у першому шарі призвело до збільшення MSE та втрати точності прогнозування (рисунок 4.3).

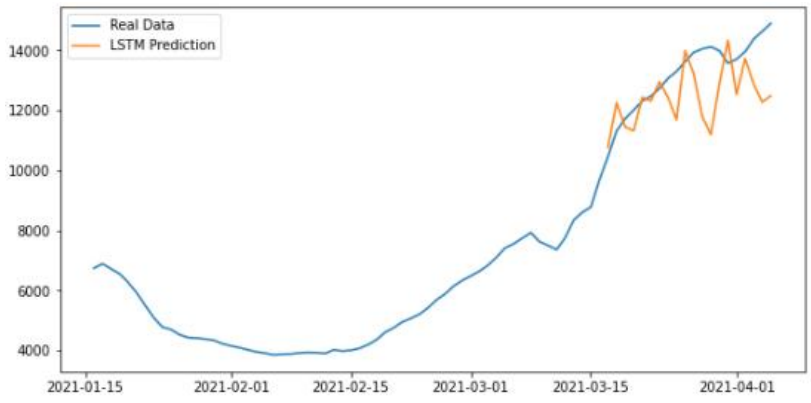

Рисунок 4.3 – Зменшення точності прогнозування через надмірне зменшення кількості вузлів у мережі LSTM

Таблиця 4.4 – Залежність MSE та часу навчання від гіперпараметрів RNN

№	units	dropout	activation	MSE	Час навчання
1	64	0,4	tanh	0,2293	8
2	64	0,4	relu	0,1453	9
3	16	0,4	tanh	0,4354	8
4	4	0,4	tanh	0,5323	8
5	128	0,4	tanh	0,1103	12
6	128	0,8	tanh	0,5831	12

Аналізуючи отримані результати (таблиця 4.4) можна зробити висновок, що зменшення розмірності (гіперпараметр units) призводить до значного збільшення MSE, у той час як збільшення розмірності до 128 дозволило скоротити MSE майже в 2 рази. Збільшення значення гіперпараметру dropout без зміни кількості епох також призвело до значного збільшення MSE.

Таблиця 4.5 – Вплив гіперпараметрів на час навчання та точність мережі на основі TCN

№	filters	kernel_size	dilations	MSE	Час навчання
1	64	3	1,2,4,8,16,32	0,2056	26
2	32	3	1,2,4,8,16,32	0,5641	24
3	128	3	1,2,4,8,16,32	0,1783	67
4	64	5	1,2,4,8,16,32	0,1825	33
5	64	2	1,2,4,8,16,32	0,2923	24
6	64	3	1,2,4,8	0,3564	16
7	64	3	1,4,16,64	0,1090	30

Як і у випадку розглянутих вище нейронних мереж, зменшення розрядності внутрішнього шару (гіперпараметр filters) призводить до збільшення значення MSE та незначного зменшення часу навчання (таблиця 4.5). Збільшення filters до 128 призвело до зменшення MSE до 0,1783 та збільшення часу навчання майже вдвічі.

Зміна розміру ядра згортки (гіперпараметр kernel_size) має незначний вплив на MSE та час навчання.

Збільшення розміру ядра призводить до зменшення MSE та збільшення часу навчання.

Зміна гіперпараметру dilations впливає на глибину отриманої нейронної мережі. Видалення 2 рівнів глибини призвело до збільшення значення MSE та скорочення часу навчання.

Зміна dilations із ступеня двійки на ступені четвірки призвело до скорочення значення MSE майже в 3 рази та збільшення часу навчання з 16 секунд до 30.

Виконання аналогічного тестування для мережі на основі CNN показало, що зменшення значення filters призводить до збільшення значення MSE і значного зменшення часу навчання, а збільшення filters до 128 покращило значення MSE до 0,3702 і збільшило час навчання до 12 секунд (таблиця 4.6).

Збільшення kernel_size з 2 до 5 призвело до збільшення MSE до 0,9928, що зробило цю модель непридатною для прогнозування.

Збільшення страйду ядра також негативно вплинуло значення MSE.

Таблиця 4.6 – Вплив гіперпараметрів на час навчання та точність мережі на основі CNN

№	filters	kernel_size	kernel_stride	MSE	Час навчання
1	64	2	1	0,4054	8
2	32	2	1	0,5635	6
3	16	2	1	0,7952	3
4	128	2	1	0,3702	12
5	64	5	1	0,9928	4
6	64	2	2	0,4623	4
7	64	2	3	0,4871	4

4.5 Візуальний аналіз якості прогнозування

Для візуального аналізу якості прогнозування вибрано конфігурації нейронних мереж, у яких за результатами експериментів було найменше значення MSE для прогнозування на 20 днів уперед при врахуванні згладжування даних у стовпцях new_cases та new_tests.

