МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розроблення програмного забезпечення для прогнозування об'ємів продажу інтернет-магазину взуття

Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-1-22
Данило ЛЕВКОВЕЦЬ
Науковий керівник:
к.т.н., доц. Тетяна ДЕМКІВСЬКА
Рецензент:
д.т.н., проф. Віктор Чупринка

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

_____Володимир ЩЕРБАНЬ

«___»_____2023_року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Левковцю Данилу Сергійовичу

1. Тема кваліфікаційної роботи: Розроблення програмного забезпечення для прогнозування об'ємів продажу інтернет-магазину взуття.

Науковий керівник роботи: Демківська Тетяна Іванівна, к.т.н., доц.

затверджені наказом закладу вищої освіти від 12.09.2023 року, № 210-уч.

2. Вихідні дані до кваліфікаційної роботи: Розробка кафедри комп'ютерних наук, рекомендована література.

3. Зміст кваліфікаційної роботи Вступ; РОЗДІЛ 1. Теорія часових рядів; РОЗДІЛ 2. Алгоритм прогнозування об'ємів продажу для інтернет-магазинів взуття; РОЗДІЛ 3.Вибір середовища та створення програмного забезпечення. Висновки; Список використаних джерел; Додаток А. Програмний код; Додаток Б. Публікація; Додаток В. Презентація.

4. Дата видачі завдання 01.08.2023р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	01.09.2023	
2	Розділ 1. Теорія часових рядів	07.09.2023	
3	Розділ 2. Алгоритм прогнозування об'ємів продажу для інтернет-магазинів взуття	18.09.2023	
4	Розділ 3. Вибір середовища та створення програмного забезпечення. Додатки – Повний код модулів програми	20.10.2023	
5	Висновки	27.10.2023	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	06.11.2023	
7	Подача кваліфікаційної роботи (проєкту) науковому керівнику для відгуку (за 14 днів дозахисту)	0811.2023	
8	Подача кваліфікаційної роботи (проєкту) для рецензування	08.11.2023	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату	09.11.2023	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри	10.11.2023	

З завданням ознайомлений:

Студент _____ Данило ЛЕВКОВЕЦЬ

Науковий керівник _____ Тетяна ДЕМКІВСЬКА

АНОТАЦІЯ

Левковець Д.С. Розроблення програмного забезпечення для прогнозування об’ємів продажу інтернет-магазину взуття.

Кваліфікаційна магістерська робота за спеціальністю 122 - «Комп’ютерні науки та технології». – Київський національний університет технологій та дизайну, Київ, 2023 рік.

В даній роботі було розроблено програмне забезпечення для прогнозування об’єму продажу товару інтернет-магазинів взуття, на базі адитивної або мультиплікативної моделей часового ряду. Програма виконує розрахунки для двох моделей, по отриманим даним дає можливість обрати найкращу та виконати проноз даних.

В ході даної роботи було розроблено додаток, який буде доступним для всіх популярних та широко використовуваних настільних та планшетних комп’ютерів, підтримуючих платформу .NET Framework.

Ключові слова: часові ряди, адитивна модель, мультиплікативна модель, тренд, прогнозування, .NET Framework, C#, Visual Studio.

ANNOTATION

Levkovets D. S. Development of software for forecasting sales values of an online shoes store.

Master's qualification thesis in specialty 122 - "Computer Science". - Kyiv National University of Technology and Design, Kyiv, 2022.

The project work involved the development of software for forecasting the sales volume of footwear in online stores based on additive or multiplicative time series models. The program performs calculations for both models, allowing users to choose the best one based on the input data and generate sales forecasts.

In the course of this work, an application was developed that will be available for all popular and widely used desktop and tablet computers that support the .NET Framework platform.

Keywords: time series, additive model, multiplicative model, trend, forecasting, .NET Framework, C#, Visual Studio.

ЗМІСТ

ВСТУП

Актуальність. Прогнозування часових рядів залишається дуже актуальною та важливою задачею на сучасний день. Застосування методів аналізу часових рядів в різних сферах індустрії надає значний потенціал для вдосконалення стратегій прийняття рішень та оптимізації діяльності в реальному часі. Серед областей, в яких досі використовують прогнозування часових рядів є: сфера фінансів, економіка, метеорологія, медицина, виробництво та логістика та соціальні науки.

На відміну від сучасних застосувань, в яких можливість прогнозу реалізована як допоміжна функція, розроблене програмне забезпечення виконує прогнозування, як свою головну ціль. Додаток демонструє користувачу модель даних в вигляді таблиць та графіків, що не завжди можливо в інших програмних засобах.

