

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»  
Кафедра інформатики

**ПРОГНОЗУВАННЯ ЕПІДЕМІЧНОЇ СИТУАЦІЇ  
ВИКЛИКАНОЇ ЗАХВОРЮВАННЯМ COVID-19 ЗА  
ДОПОМОГОЮ МЕТОДУ АНАЛІЗУ ЧАСОВИХ РЯДІВ**

**Текстова частина до курсової роботи  
за спеціальністю „Інженерія програмного забезпечення” 6.050103**

Керівник курсової роботи  
канд. техн. наук Ковалюк Т.В.

\_\_\_\_\_  
(підпис)  
“ \_\_\_\_ ” \_\_\_\_\_ 2020 р.

Виконав студент  
Киян М.Є.  
“ \_\_\_\_ ” \_\_\_\_\_ 2020 р.

Київ 2020

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»  
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Кандидат технічних наук,  
\_\_\_\_\_ Ковалюк Т. В.

(підпис)

“ \_\_\_\_ ” \_\_\_\_\_ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ  
на курсову роботу

студенту Кияну М.Є. факультету інформатики 1 курсу 2 (магістерського) рівня

ТЕМА «Прогнозування епідемічної ситуації викликаной захворюванням COVID-19 за допомогою методу аналізу часових рядів»

Вихідні дані:

- кінцевий застосунок, у якому реалізовано прогнозування епідемічної ситуації на основі сингулярного спектрального аналізу на мові програмування C# з використанням фреймворку ML.NET та WPF технології для реалізації візуальної частини застосунку;
- текстова частина курсової роботи.

Зміст текстової частини до курсової роботи:

Індивідуальне завдання

Вступ

1 Теоретичні відомості

2 Розробка моделі та візуалізація результатів прогнозування

3 Аналіз отриманих результатів

Висновки

Список літератури

Дата видачі “ \_\_\_\_ ” \_\_\_\_\_ 2020 р. Керівник \_\_\_\_\_  
(підпис)

Завдання отримав \_\_\_\_\_  
(підпис)

**Тема:** Прогнозування епідемічної ситуації викликаній захворюванням COVID-19 за допомогою методу аналізу часових рядів

**Календарний план виконання роботи:**

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	15.11.2020	
2.	Огляд літератури за темою роботи.	30.12.2020	
3.	Огляд алгоритму Singular Spectrum Analysis	05.02.2020	
4.	Порівняння алгоритму Singular Spectrum Analysis з регресійними алгоритмами прогнозування	10.03.2020	
5.	Визначення функціональних вимог до застосунку.	20.03.2020	
6.	Реалізація прогнозування за допомогою алгоритму Singular Spectrum Analysis з використанням ML.NET.	20.04.2020	
7.	Розробка дизайну застосунку.	22.04.2020	
9.	Написання пояснювальної роботи.	06.05.2020	
10.	Створення слайдів для доповіді та написання доповіді.	07.05.2020	
11.	Аналіз отриманих результатів з керівником, написання доповіді та попередній захист курсової роботи.	08.05.2020	
13.	Захист курсової роботи.	18.05.2020	

Студент \_\_\_\_\_ Киян М. Є. \_\_\_\_\_

Керівник \_\_\_\_\_ Ковалюк Т.В. \_\_\_\_\_

“        ”  
\_\_\_\_\_

## ЗМІСТ

АНОТАЦІЯ.....	3
ВСТУП.....	4
РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ.....	7
1.1 Поняття часового ряду та його аналізу.....	8
1.2 Сингулярний спектральний аналіз.....	9
РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ПРОГНОЗУВАННЯ.....	10
2.1 ML.NET та принципи роботи.....	10
2.2 Обробка статистичних даних.....	12
2.3 Визначення функціональних вимог.....	13
2.4 Реалізація функціональної частини застосунку.....	14
2.5 Реалізація візуальної частини застосунку.....	15
РОЗДІЛ 3. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ.....	20
3.1 Основні характеристики прогнозування.....	20
3.2 Розрахунок основних метрик отриманого прогнозу.....	22
3.3 Тестування застосунку.....	24
ВИСНОВКИ.....	29
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	31
ДОДАТОК А.....	32

## АНОТАЦІЯ

- Об'єм роботи – 30 сторінок (13 рисунків, 9 таблиць, 4 формули);
- Кількість використаних джерел – 11;
- Мета роботи – прогнозування епідемічної ситуації викликаной захворюванням COVID-19 у різних країнах, у тому числі і України;
- Основні результати роботи – розроблено кінцевий застосунок, у якому реалізовано прогнозування епідемічної ситуації на основі сингулярного спектрального аналізу на мові програмування C# з використанням фреймворку ML.NET та WPF технології для реалізації візуальної частини застосунку;
- Ключові слова: аналіз часових рядів, сингулярний спектральний аналіз, ML.NET, прогнозування епідемічних ситуацій, C#, WPF.

## ВСТУП

Новий 2020 рік почався для людства дуже тривожно. 31 грудня 2019 року Всесвітня організація охорони здоров'я отримала сповіщення від китайської влади про новий тип вірусу, який з'явився у китайському місті Ухань (провінція Хубей) з населенням понад 11 млн осіб. Також повідомлялось про те, що задля запобігання поширення хвороби влада у місті закрила місцевий оптовий ринок морепродуктів, який згодом буде названий джерелом вірусу.

Пізніше з'ясувалося, що причиною захворювання став раніше невідомий вірус SARS-CoV-2, що відноситься до сімейства коронавірусів. Загалом сімейство даних вірусів на січень 2020 року включає 40 видів РНК-вірусів, які вражають людей і тварин. Назва пов'язана з будовою вірусу, оскільки його шиповидні відростки нагадують сонячну корону. Загроза SARS-CoV-2 полягає у викликанні небезпечного інфекційного захворювання – COVID-19, що у свою чергу може бути ускладнене вірусною пневмонією і смертельною дихальною недостатністю.

Після двох засідань Комітету Всесвітньої організації охорони здоров'я, голова Організації доктор Тедрос Адханом Гебрейесус оголосив епідемію коронавірусу надзвичайною ситуацією міжнародного значення (тоді масштаби захворювання не відповідали означенню пандемії, про яку почнуть говорити пізніше).

13 січня було виявлено перший випадок зараження за межами Китаю – у Тайланді, у жінки з Ухані. Згодом коронавірус почав різко поширюватися в Європі та світі. Одна за одною країни оголошували про виявлення перших заражених на новий тип вірусу. З тих пір COVID-19 заразилися більше 3 млн людей, і майже 240 тис. померли у 180 країнах світу.

11 березня 2020 року Всесвітня організація охорони здоров'я охарактеризувала світовий масштаб поширення хвороби як пандемію. Цього ж дня Кабінет міністрів прийняв постанову «Про запобігання поширенню на

території України коронавірусу COVID-19», якою на території країни встановлено карантин з 12 березня до 3 квітня 2020 р., який згодом був продовжений спочатку до 27 квітня, а пізніше до 11 травня. Влада України евакуювала громадян з інших країн, закрила кордони, призупинила залізничне, авіа- й автобусне сполучення, міжміські і міжобласні пасажирські перевезення, і це лише частина заходів запобігання поширенню хвороби на території нашої країни, що були використані.

Вже майже півроку перед владою кожної держави щодня постають серйозні питання: чи вводити/послаблювати/знімати карантинний режим, за яких умов можна це робити, якого навантаження лікарень очікувати, коли можна вважати, що громадяни у безпеці, а ризик виникнення нового спалаху епідемії мінімальний тощо.

Іншими словами, для усього світу прогнозування епідемічної ситуації стало дуже актуальним та важливим питанням, адже це необхідна процедура для прийняття рішення щодо запобігання або виходу з такої критичної ситуації як епідемія.

Кожен день оприлюднюються статистичні дані щодо кількості виявлених хворих на COVID-19, кількості летальних випадків та пацієнтів, що одужали. У багатьох країнах, у тому числі і в Україні, збирається розширена статистика з урахуванням більш детальних характеристик таких як стать, вік, місто, професія. З математичної точки зору ці дані є часовим рядом. Саме завдяки аналізу часових рядів ми можемо описати дані (відобразити на графіку), створити модель, спрогнозувати дані через деякий проміжок часу, що може послужити основою прийняття багатьох рішень.

Метою роботи є прогнозування епідемічної ситуації викликаной захворюванням COVID-19 у різних країнах, у тому числі і України.