Вибрано наступні конфігурації:

- LSTM (Units = 8; Dropout = 0.4, Activation = tanh). Досягнуте значення MSE дорівнює 0,3176;

- RNN (Units = 128; Dropout = 0.4, Activation = tanh). Досягнуте значення MSE дорівнює 0,1103;

- TCN (Filters = 64; KernelSize = 3; Dilations = 1,4,16,64; Dropout = 0.3). Досягнуте значення MSE дорівнює 0,1090;

- CNN (Filters = 128; KernelSize = 2; KernelStride = 1). Досягнуте значення MSE дорівнює 0,3702.

В якості тестових даних з перевірочного набору даних були обрані 2 проміжки зі зростаючим (рисунок 4.4) і спадним трендом (рисунок 4.5).

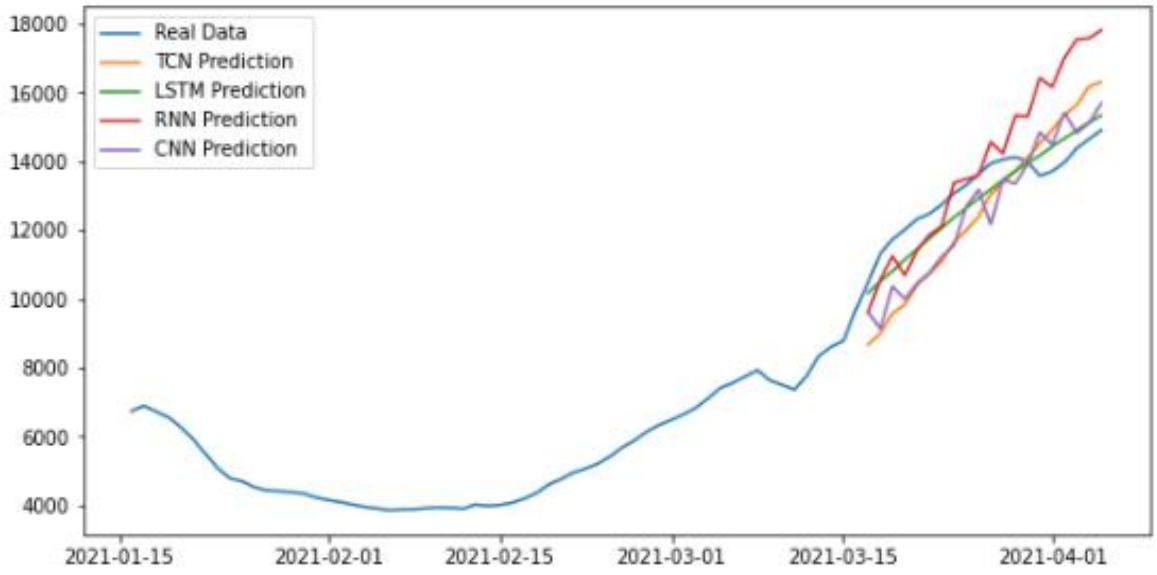

Рисунок 4.4 – Результати прогнозування чотирьох нейронних мереж на відрізку зі зростаючим трендом

Враховуючи візуалізацію прогнозування на рисунку 4.4, можна сказати, що мережі на основі TCN і LSTM показують кращі результати. Найгірший результат показала нейронна мережа на основі RNN, результати якої значно відрізняються від реальних даних.