Мета дослідження. Метою дослідження проєкту є розробка програмного забезпечення для прогнозування об’єму продажу товару інтернет-магазинів взуття, на базі адитивної або мультиплікативної моделей часового ряду. Програма повинна виконати розрахунки для двох моделей, по отриманим даним обрати найкращу та виконати проноз даних.

Об’єктом дослідження цієї роботи є алгоритм прогнозування об’ємів продажу та можливості його програмної реалізації.

Предметом дослідження є розробка додатку в середовищі Microsoft Visual Studio, що виконує алгоритм прогнозу об’ємів продажу, та демонструє користувачу моделі даних за якими здійснюється прогноз.

Елементи наукової новизни. Виконання прогнозу об’ємів продажу для магазинів взуття за допомогою програмного забезпечення, відображення моделі прогнозу елементами графічного інтерфейсу та генерація звіту.

Практична значущість роботи. Виконані за допомогою додатку, прогнози впливають на стратегії прийняття рішень та призводять до оптимізації діяльності в реальному часі.

Апробація результатів роботи. Додаток має сучасний та зрозумілий графічний інтерфейс і всі необхідні елементи для виконання своїх функцій.

РОЗДІЛ 1. ТЕОРІЯ ЧАСОВИХ РЯДІВ

1.1. Основні поняття часового ряду

Часовий ряд - це послідовність значень статистичного показника (ознаки), впорядковану у хронологічному порядку, або послідовність даних, зібраних, виміряних або зафіксованих у регулярні проміжки часу. Ці дані можуть представляти зміни в якій-небудь величині, що спостерігається чи вимірюється впродовж часу. Використовують також терміни “ряд динаміки”, “динамічний ряд”, або “time series”.

Прикладами часових рядів є температура повітря в середині кожної доби, щорічний врожай зернових культур, вартість акцій підприємства, рівень інфляції, обмінний курс, продажі в роздрібній торгівлі, кількість відвідувань веб-сайту, виробництво та витрати електроенергії, рівень безробіття тощо.

Математично часовий ряд має вигляд:

$y_{t,}\ t = 1,2,3 \dots, n;$

Де: *t*- рівновіддалені моменти спостережень (час), *y*-конкретне значення показника (рівень ряду).

Окремі спостереження часового ряду називають його рівнями, або елементами. Кожний рівень ряду відповідає певному моменту часу.

Для вивчення часового ряду $y_{t_1}, y_{t_2}, y_{t_3}, \dots, y_{t_n}$ важливий порядок послідовності $t_1, t_2, t_3, \dots, t_n$ оскільки час є визначальним фактором. Це відрізняє часовий ряд від звичайної випадкової вибірки, де індекси використовуються лише для зручності ідентифікації. Характерні особливості часового ряду включають:

- по-перше, значення часового ряду не є незалежними. Якщо майбутні значення змінної можна передбачити, то вони є функцією минулих значень цієї змінної.

- по-друге, значення часового ряду розподілені неоднаково. Закон розподілу ймовірностей цих випадкових величин, зокрема, їхні математичні очікування та дисперсії можуть змінюватися з часом.

Отже, властивості та правила статистичного аналізу випадкових вибірок не можна застосовувати до часових рядів. Порушення умови незалежності між спостереженнями може призвести до негативних наслідків при застосуванні цих методів. В кінці 1980-х - на початку 1990-х років дослідники встановили, що лише врахування часової структури даних про реальні економічні процеси дозволяє адекватно відображати їх у економіко-математичних моделях. Це усвідомлення спричинило перегляд багатьох макроекономічних теорій і сприяло розвитку специфічних методів аналізу таких даних, що отримали назву аналіз часових рядів.

Кожен рівень часового ряду формується під впливом великого числа факторів, основні з яких:

1) фактори, що формують тенденцію ряду (тренд);

2) фактори, що формують циклічні (сезонні) коливання ряду (ефект в динаміці ряду, який повторюється через певний період);

3) випадкові фактори.

Ряди можуть буди одновимірними або багатовимірними. Одновимірним часовий ряд називають, якщо кожному моменту часу відповідають значення лише одного показника. Багатовимірним – якщо кожному моменту часу відповідають значення декількох показників.