Для досягнення поставленої мети необхідно було виконати наступні завдання:

1. Опрацювати літературу, присвячену часовим рядам, та методам їх аналізу.

2. Знайти релевантні дані щодо кількості хворих на коронавірусну інфекцію для подальшої роботи з ними.
3. Обрати метод для власного прогнозування епідемічної ситуації викликаной захворюванням COVID-19.
4. На мові C# розробити модель, а також функціональну та візуальну частини застосунку згідно до визначених вимог.
5. Протестувати модель на різних даних (у залежності від країн).
6. Зробити висновки.

Об'єктом дослідження є часові ряди.

Методи дослідження – аналіз та комп'ютерне моделювання.

Робота складається зі вступу, 3 розділів та списку використаної літератури з 11 джерел.

У першому розділі описані поняття часового ряду та його аналізу, а також сингулярний спектральний аналіз, на основі якого у кінцевому застосунку відбувається прогнозування епідемічної ситуації викликаной захворюванням COVID-19 у різних країнах, у тому числі і України.

У другому розділі був описаний фреймворк для машинного навчання ML.NET та принципи його роботи, а також процес отримання та перетворення набору даних для подальшої роботи з ними. Було визначено функціональні вимоги для майбутнього застосунку, відповідно до яких було реалізовано функціональну та візуальну частини, що також було описано у цьому ж розділі.

У третьому розділі був проведений аналіз отриманих результатів, а саме: описані основні характеристики прогнозування, які були використані для розрахунку отриманого прогнозування. Також було проведено тестування, та приведені метрики для таких країн світу як Україна, Китай, США.



## РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Прогнозування інфекційної захворюваності активно розвивається з початку XX століття. Число робіт на цю тему стрімко зростає завдяки розгортанню інформаційних систем та появи великих обсягів статистичних даних, доступних для аналізу. Епідемічні прогнози обраховуються для різних термінів, залежно від яких, використовуються для різних цілей. Так, короткостроковий прогноз на кілька тижнів вперед застосовується в оперативному управлінні і при виявленні епідемічних спалахів захворюваності. У випадку пандемії 2020 року, враховуючи, що за думкою науковців вакцина буде розроблюватись та тестуватись впродовж як мінімум 2 років, актуальним є як короткостроковий прогноз, так і довгостроковий.

Розглянемо доцільність застосування методів аналізу часових рядів для даних, що характеризують епідемічну ситуацію. Даний підхід полягає в ідентифікації моделі ряду, що описує кореляційну залежність між відліками та дозволяє врахувати:

- 1) стохастичний характер вхідних даних;
- 2) одночасно епідемічний період і період між епідеміями;
- 3) взаємозв'язок показників та розгляд епідемічної ситуації як єдиного процесу.

Усі реальні процеси у природі, як і епідемічний процес, безумовно є нелінійними динамічними процесами. Проте на практиці частіше використовують лінійне наближення через простоту та високу точність результатів у багатьох ситуаціях. Значення часового ряду, що прогнозуються, у даному випадку представляє собою лінійну функцію поточних та попередніх значень процесу.

Отже, можемо зробити висновок, що використовувати часові ряди та їх аналіз для прогнозування епідемічної ситуації є доцільним.

### *1.1 Поняття часового ряду та його аналізу*

Часовий ряд (або ряд динаміки) – зібраний у різні моменти часу статистичний матеріал про значення будь-яких параметрів досліджуваного процесу. Кожна одиниця статистичного матеріалу називається виміром або відліком, також допустимо називати його рівнем у пов'язаний з ним момент часу. У часовому ряді для кожного відліку має бути зазначено час вимірювання або номер вимірювання за порядком. Часовий ряд істотно відрізняється від простої вибірки даних, оскільки при аналізі враховується взаємозв'язок вимірювань з часом, а не тільки статистична різноманітність і статистичні характеристики вибірки [1].

У якості показника часу можуть бути використані або певні моменти часу (дати), або окремі періоди (дні, тижні, місяці, квартали, роки тощо). Таким чином, часові ряди за характером часового параметру можуть бути моментні та інтервальні відповідно.

Аналіз часових рядів — сукупність математико-статистичних методів аналізу, призначених для виявлення структури часових рядів і для їх прогнозування. Виявлення структури часового ряду необхідно для того, щоб побудувати математичну модель того явища, яке є джерелом часового ряду.

На даний момент розроблено і обґрунтовано багато різних методів для аналізу часових рядів: методи інтерполяції, двох крайніх точок, середніх групових точок, Холта-Уінтерса, Трігга-Ліча, Чоу, з використанням експертних оцінок тощо. Однак усі вони поділяються на два основні класи: локальні та глобальні. Такий поділ проводиться за областю визначення параметрів апроксимуючої функції, що рекурентно встановлює таке значення часового ряду по кількох попередніх [2].

Історично першими були розроблені глобальні методи, у яких на основі статистичного аналізу пропонувалося використовувати авторегресії, згладжування тощо. Пізніше в рамках нелінійної динаміки були розроблені нові практичні методики:

- сингулярний спектральний аналіз (SSA), який є глобальним методом;

- локальна апроксимація (LA);
- поєднання SSA-LA.

Для реалізації застосунку, який обраховує прогнозування епідемічної ситуації викликаной захворюванням COVID-19 у різних країнах, було обрано перший наведений метод, а саме сингулярний спектральний аналіз.

### *1.2 Сингулярний спектральний аналіз*

Сингулярний спектральний аналіз (у російськомовній літературі можна зустріти назву «Гусениця», а з англійської Singular spectrum analysis, далі – SSA) є непараметричний спектральний метод оцінки. Він поєднує у собі елементи класичного аналізу часових рядів, багатовимірною статистичного аналізу, геометрії багатовимірною, динамічних систем і обробки сигналів [3]. Основа методу полягає у перетворенні одновимірною часового ряду у багатовимірний за допомогою однопараметричної зсувної процедури та дослідження отриманої багатовимірною траєкторії за допомогою сингулярного розкладання та відновлення (апроксимації) часового ряду за вибраним головним компонентом. Метою методу є розробка часового ряду на адитивні складові, що інтерпретуються.

При цьому метод не вимагає стаціонарності часового ряду, що аналізується, знання моделі тренду, а також інформації про наявність у часових рядів періодичних складових та їх періодів. При таких слабких припущеннях метод SSA дозволяє розв'язувати різні завдання, наприклад, виділити тренд, виявити періодику, згладити часовий ряд, побудувати повне розкладання часового ряду у суму тренду, періодик і шума. Важливо, що цей непараметричний метод дозволяє отримати результати, часто лише незначно менш точні, ніж багато параметричні методи, у яких використовуються відомі моделі часових рядів [4].

Варто відзначити, що даний метод у 80-х роках минулого століття активно використовувався для аналізу часових рядів, що характеризують стан різних біологічних і гідрологічних систем. Області, у яких SSA може

застосовуватися дуже широкі: кліматологія, морська наука, геофізика, машинобудування, обробка зображень, медицина, економетрика тощо. Навіть відомі приклади успішного застосування методу SSA в астрономічних задачах [4]. Науковцями були запропоновані різні модифікації та різні методики SSA, що зараз використовуються у практичних застосуваннях, такі як тенденція екстракція, періодичність виявлення, сезонне коригування, згладжування, зменшення шуму. У той же час, незважаючи на активні дослідження даного методу різними авторами, ряд питань, пов'язаних з вибором його параметрів і інтерпретацією отриманих результатів, залишаються недослідженими.

## РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ПРОГНОЗУВАННЯ

Для реалізації програмного застосунку, що має прогнозувати, а також відображати епідемічну ситуацію, що викликана захворюванням COVID-19 у різних країнах, було обрано мову програмування C#, а також модульну платформу для розробки з відкритим вихідним кодом – .NET Core.

У 2018 році компанія Microsoft вперше представила ML.NET – безкоштовний, кросплатформений і відкритий фреймворк машинного навчання. Звіт Microsoft про машинне навчання за допомогою ML.NET продемонстрував, що він здатний навіть тренувати моделі аналізу настроїв, використовуючи великі набори даних, досягаючи високої точності. Результати свідчать про 95% точність на даних 9GB огляду Amazon. Оскільки, однією з можливостей ML.NET є саме реалізація сингулярного спектрального аналізу, було вирішено використати даний фреймворк.