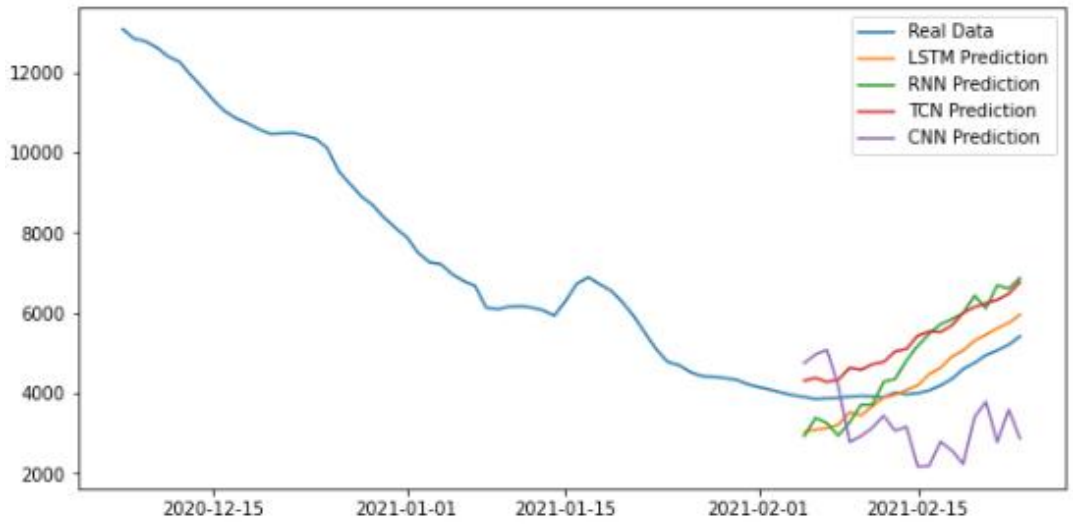

Рисунок 4.5 – Результати прогнозування чотирьох нейронних мереж на відрізку з спадним трендом

Проаналізувавши візуалізацію прогнозування на рисунку 4.5 можна помітити, що результати нейронної мережі CNN представлені у вигляді ламаної лінії і значно відрізняються від реальних даних.

Результати LSTM мережі виявилися найближчими до реальних даних. Мережі RNN і TCN також як і LSTM врахували можливе зростання захворюваності, проте прогнозована кількість нових випадків захворювання виявилася вищою за реальні дані приблизно на 2 тисячі.

З огляду на отримані результати можна дійти висновку, що найкращі результати показує LSTM мережа. Результати, що досягнуті за допомогою TCN, порівняні з результатами LSTM, але мають меншу точність. Мережі RNN та CNN застосовувати недоцільно, т.к. результати, отримані за допомогою цих мереж, значно відрізняються від реальних даних.

ВИСНОВКИ

У роботі досліджені і проаналізовані методи прогнозування пандемій за допомогою штучних нейронних мереж з урахуванням зовнішніх факторів. В якості вихідних даних використовувалися дані про захворюваність COVID-19, зібрані Всесвітньою Організацією Охорони Здоров’я (ВООЗ). Створено та проаналізовано 60 моделей штучних нейронних мереж на основі 4 архітектур з різними гіперпараметрами.

Наукова новизна полягає в застосуванні різних моделей нейронних мереж для прогнозування числа захворілих під час пандемії COVID-19, як на основі даних про кількість хворих, так і на основі даних про кількість вакцинованих, даних про погоду, кількість виконаних тестів, а також індексу суворості запроваджених карантинних обмежень.

Для дослідження і аналізу даних використовувався інтерфейс обчислювальної платформи Jupyter Notebook та мова програмування Python. Для створення моделей були використані наступні види штучних нейронних мереж:

- RNN і LSTM;
- CNN і TCN.

Отримані моделі доцільно застосувати для прогнозування перебігу інших пандемій при наявності достатнього набору даних і перенавчанні моделей під нові дані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методы прогнозирования [Електронний ресурс] – Режим доступу : www/ URL: https://up-pro.ru/encyclopedia/metody-prognozirovaniya/ – 15.11.2021 р. – Загол. з екрану.

2. Бриллинджер, Д. Р. Временные ряды. Обработка данных и теория [Текст] / Д. Р. Бриллинджер. – М.: Мир, 1980. – 536 с.

3. Chatfield, C. Time-Series Forecasting [Текст] / C. Chatfield. – Chapman and Hall/CRC, 2000. – 280 с.

4. Shahin, A. A. Deep Learning BiLSTM Encoding-Decoding Model for COVID-19 Pandemic Spread Forecasting [Текст] / A. Shahin, S. Almotairi // Fractal and Fractional. – 2021. – Vol. 5, №4. – P. 175–176.

5. Marzouka, M. Deep learning model for forecasting COVID-19 outbreak in Egypt [Текст] / M. Marzouka, N. Elshaboury, A. Abdel, S. Azab // Process Safety and Environmental Protection. – 2021. – Vol. 153. – P. 363–375.

6. Lecun, Y. Gradient-Based Learning Applied to Document Recognition [Текст] / Y. Lecun et al // Proceedings of the IEEE. – 1998. – Vol. 86, №11. – P. 2278–2324.

7. Understanding LSTM Networks [Електронний ресурс] – Режим доступу : www/ URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ – 15.11.2021 р. – Загол. з екрану.

8. Understanding outliers in time series analysis [Електронний ресурс] – Режим доступу : www/ URL: https://pro.arcgis.com/ru/pro-app/latest/tool-reference/space-time-pattern-mining/understanding-outliers-in-time-series-analysis.htm. – 15.11.2021 р. – Загол. з екрану.