Основним завданням дослідження динамічних рядів є:

- відокремлення та опис основних характерних особливостей ряду; підбір статистичної моделі, що найкращим у певному розумінні способом відображає ряд;
- прогнозування майбутніх значень показників, що утворюють ряд, за попередніми спостереженнями;
- підготовка рекомендацій з управління процесом, що породжує досліджуваний часовий ряд.
- аналіз часових рядів, як правило, передбачає проведення таких основних етапів:
- графічне подання й попередній аналіз поведінки часового ряду;

- відокремлення і видалення низько- та високочастотних складових (фільтрація); - дослідження випадкової складової часового ряду, що залишилася після видалення вищезазначених компонент;
- відокремлення і видалення закономірних складових ряду (тренду, сезонних та циклічних компонент);
- побудова загальної моделі досліджуваного ряду;
- побудова і перевірка адекватності моделі випадкової складової;
- дослідження отриманої моделі і прогнозування майбутньої поведінки об’єкта що вивчається;
- вивчення взаємодії між різними часовими рядами, що характеризують певну систему або процес.

1.2. Основні методи дослідження часових рядів

При дослідженні часових рядів застосовують основні методи:

1. **Кореляційний аналіз**, який дає змогу виявляти істотні періодичні залежності та їх затримки (лаги) всередині певного процесу (автокореляція) або між декількома процесами (кроскореляція).

2. **Спектральний аналіз**, що застосовують для визначення періодичних та квазіперіодичних компонент часового ряду.

3. **Методи згладжування та фільтрації**, призначені для перетворення часових рядів з метою видалення з них високочастотних та сезонних коливань.

4. **Методи авторегресії та ковзних середніх**, які використовують для опису і прогнозування процесів, що здійснюють випадкові коливання навколо певного середнього значення.

5. **Методи прогнозування**, що дають можливість на основі обраної моделі часового ряду оцінювати його найбільш імовірні значення в майбутньому.

Залежно від характеру часового параметра можна виділити моментні та інтервальні часові ряди. У моментних рядах значення показника характеризують стан на певний момент часу, відповідаючи змінним типу запасу (stock variables). Прикладами є курсів валют, кількості населення, ряди цін на товари, маси об'єкта, температури хворого, електричного напруження у мережі, тощо. У інтервальних

рядах значення показника характеризуються за певні періоди часу, що відповідає змінним типу потоку (flow variables). Прикладами є ряди динаміки виробництва продукції (добового, квартального, місячного, річного,), витрат реагентів у хімічному реакторі, міграції населення, прибутку підприємства, валового національного доходу, розсіюваної потужності, тощо.

Часовий ряд, утворений із перших різниць рівнів моментного ряду, є інтервальним. Діє і зворотнє правило: динамічний ряд, рівні якого отримані підсумовуванням або наростаючим підсумком всіх рівнів інтервального ряду, починаючи з певного рівня, є моментним. Співвідношення між моментними та інтервальними часовими рядами подібне відношенню між функціями та їх похідними. Наприклад: ряди, зображені на рисунках 1.2−1.4, є моментними, тоді як той, що представлений на рисунку 1.1, є інтервальним.