### *2.1 ML.NET та принципи роботи*

Як було сказано раніше, ML.NET – це кросплатформна середовище машинного навчання з відкритим вихідним кодом (Windows, Linux, macOS) для розробників .NET. Вона дозволяє додавати у застосунки .NET

можливості машинного навчання в автономному і підключеному режимах. Використовуючи цю функцію, можна отримувати автоматичні прогнози на основі даних, що доступні застосунку. Додатки машинного навчання використовують для прогнозування закономірностей, знайдених у даних, не будучи явно запрограмованими [5].

У основі ML.NET лежить модель машинного навчання. Ця модель визначає кроки, які необхідно виконати для отримання прогнозів на основі вхідних даних. За допомогою ML.NET можна навчити призначену для користувача модель, вказавши відповідний алгоритм, а також імпортувати попередньо навчені моделі TensorFlow і ONNX.

Створену модель можна додати в застосунок і використовувати її для отримання прогнозів.

ML.NET працює у Windows, Linux і macOS з .NET Core або в Windows з .NET Framework. 64-розрядна версія підтримується на всіх платформах, а 32-розрядна версія підтримується у Windows, за винятком функцій, пов'язаних з TensorFlow, LightGBM та ONNX.

ML.NET пропонує Model Builder (простий інструмент для користувача інтерфейсу) та інтерфейс командного рядка, створені для того, щоб спростити створення призначених для користувача моделей ML з використанням AutoML [6]. За допомогою ML.NET можна отримувати прогнози наступних типів:

- Класифікація/категоризація (автоматичний поділ відгуків клієнтів на позитивні і негативні);
- Регресія/прогноз безперервних значень (прогноз ціни на будинки, залежно від їх розміру і місцезнаходження);
- Виявлення аномалій (виявлення шахрайських банківських операцій);
- Рекомендації (пропозиція продуктів, які онлайн-покупці можуть захотіти купити, на основі їх попередніх покупок);

- Тимчасові ряди/послідовності (прогнози погоди і обсягів продажів);
- Класифікація зображень (класифікація патологій на медичних зображеннях).

## 2.2 Обробка статистичних даних

Одним з важливих етапів роботи над створенням кінцевого застосунку був пошук релевантних даних щодо кількості виявлених хворих на COVID-19, кількість летальних випадків та пацієнтів, що одужали у кожній країні світу. Було обрано набір даних на одному з найпопулярніших порталів у сфері Data Science – Kaggle [7]. Загалом, Kaggle містить більше ніж 19 000 датасетів.

Прямого шляху завантаження набору даних, на жаль, немає. Kaggle дозволяє підключитись до даних через публічний API. Для цього треба бути зареєстрованим користувачем платформи та авторизуватись через API (Authentication Header).

Kaggle дозволяє завантажити потрібний набір даних лише у .zip архіві. Тому потрібно програмно розархівувати файл, як результат буде отриманий .csv файл. На рисунку 2.1 приведений метод DownloadDataFile(), у якому описані вищенаведені кроки отримання даних з платформи Kaggle:

```
public const string DataFileName = "covid_19_clean_complete.csv";

1 reference
public static void DownloadDataFile()
{
    var auth = new { Username = string.Empty, Key = string.Empty };
    auth = JsonConvert.DeserializeObject<File>(File.ReadAllText("kaggle.json"), auth);
    var authToken = Convert.ToBase64String(Encoding.ASCII.GetBytes(string.Format($"{auth.Username}:{auth.Key}", auth)));
    var client = new HttpClient();
    client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", authToken);
    var archiveStreamTask = client.GetStreamAsync("https://www.kaggle.com/api/v1/datasets/download/imdevskp/corona-virus-report/"
        + DataFileName);
    const string archiveName = "covid_19_clean_complete.zip";
    using (var stream = archiveStreamTask.Result)
    using (var output = new FileStream(archiveName, FileMode.OpenOrCreate))
    {
        stream.CopyTo(output);
    }

    using (ZipArchive archive = ZipFile.OpenRead(archiveName))
    {
        archive.Entries.First().ExtractToFile(DataFileName, true);
    }
}
```

Рисунок 2.1 – Метод DownloadDataFile

Наступним етапом роботи було перетворення даних (розпарс) з отриманого .csv файлу. Для цього було використано бібліотеку TinyCsvParser.

Її було використано в обгортці відомого патерну програмування – фабрика, для створення класу (Рисунок 2.2):

```
public class CsvCoronaParserFactory
{
    1 reference
    public static CsvParser<CoronaData> CreateParser()
    {
        var parserOptions = new CsvParserOptions(true, ',');
        var mapper = new CsvCoronaMapper();
        var parser = new CsvParser<CoronaData>(parserOptions, mapper);

        return parser;
    }

    2 references
    private class CsvCoronaMapper : CsvMapping<CoronaData>
    {
        1 reference
        public CsvCoronaMapper()
        {
            MapProperty(1, m => m.Country);
            MapProperty(4, m => m.CoronaDate, new DateTimeConverter("M/d/y"));
            MapProperty(5, m => m.CoronaConfirmed);
            MapProperty(6, m => m.CoronaDeaths);
            MapProperty(7, m => m.CoronaRecovered);
        }
    }
}
```

Рисунок 2.2 – Клас CsvCoronaParserFactory

### 2.3 Визначення функціональних вимог

На даному етапі роботи було визначено наступні вимоги до функціоналу:

1. Оновлення статистичних даних з платформи Kaggle за допомогою її API.
2. Завантаження користувачем статистичних даних у .csv форматі.

3. Відображення на карті світу даних щодо кількості хворих, одужавших та померлих за останній день з набору даних по кожній країні.
4. Вибір країни для подальших обрахунків та прогнозування.
5. Відображення на графіку кількості хворих, одужавших та померлих за увесь час вибраної країни.
6. Можливість відображати лише дані окремо хворих, окремо одужавших, окремо померлих на графіку.
7. Побудова кругової діаграми за вибраний день.
8. Прогноз даних з наступними введеними параметрами: кількість днів та параметр розміру вікна для алгоритму SSA.
9. Тестування прогнозу на основі порівняння з реальними даними з датасету (параметри аналогічні попереднім вимогам: кількість днів та параметр розміру вікна для алгоритму SSA).
10. Відображення реальних, спрогнозованих, а також спрогнозованих для тестування даних на графіку.
11. Обрахунок метрик прогнозування даних.

#### *2.4 Реалізація функціональної частини застосунку*

Згідно вимог до функціональної частини застосунку на мові програмування C# [8], з використанням платформи .NET Core і фреймворку ML.NET було реалізовано ряд методів.

Першим важливим кроком реалізації функціональної частини було взаємодія готових для роботи статистичних даних та моделі ML.NET. Для цього було написано клас `TimeSeriesAnalyzer` з методами підгрузки даних до моделі, а також найважливішими методами створення та тренування моделі за наявними даними, а також прогнозування даних.

Підгрузка даних до моделі була реалізована у конструкторі класу (Рисунок 2.3):



```

public TimeSeriesAnalyzer(CoronaData[] coronaDataArr, int? daysToIgnore = null)
{
    _mlContext = new MLContext();

    if (daysToIgnore.HasValue)
    {
        coronaDataArr = coronaDataArr.Take(coronaDataArr.Length - daysToIgnore.Value).ToArray();
    }

    _dataToTrainLength = coronaDataArr.Length;
    _coronaDataView = _mlContext.Data.LoadFromEnumerable(coronaDataArr);
}

```

Рисунок 2.3 – Конструктор класу TimeSeriesAnalyzer

Створення моделі та її тренування на основі алгоритму сингулярного спектрального аналізу було реалізовано у методі PredictCoronaUnits:

```

public CoronaTimeSeriesPrediction PredictCoronaUnits(int daysToPredict, string columnToAnalyze, int windowSize)
{
    IEstimator<ITransformer> forecastEstimator = _mlContext.Forecasting.ForecastBySsa(
        outputColumnName: nameof(CoronaTimeSeriesPrediction.PredictedCoronaUnits),
        inputColumnName: columnToAnalyze,
        windowSize: windowSize,
        seriesLength: _dataToTrainLength,
        trainSize: _dataToTrainLength,
        horizon: daysToPredict,
        confidenceLowerBoundColumn: nameof(CoronaTimeSeriesPrediction.PredictedCoronaLowerBound),
        confidenceUpperBoundColumn: nameof(CoronaTimeSeriesPrediction.PredictedCoronaUpperBound));

    ITransformer forecastTransformer = forecastEstimator.Fit(_coronaDataView);

    var coronaPredictEngine = forecastTransformer.CreateTimeSeriesEngine<CoronaData, CoronaTimeSeriesPrediction>(_mlContext);

    coronaPredictEngine.CheckPoint(_mlContext, "Model.zip");

    var coronaPrediction = coronaPredictEngine.Predict();
    return coronaPrediction;
}

```

Рисунок 2.4 – Метод PredictCoronaUnits

## 2.5 Реалізація візуальної частини застосунку

Після реалізації усіх функціональних вимог до кінцевого застосунку, було розроблено візуальну частину для відображення даних. Для розробки візуальної частини застосунку було обрано WPF технології [9].