9. Shi, X. Convolutional LSTM Network: a machine learning approach for precipitation nowcasting [Текст] / X. Shi, Z. Chen, H. Wang, D. Yeung // Proceedings of the 28th International Conference on Neural Information Processing Systems. – 2015. – Vol. 1. – P. 802–810.

10. How to Develop Convolutional Neural Network Models for Time Series Forecasting [Електронний ресурс] – Режим доступу : www/ URL: https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/ – 15.11.2021 р. – Загол. з екрану.

11. How to Develop LSTM Models for Time Series Forecasting [Електронний ресурс] – Режим доступу : www/ URL: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/ – 15.11.2021 р. – Загол. з екрану.

12. Duccio, F. Analysis and forecast of COVID-19 spreading in China, Italy and France [Текст] / F. Duccio, P. Francesco // Chaos, Solitons & Fractals. – 2020. – Vol. 134. – P. 1–5.

13. Обработка пропусков в данных [Електронний ресурс] – Режим доступу : www/ URL: https://loginom.ru/blog/missing/ – 15.11.2021 р. – Загол. з екрану.

14. Kingma, D. A. A method for stochastic optimization [Текст] / D. Kingma, B. Jimmy // 3rd International Conference on Learning Representations. – 2015. – Vol. 1. – P. 1–15.

15. Time-Series Forecasting: Predicting Stock Prices Using An LSTM Model [Електронний ресурс] – Режим доступу : www/ URL: https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f/ – 15.11.2021 р. – Загол. з екрану.

16. Obrubov, M. Using LSTM network for solving the multidimentional time series forecasting problem [Текст] / M. Obrubov, S. Kirillova // Национальная ассоциация ученых. – 2021. – Том 2, №68. – С. 43–48.

17. Convolutional autoencoders in python/theano/lasagne [Електронний ресурс] – Режим доступу : www/ URL: https://swarbrickjones.wordpress.com/2015/04/29/convolutional-autoencoders-in-pythontheanolasagne/ – 15.11.2021 р. – Загол. з екрану.

18. How to Check if Time Series Data is Stationary with Python [Електронний ресурс] – Режим доступу : www/ URL:

https://machinelearningmastery.com/time-series-data-stationary-python/ – 15.11.2021 р. – Загол. з екрану.

19. Crash Course in Recurrent Neural Networks for Deep Learning [Електронний ресурс] – Режим доступу : www/ URL: https://machinelearningmastery.com/crash-course-recurrent-neural-networks-deep-learning/ – 15.11.2021 р. – Загол. з екрану.

20. Srivastava, N. Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Текст] / N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever // Journal of Machine Learning Research. – 2014. – №15. – P. 1929–1958.

21. Jose, M. Adding external factors in Time Series Forecasting [Текст] / M. Jose. – School of Electrical Engineering and Computer Science (EECS), 2020. – 68 p.

22. COVID-19: Stringency Index [Електронний ресурс] – Режим доступу : www/ URL: https://ourworldindata.org/covid-stringency-index. – 15.11.2021 р. – Загол. з екрану.

23. Oxford COVID-19 Government Response Tracker [Електронний ресурс] – Режим доступу : www/ URL: https://covidtracker.bsg.ox.ac.uk. – 15.11.2021 р. – Загол. з екрану.

24. Temporal Convolutional Networks, The Next Revolution for Time-Series? [Електронний ресурс] – Режим доступу : www/ URL: https://towardsdatascience.com/temporal-convolutional-networks-the-next-revolution-for-time-series-8990af826567. – 15.11.2021 р. – Загол. з екрану.

25. Jining, Y. Temporal Convolutional Networks for the Advance Prediction of ENSO [Текст] / Y. Jining // Scientific Reports. – 2020. – Vol. 10, №6055. – P. 1–15.

26. Wu, Y. Effects of temperature and humidity on the daily new cases and new deaths of COVID-19 in 166 countries [Текст] / Y. Wu, W. Jing, L. Jue // Science of The Total Environment. – 2020. – Vol. 729. – P. 1–7.

27. Effects of temperature and humidity on the spread of COVID-19: A systematic review [Електронний ресурс] – Режим доступу : www/ URL:

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0238339. – 15.11.2021 р. – Загол. з екрану.

28. Maltsev, V. Neural network optimizers [Текст] / V. Maltsev // Научный форум: Инновационная наука. – 2019. – Том. 1, №4. – С. 61-65.

29. Keras TCN [Електронний ресурс] – Режим доступу : www/ URL: https://github.com/philipperemy/keras-tcn. – 15.11.2021 р. – Загол. з екрану.

30. Awad, M. Efficient learning machines: theories, concepts, and applications for engineers and system designers [Текст] / M. Awad, R. Khanna. – Apress Open, 2015. – 263 с.