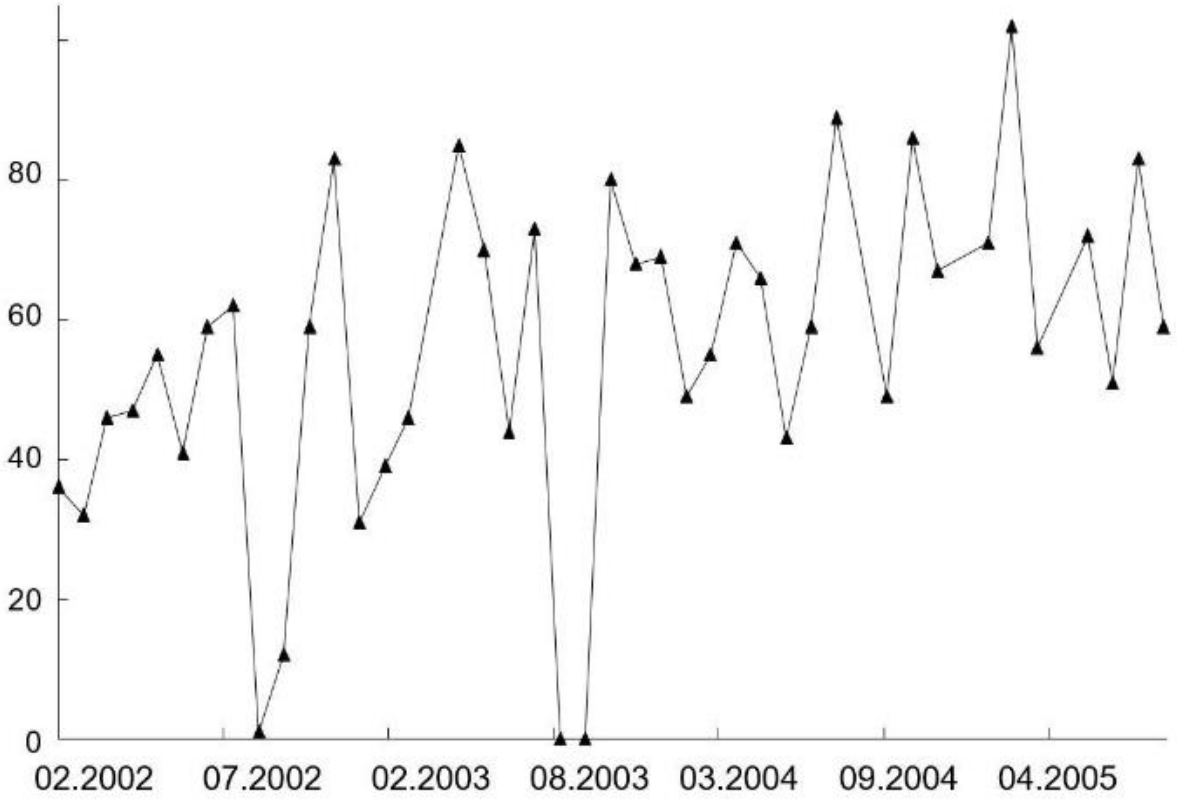


Рис. 1.1 Часовий ряд демонструючий кількість затверджених док. дисертацій (за ВАК України у 2002–2005 рр.)