Для того, щоб оновити статистичні дані з платформи Kaggle або завантажити свої статистичні дані у форматі .csv було додано дві кнопки на верхню панель (зображено на рисунку 2.5).

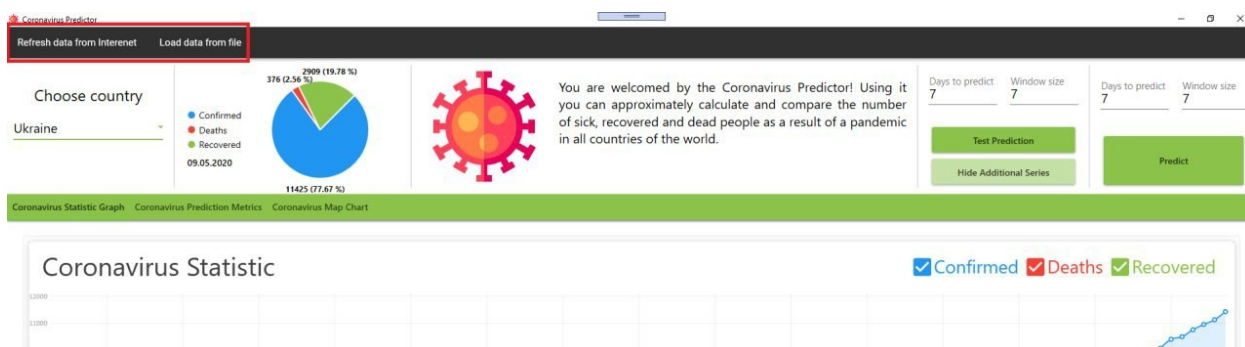


Рисунок 2.5 – UI оновлення та завантаження статистичних даних

При оновленні було додано додаткове модальне вікно для запобігання випадкового натискання ((зображено на рисунку 2.6):



Рисунок 2.6 – Модальне вікно про оновлення статистичних даних

На вкладці Coronavirus Map Chart користувач може побачити відображення на карті світу даних щодо кількості хворих, одужавших та померлих за останній день з набору даних по кожній країні. Так, наприклад, на рисунку 2.7 користувач вибрав фільтр за померлими, та навів мишкою на карті на Україну:

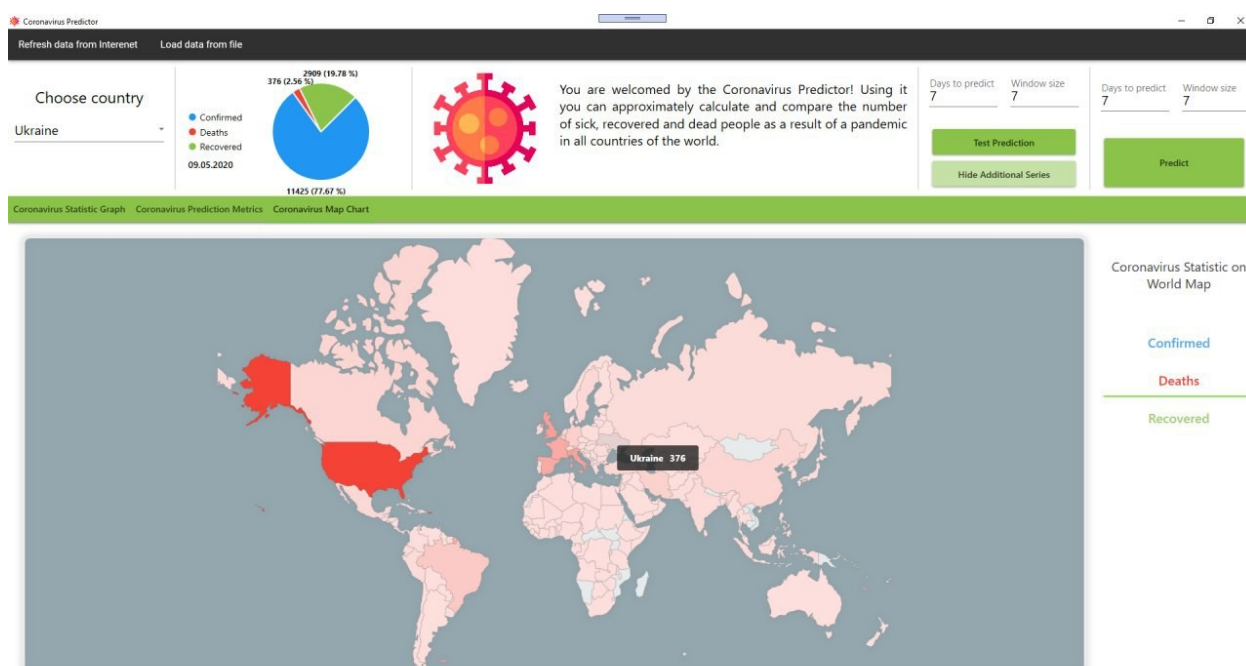


Рисунок 2.7 – Вкладка Coronavirus Map Chart

У верхньому лівому куту користувач може обрати країну з випадального списку, у якому відображаються усі країни з завантаженого або автоматично підгруженого набору даних. Після вибору країни вкладка Coronavirus Statistic Graph автоматично оновлюється: на ній відображаються дані щодо вибраної країни. Також оновлюється кругова діаграма. Аналогічно роботі карти, користувач може фільтрувати дані (показувати лише графік щодо хворих, померлих, одужавших), а також аналогічною є дія наведення мишкою на точку – відображаються дані за вказаний день. Вищеописаний сценарій показаний на Рисунку 2.8:

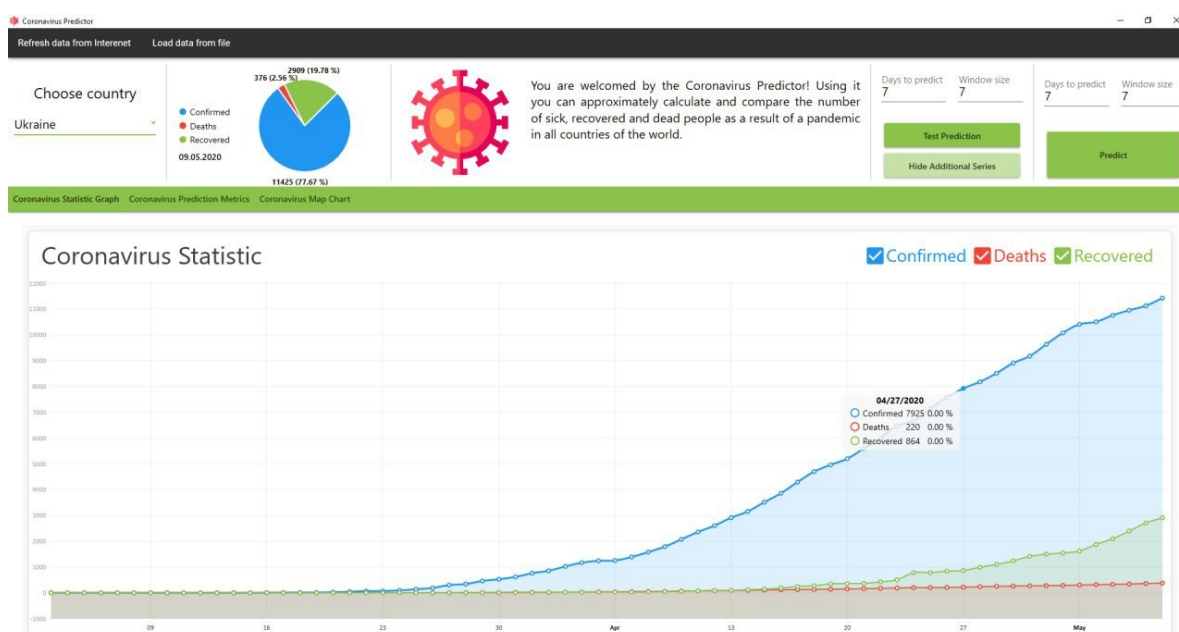


Рисунок 2.8 – Вкладка Coronavirus Statistic Graph

Мабуть, найважливіші функції – прогнозування та тестування отриманих результатів. Для обраної країни користувач має ввести два параметри – кількість днів та параметр розміру вікна для алгоритму SSA, натиснувши кнопку Predict, користувачу відображаються точки фіолетового кольору – це нові спрогнозовані дані. Результат функції прогнозування зображено на рисунку 2.9:

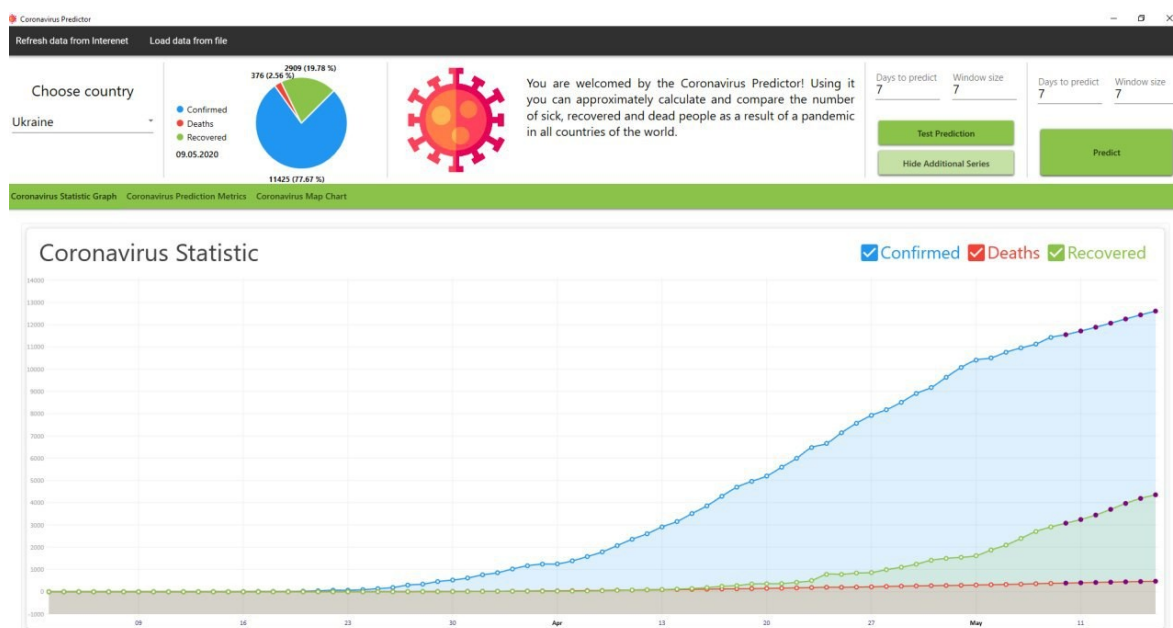


Рисунок 2.9 – Відображення результату прогнозування на графіку

Якщо користувач натискає Test Prediction – модель тренується на даних за винятком останніх  $N$  днів, де  $N$  – число вказане користувачем у полі Days to predict. У якості результату роботи на вказаній частині графіку для тестування прогнозування відображаються реальні дані (Рисунок 2.10):

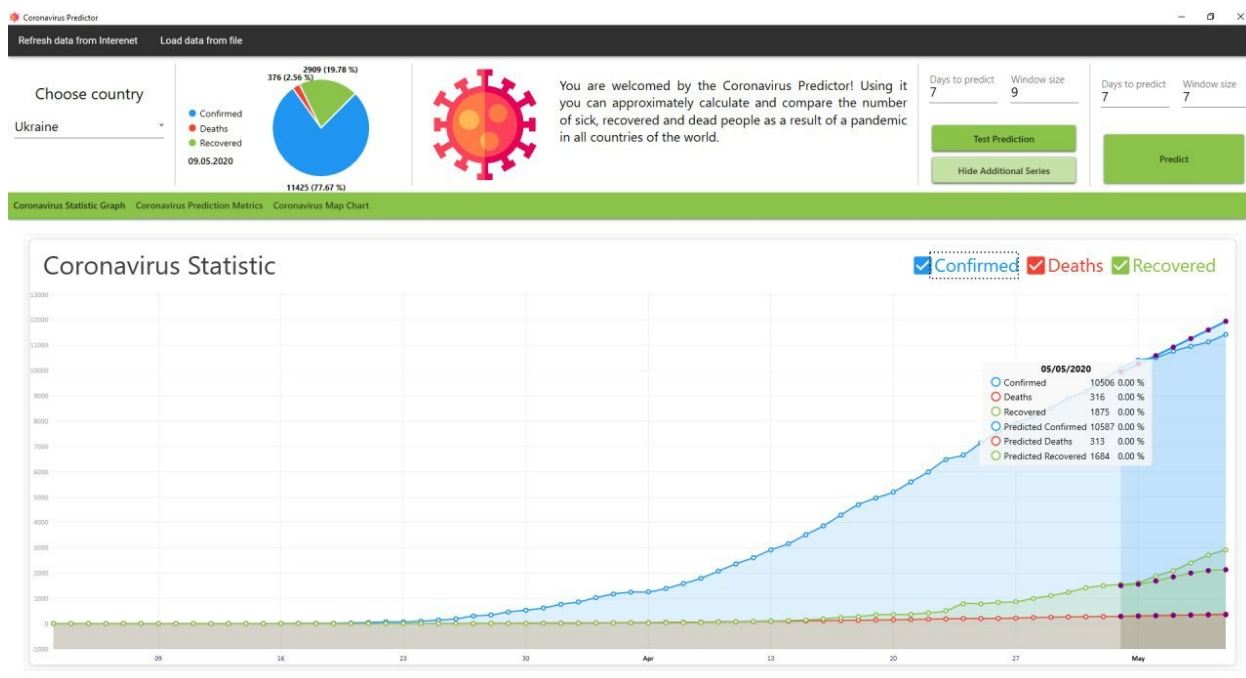


Рисунок 2.10 – Відображення результату тестування прогнозування на графіку

Усі прораховані метрики прогнозування відображаються на вкладці Coronavirus Prediction Metrics. У третьому розділі детально описано, як саме обраховуються дані метрики з математичної точки зору, а також опис реалізації. На даній вкладці відображаються метрики для окремо прорахованих категорій прогнозування: кількості померлих, одужавших та хворих (зображено на рисунку 2.11).