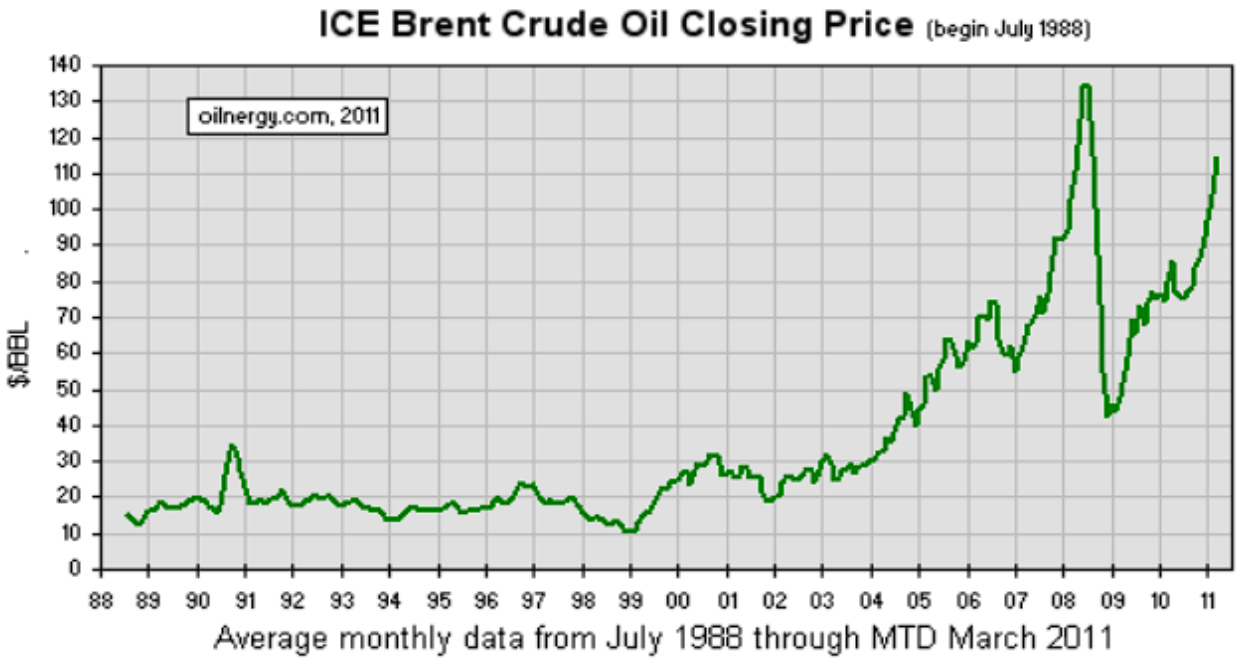


Рис. 1.2 Середня ціна нафти Brent з липня 1988 до березня 2011року

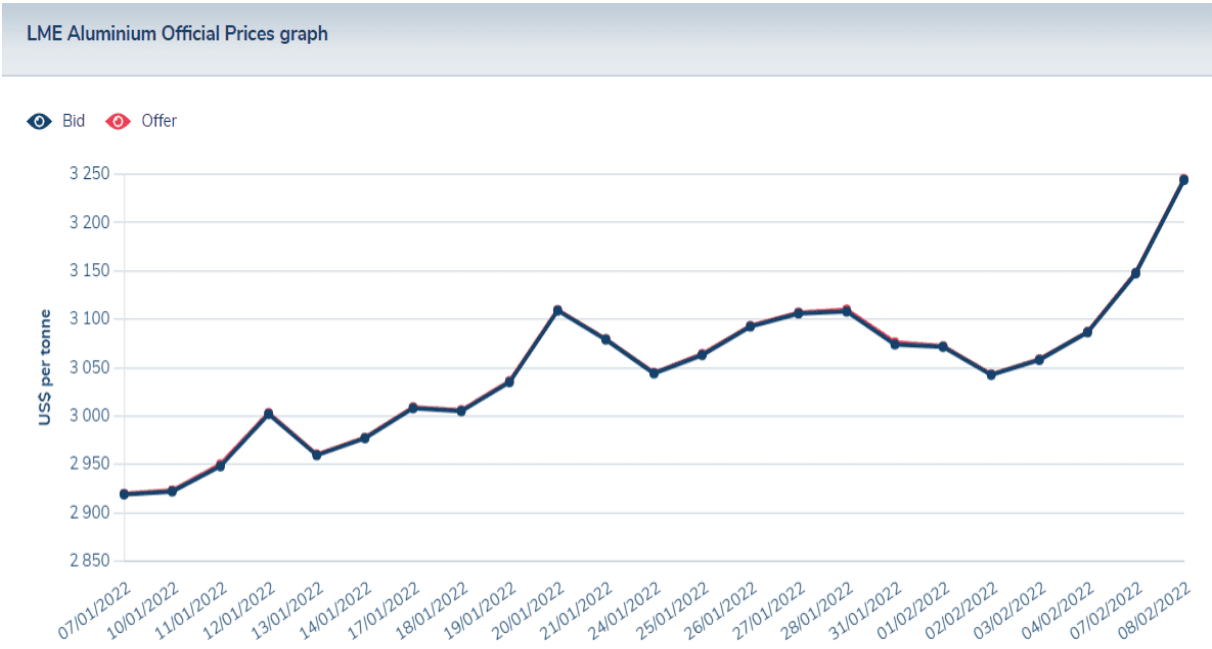


Рис. 1.3 Динаміка ціни на алюміній

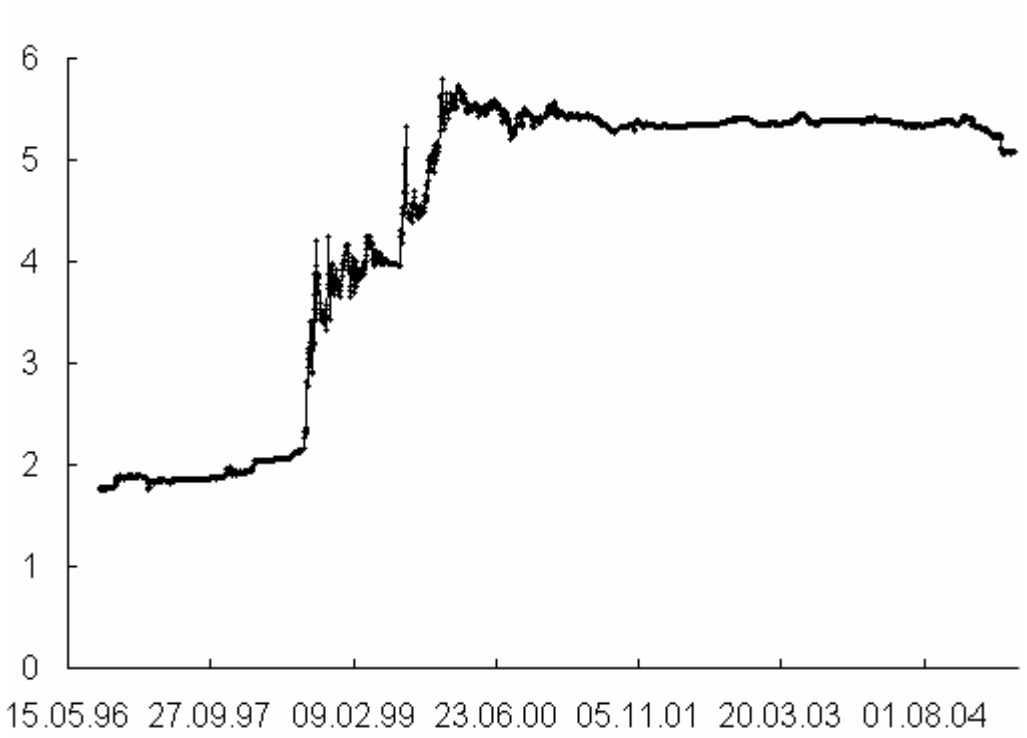


Рис. 1.4 Зміна валютного курсу " долар США до гривні" у 1996−2005 рр. за даними Нац. банку України

Розрізняють **повні** й **неповні** часові ряди, при цьому в неповних часових рядах відсутні рівні, які відповідали б певним моментам часу.

Рівні рядів динаміки можуть бути абсолютними, відносними або середніми значеннями певних показників. Якщо ці рівні не є величинами, що вимірюються безпосередньо, а є похідними від них, такими як середні, відносні тощо, то відповідні ряди називають **похідними**. Наприклад, ряди середньомісячного або середньодобового виробництва певної продукції є похідними. Ряд, показаний на рис. 1.2, є прикладом похідного ряду.

Інтервальні ряди відзначаються можливістю підсумовування їх рівнів, що призводить до отримання **нагромаджених підсумків**. Ці значення представляють собою величини певного показника за відповідний період часу. Наприклад, сумуючи місячні обсяги виробництва продукції протягом року, ми отримуємо загальний обсяг виробництва за цей рік.

У випадку моментних рядів суми їхніх рівнів не мають суттєвого змісту. Наприклад, якщо ці рівні представляють значення середньодобової температури, то сумування їх не приведе до отримання величини з фізичним змістом. Для

отримання такої величини потрібно результат поділити на кількість діб у досліджуваному періоді, щоб отримати середнє значення температури за відповідний період часу. Різниця між рівнями моментного ряду вказує на зміну показника за відповідний період.

Часовим рядам притаманні такі властивості:

- рівні часового ряду, як правило, не є статистично незалежними;
- члени часового ряду зазвичай не є однаково розподіленими.