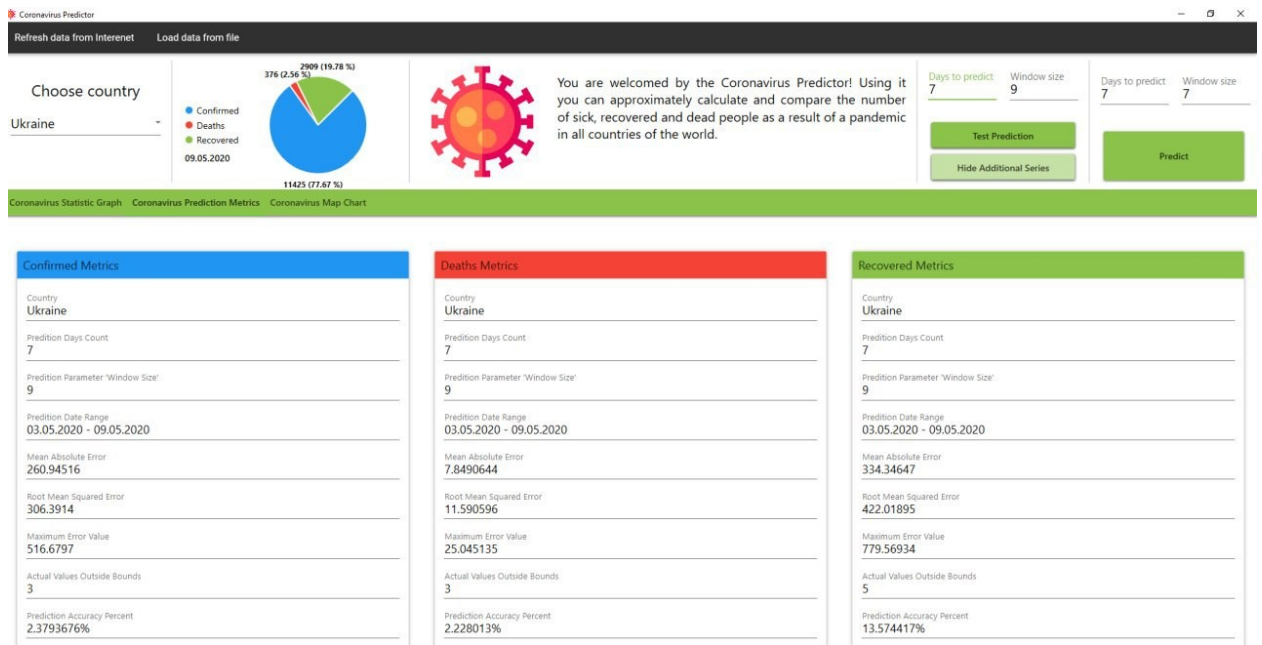


Рисунок 2.11 – Вкладка Coronavirus Prediction Metrics

## РОЗДІЛ 3. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

### 3.1 Основні характеристики прогнозування

При роботі над прогнозуванням часових рядів зазвичай проводиться розробка кількох можливих статистичних моделей досліджуваного процесу, а вже з них вибирається найбільш обґрунтована і відповідна ситуації модель. Яким чином можна оцінити розроблену модель? Існують спеціальні метрики для оцінки прогнозування. Найбільш популярними та часто використовуваними серед них є помилка прогнозування, середня процентна помилка прогнозування. Розглянемо ті з них, завдяки яким після розробки власної моделі ми зможемо оцінити отримані прогнози.

Помилка прогнозування — це величина, яка показує, як сильно прогнозоване значення відхилилося від фактичного. Вона використовується для розрахунку точності прогнозування, що у свою чергу допомагає нам оцінювати як точно і коректно ми сформуваємо прогноз. Розрахувати помилку прогнозування можна за формулою 3.1:

$$e_y = y_j - \hat{y}_j, \quad (3.1)$$

де  $y_j$  – фактичне значення змінної для  $j$ -го виміру, а  $\hat{y}_j$  – прогнозне значення досліджуваної змінної.

Важливою характеристикою оцінки прогнозування є середня відсоткова помилка прогнозування. Позначимо дану величину *MAE* як скорочення від англ. the mean absolute error. Розрахувати її значення можна за наступною формулою (наведено у формулі 3.2):

$$MAE = \frac{\sum_{j=1}^N |y_j - \hat{y}_j|}{N}, \quad (3.2)$$

де  $y_j$  – фактичне значення змінної для  $j$ -го виміру, а  $\hat{y}_j$  – прогнозне значення досліджуваної змінної,  $N$  – розмір певної вибірки вимірювань.

Точність прогнозування є поняттям прямо оберненим до помилки прогнозування. Якщо помилка прогнозування велика, то точність мала і навпаки, якщо помилка прогнозування мала, то точність велика. Говорячи про високу точність, ми завжди говоримо про низьку помилку прогнозу і в цій області непорозуміння бути не повинно [10]. Важко знайти матеріали про прогнозування, у яких наведені оцінки саме точності прогнозу, хоча з точки зору людського сприйняття коректніше говорити саме про високу точність. Мабуть, тому у рекламних джерелах інформації завжди буде сказано про високу точність.

Ще однією загальноживаною метрикою є середнє квадратичне відхилення (з англ. the root mean square error, далі RMSE). Дана величина обраховується за формулою 3.3:

$$RMSE = \sqrt{\frac{\sum_{j=1}^N (y_j - \hat{y}_j)^2}{N}}, \quad (3.3)$$

де  $y_j$  – фактичне значення змінної для  $j$ -го виміру, а  $\hat{y}_j$  – прогнозне значення досліджуваної змінної,  $N$  – розмір певної вибірки вимірювань.

Ще одним поширеним показником середніх помилок прогнозування є процентна середня абсолютна помилка (з англ. the mean absolute percentage error). Оскільки даний показник ми будемо також обраховувати для оцінки



реалізації власного прогнозування, тому нижче наведемо формулу її обчислення (формула 3.4):

$$MAPE = \frac{1}{N} \sum_{j=1}^N \frac{|y_j - \hat{y}_j|}{y_j} * 100\%, \quad (3.4)$$

де  $y_j$  – фактичне значення змінної для  $j$ -го виміру, а  $\hat{y}_j$  – прогнозне значення досліджуваної змінної,  $N$  – розмір певної вибірки вимірювань.

Також, у розрізі даної роботи було введено поняття кількості значень поза прогнозованими межами (з англ. the actual values outside the predicted limits). Воно означає кількість днів, для яких фактичне значення не знаходиться у спрогнозованих границях.

### 3.2 Розрахунок основних метрик отриманого прогнозу

Для оцінки прогнозу був створений окремий клас PredictionMetrics, що містить у собі змінні та метод CalculateMetrics для обрахування наступних метрик (детально описані у попередньому пункті даного розділу):

- середня відсоткова помилка прогнозування (MAE);
- середнє квадратичне відхилення (RMSE);
- кількість значень поза прогнозованими межами (AVOP);
- максимальна помилка;
- процентна середня абсолютна помилка (MAPE).

На рисунку 3.1 наведено метод CalculateMetrics, що належить класу PredictionMetrics та використовується для обчислення метрик оцінки прогнозування:



```

public static PredictionMetrics CalculateMetrics(float[] actualValues, CoronaTimeSeriesPrediction prediction, string country,
int windowSize, DateTime firstActualValueDate)
{
    var metrics = new PredictionMetrics
    {
        Country = country,
        PredictionDaysCount = actualValues.Length,
        PredictionParameterWindowSize = windowSize,
        StartPredictionDate = firstActualValueDate,
        EndPredictionDate = firstActualValueDate.AddDays(actualValues.Length - 1)
    };

    var errorForMetrics = actualValues.Zip(prediction.PredictedCoronaUnits,
        (actualValue, forecastValue) => actualValue - forecastValue).ToArray();

    metrics.MAE = errorForMetrics.Average(error => Math.Abs(error));
    metrics.RMSE = (float)Math.Sqrt(errorForMetrics.Average(error => Math.Pow(error, 2)));
    metrics.MaxError = errorForMetrics.Max(error => Math.Abs(error));

    var percentageDifference = new List<float>(actualValues.Length);
    for (int i = 0; i < actualValues.Length; i++)
    {
        percentageDifference.Add(Math.Abs(errorForMetrics[i] * 100 / actualValues[i]));

        if (actualValues[i] < prediction.PredictedCoronaLowerBound[i] ||
            actualValues[i] > prediction.PredictedCoronaUpperBound[i])
        {
            metrics.ActualValuesOutsideBounds++;
        }
    }

    metrics.PredictionErrorPercent = percentageDifference.Average();
    return metrics;
}

```

Рисунок 3.1 – Метод CalculateMetrics

Головною ознакою алгоритму SSA є наявність коефіцієнту розміру вікна. Умови відокремлюваності даних дають рекомендації щодо вибору розміру вікна  $L$ : він повинен бути достатньо великим ( $L \sim N/2$ , де  $N$  – розмір певної вибірки вимірювань). А якщо ми хочемо отримати періодичну складову з відомим періодом, то розміри вікон, які поділяються на період націло, забезпечують кращу відокремлюваність. Алгоритм SSA з малим  $L$  частіше забезпечує більше згладжування спрогнозованих даних. Як правило, вибір розміру вікна важливий, але результат стабільний і при невеликих змінах значення  $L$  [11].

У реалізації готового застосунку був написаний алгоритм знаходження значення розміру вікна, при якому досягається найкращі результати прогнозування. Код розрахунку найкращого вікна приведений у додатку А.

Отже, до виконання тестування прогнозування (кнопка Calculate best widnow sizes) на заданих користувачем параметрах відкривається вікно з наступними обчисленими за допомогою власного алгоритму даними:

- значення розміру вікна при якому досягається найближчий до фактичних результатів прогноз для кількості хворих;
- значення розміру вікна при якому досягається найближчий до фактичних результатів прогноз для кількості померлих;
- значення розміру вікна при якому досягається найближчий до фактичних результатів прогноз для кількості одужавших.

На рисунку 3.2 зображено приклад такого вікна для України на 7 днів:

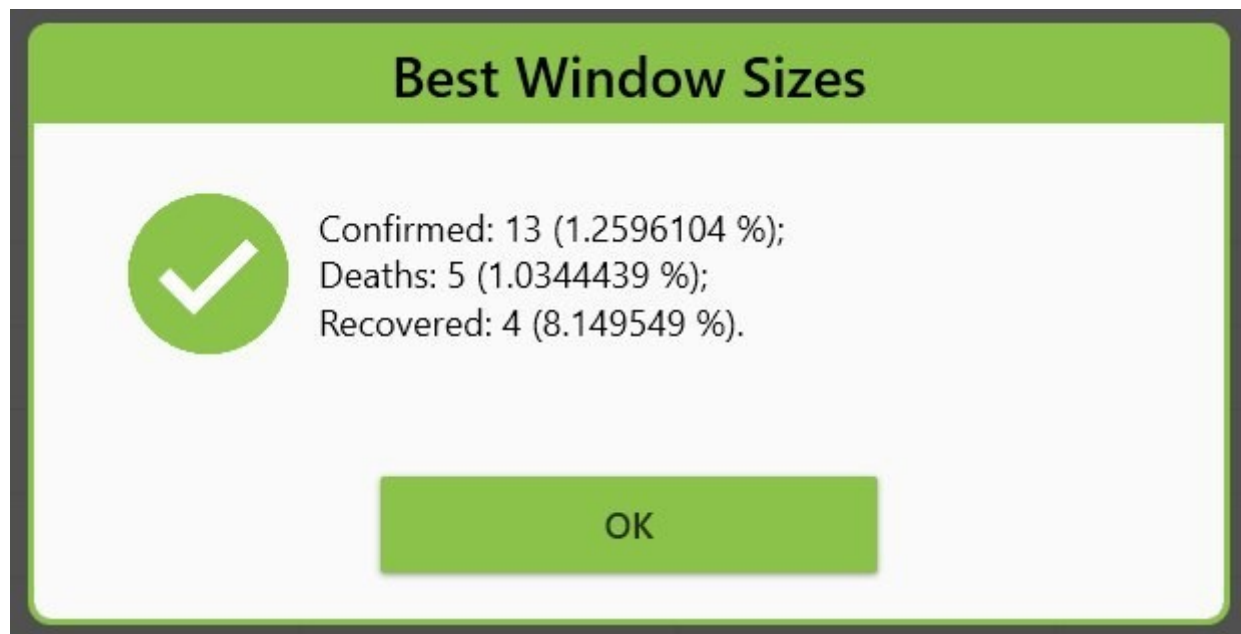


Рисунок 3.2 – Результат роботи алгоритму пошуку значення розміру вікна

### 3.3 Тестування застосунку

На даному етапі роботи було проведено тестування на різних країнах. Нижче приведемо оцінки для таких країн як Україна, Китай, США. Обґрунтовуючи вибір саме цих країн, можна сказати наступне: згідно джерелу Worldometers, що регулярно оновлює статистику зареєстрованих випадків на коронавірусну інфекцію у світі, у Китаї станом на 10 травня діагностували вірус лише у однієї людини. Загалом, ситуацію у Китаї можна описати як поступове відходження від карантинного режиму, що принципово відрізняється від ситуації у Сполучених Штатах Америки. Ще наприкінці

березня США вийшли на перше місце у світі за кількістю заражених, обігнавши Італію, Іспанію та Китай. З тих пір відрив тільки збільшується – станом на 9 травня в Сполучених Штатах було більше 1 млн хворих і понад 78 тисяч померлих.

Отже, спочатку застосунок був протестований на даних про Україну, на таблицях 3.1 – 3.3 показані результати обчислення метрик оцінки прогнозування на 3 періоди: 7, 14 та 21 день:

*Таблиця 3.1 – Оцінка прогнозу для України на 7 днів (03.05.2020 - 09.05.2020)*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	13	5	4
Середня помилка	135.06389	3.476846	187.50938
Середньоквадратичне відхилення	159.67453	4.4125595	230.72154
Максимальна помилка	231.27051	8.64679	412.40918
Кількість значень поза прогнозованими межами	2	0	2
Процент помилки прогнозу	1.2596104%	1.0344439%	8.149549%

*Таблиця 3.2 – Оцінка прогнозу для України на 14 днів 26.04.2020 - 09.05.2020*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	5	8	13
Середня помилка	155.21317	4.4264884	86.27555
Середньоквадратичне відхилення	194.53049	6.781962	110.394295
Максимальна помилка	413.62305	17.135315	258.84692
Кількість значень поза прогнозованими межами	1	2	0
Процент помилки прогнозу	1.6019118%	1.4954331%	5.4170346%

*Таблиця 3.3 – Оцінка прогнозу для України на 21 день 19.04.2020 - 09.05.2020*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	7	14	7
Середня помилка	106.77923	11.313216	750.15576
Середньоквадратичне відхилення	139.33223	14.608998	977.07086
Максимальна помилка	280.23242	35.428864	2185.4568

Кількість значень поза прогнозованими межами	1	0	21
Процент помилки прогнозу	1.293293%	3.9734206%	48.551502%

Аналогічно, були проведені оцінки прогнозування для США (на 7, 14 та 21 день). Результати можна побачити на таблицях 3.4 -3.6:

*Таблиця 3.4 – Оцінка прогнозу для США на 7 днів (03.05.2020 - 09.05.2020)*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	3	18	46
Середня помилка	2681.366	130.18973	9762.984
Середньоквадратичне відхилення	3266.5452	150.37303	11116.676
Максимальна помилка	4677.1875	231.6875	16075.281
Кількість значень поза прогнозованими межами	0	0	0
Процент помилки прогнозу	0.27616414%	0.17966837%	4.982132%

*Таблиця 3.5 – Оцінка прогнозу для США на 14 днів (26.04.2020 - 09.05.2020)*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	28	13	5
Середня помилка	8238.696	341.8859	9137.05
Середньоквадратичне відхилення	12008.038	400.97995	10690.84
Максимальна помилка	25366	676.08594	18068.969
Кількість значень поза прогнозованими межами	0	2	7
Процент помилки прогнозу	0.9218213%	0.51940787%	5.8859706%

*Таблиця 3.6 – Оцінка прогнозу для США на 21 день (19.04.2020 - 09.05.2020)*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	20	23	33
Середня помилка	22578.63	1515.7855	11338.111
Середньоквадратичне відхилення	29544.168	1685.3357	15310.484
Максимальна помилка	65501	2638.5625	30065.953
Кількість значень поза прогнозованими межами	0	1	4
Процент помилки прогнозу	2.4877393%	2.590318%	7.005208%

Нижче приведені аналогічні дані для Китаю (зображено на Таблицях 3.7 - 3.9):

*Таблиця 3.7 – Оцінка прогнозу для Китаю на 7 днів 03.05.2020 - 09.05.2020*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	50	3	12
Середня помилка	59.990314	10.791504	48.391743
Середньоквадратичне відхилення	73.67109	11.886172	56.006077
Максимальна помилка	157.24704	18.265137	88.765625
Кількість значень поза прогнозованими межами	1	0	0
Процент помилки прогнозу	18.746439%	0.23272598%	0.06132139%

*Таблиця 3.8 – Оцінка прогнозу для Китаю на 14 днів (26.04.2020 - 09.05.2020)*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	12	8	5
Середня помилка	36.447945	19.36105	32.13672
Середньоквадратичне відхилення	47.57471	20.412043	38.845207
Максимальна помилка	111.63818	29.018555	79.734375
Кількість значень поза прогнозованими межами	0	0	0
Процент помилки прогнозу	5.423622%	0.41753393%	0.040888093%

*Таблиця 3.9 – Оцінка прогнозу для Китаю на 21 день (19.04.2020 - 09.05.2020)*

	Хворі	Померлі	Одужавші
Параметр «розмір вікна»	12	2	38
Середня помилка	52.241753	39.267414	465.10678
Середньоквадратичне відхилення	64.34353	39.270245	552.4376
Максимальна помилка	141.52686	39.934082	1091.1094
Кількість значень поза прогнозованими межами	0	0	0
Процент помилки прогнозу	6.5289917%	0.84689%	0.5938319%

Отже, можна зробити наступні висновки. Застосований алгоритм надає досить точні результати прогнозування. У більшості випадків при великому

початковому наборі даних (тобто часовий ряд має багато даних, а також немає великих неперіодичних стрибків) помилка прогнозування не перевищує 5%.

Також можна помітити, що велику роль відіграє параметр розміру вікна. Більше того, його не можна підібрати одразу для різних наборів даних: для кожної категорії даних (кількість хворих, померлих, одужавших), а також періоду прогнозування, варто підібрати окреме значення параметру розміру вікна.

## ВИСНОВКИ

За свою історію людство не раз стикалося з епідемічними цунами, що забирало від декількох тисяч до мільйона людей. У 1918, 1957, 1968 (гонконгський грип), поширення лихоманки Ебола у 2003-2008 роках та "свинячого грипу" у 2009-2010-му – це далеко не повний список за останні 120 років.

Майже півроку пройшло з моменту, коли у китайському місті Ухань зафіксували спалах нової коронавірусної інфекції. За цей час вірус поширився по всьому світу, що призвело до порушення звичного життя суспільства, збоїв в економіці, а системи охорони здоров'я багатьох країн працюють на межі своїх можливостей.

Кількість інфікованих все ще стрімко зростає, що загрожує потенційно катастрофічними наслідками для здоров'я населення і глобальної економіки. Деякі країни ще не мали досвіду боротьби з великими спалахами інфекційних захворювань. Тим не менш, урядам, що зіткнулися з пандемією, треба було приймати рішення та діяти швидко. Так, за оцінками аналітиків BCG, заходи, які застосовували різні країни для зниження поширення інфекції, показали різну ефективність. Наприклад, найпродуктивнішим виявилось обмеження міжнародного сполучення (37%).

При прийнятті рішень потрібно переконатися у тому, що вони приведуть до очікуваного результату у майбутньому. Тому для розробки рішення найчастіше потрібно прогнозування. Загалом, прогнозування тісно пов'язане з плануванням і використовується для ефективного прийняття рішень.

Кількість хворих, одужавших та померлих людей кожного дня з початку епідемії з математичної точки зору можна розглянути як часовий ряд. Тому, доцільним та актуальним є використання аналізу часового ряду для прогнозування епідемічної ситуації викликаного захворюванням, у наших реаліях саме COVID-19 у різних країнах, у тому числі і Україні.

Методи аналізу часових рядів безперервно збагачуються і удосконалюються, оскільки постійно застосовуються у практичних задачах людства. Так, наприклад, одним з популярних методів є сингулярний спектральний аналіз. Саме завдяки ньому, у даній роботі було реалізовано прогнозування епідемічної ситуації.

Загалом, у даній роботі було:

- ✓ опрацьовано літературу, присвячену часовим рядам, та методам їх аналізу;
- ✓ знайдено релевантні дані щодо кількості хворих на коронавірусну інфекцію для подальшої роботи з ними;
- ✓ визначено вимоги до застосунку;
- ✓ розроблено модель для прогнозування епідемічної ситуації на основі сингулярного спектрального аналізу;
- ✓ розроблено функціональну та візуальну частини застосунку згідно до визначених вимог на мові програмування C# з використанням фреймворку ML.NET;
- ✓ протестовано готовий застосунок на різних країнах.



## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Hamilton, J.* Time Series Analysis. Princeton University Press. 1994. 820 p.
2. *Brockwell P., Davis R.* Introduction to Time Series and Forecasting (Springer Texts in Statistics). Springer, 2016. 425 p.
3. Голяндіна, Н. Метод "Гусеница"-SAA: анализ временных рядов: Учебное пособие. С.-Петербургский державний університет, 2004. 76 p.
4. *Golyandina, N., Korobeynikov, A., Zhigljavsky, A.* Basic Singular Spectrum Analysis and Forecasting with R. Springer; 1st ed, 2018. 288 p.
5. *Capellman, J.* Hands-On Machine Learning with ML.NET: Getting started with Microsoft ML.NET to implement popular machine learning algorithms in C#. Packt Publishing, 2020. 296 p.
6. *Price, M.* C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code, 4th Edition. Packt Publishing, 2019. 818 p.
7. [https://www.kaggle.com/imdevskp/corona-virus-report#covid\\_19\\_clean\\_complete.csv](https://www.kaggle.com/imdevskp/corona-virus-report#covid_19_clean_complete.csv) [Інтернет ресурс]
8. *Griffiths I.* Programming C# 8.0: Build Cloud, Web, and Desktop Applications. O'Reilly Media, 2019. 802 p.
9. *Nathan A.* WPF 4.5 Unleashed. Sams Publishing, 2013. 864 p.
10. *Nielsen, A.* Practical Time Series Analysis: Prediction with Statistics and Machine Learning. O'Reilly Media, 2019. 504 p.
11. *Golyandina, N.* On the choice of parameters in singular spectrum analysis and related subspace-based methods. Stat. Interface 3, 2010. 259-279 pp.

## ДОДАТОК А

```

public static class CoronaHelper
{
    public static void CalculateBestWidnowSizes(CoronaData[] coronaDataArr,
int daysToTestPrediction)
    {
        if (daysToTestPrediction < 1)
        {
            CustomMessageBox.Show($"Days to test prediction must be positive
value",
                                "Parameters validation error", MessageBoxButton.OK,
MessageBoxImage.Error);
            return;
        }
        var windowSizeLimit = (coronaDataArr.Length - daysToTestPrediction) /
2;
        if (windowSizeLimit < 3)
        {
            CustomMessageBox.Show($"Can not calculate best window sizes for
{daysToTestPrediction} days to test prediction and {coronaDataArr.Length} amount
of days",
                                "Parameters validation error", MessageBoxButton.OK,
MessageBoxImage.Error);
            return;
        }

        var country = coronaDataArr.First().Country;
        var analyzer = new TimeSeriesAnalyzer(coronaDataArr,
daysToTestPrediction);
        int bestWindowConfirmed = -1, bestWindowDeaths = -1,
bestWindowRecovered = -1;
        float errorConfirmed = float.MaxValue, errorDeaths = float.MaxValue,
errorRecovered = float.MaxValue;

        for (int windowSize = 2; windowSize < windowSizeLimit; windowSize++)
        {
            var confirmedPrediction =
analyzer.PredictCoronaUnits(daysToTestPrediction,
nameof(CoronaData.CoronaConfirmed), windowSize);
            var deathsPrediction =
analyzer.PredictCoronaUnits(daysToTestPrediction, nameof(CoronaData.CoronaDeaths),
windowSize);
            var recoveredPrediction =
analyzer.PredictCoronaUnits(daysToTestPrediction,
nameof(CoronaData.CoronaRecovered), windowSize);

            var actualCoronaValues =
coronaDataArr.TakeLast(daysToTestPrediction);
            var startDate = actualCoronaValues.First().CoronaDate;
            var confirmedPredictionMetrics =
PredictionMetrics.CalculateMetrics(actualCoronaValues.Select(c =>
c.CoronaConfirmed).ToArray(), confirmedPrediction, country, windowSize,
startDate);
            var deathsPredictionMetrics =
PredictionMetrics.CalculateMetrics(actualCoronaValues.Select(c =>
c.CoronaDeaths).ToArray(), deathsPrediction, country, windowSize, startDate);
            var recoveredPredictionMetrics =
PredictionMetrics.CalculateMetrics(actualCoronaValues.Select(c =>

```

```

c.CoronaRecovered).ToArray(), recoveredPrediction, country, windowSize,
startDate);

        GetBestSizeAndError(confirmedPredictionMetrics, windowSize, ref
errorConfirmed, ref bestWindowConfirmed);
        GetBestSizeAndError(deathsPredictionMetrics, windowSize, ref
errorDeaths, ref bestWindowDeaths);
        GetBestSizeAndError(recoveredPredictionMetrics, windowSize, ref
errorRecovered, ref bestWindowRecovered);
    }

    CustomMessageBox.Show($"Confirmed: {bestWindowConfirmed}
({errorConfirmed} %);" +
        $"Deaths: {bestWindowDeaths} ({errorDeaths} %);" +
        $"Recovered: {bestWindowRecovered} ({errorRecovered} %).",
        "Best Window Sizes", MessageBoxButtons.OK, MessageImage.Success);
}

private static void GetBestSizeAndError(PredictionMetrics
predictionMetrics, int windowSize, ref float error, ref int bestWindowSize)
{
    if (predictionMetrics.PredictionErrorPercent < error)
    {
        error = predictionMetrics.PredictionErrorPercent;
        bestWindowSize = windowSize;
    }
}
}

```