Успішність аналізу часового ряду суттєво залежить від правильності вибору інтервалу між його сусідніми рівнями. Зручніше вибирати рівновіддалені рівні ряду, але це не завжди можливо. Наприклад, фондові індекси та офіційні курси валют не визначаються у вихідні та святкові дні, тому рівні відповідних рядів не є рівновіддаленими. Вибір надто великого інтервалу може призвести до втрати інформації про суттєві особливості динамічного ряду, і кількість членів цього ряду може бути недостатньою для застосування деяких методів аналізу. З іншого боку, надто малі інтервали збільшують обсяг розрахунків і можуть затінювати особливості динаміки показника випадковими флуктуаціями.

Важливою умовою аналізу є порівнянність рівнів ряду. Непорівнянність може виникнути через зміни методик розрахунку, адміністративно-територіальних одиниць, законодавства, технологій виробництва, та інші фактори. Інформація про ряд повинна бути достатньо повною, зокрема, для рядів із циклічною складовою, потрібна інформація про проміжок, який перевищує 4–7 повних циклів. При побудові регресійних моделей важливо мати ряди, довжина яких у кілька разів перевищує кількість параметрів.

Рівні часових рядів можуть містити викиди (аномальні значення), які можуть бути зумовлені помилками в зборі та обробці інформації чи реальними стрибками у динаміці показників. Урахування цих викидів важливо при аналізі, оскільки вони можуть вплинути на достовірність результатів.

1.3. Декомпозиція часового ряду

Економічні часові ряди з реального життя є динамічно нестабільними, тобто не стаціонарними, і використання поняття стаціонарності часто є зручною

абстракцією для використання методів статистичного аналізу. Кожен рівень часового ряду формується під впливом великої кількості різноманітних чинників, що відображають як систематичні, так і випадкові аспекти його формування. В аналізі часових рядів ряд yt зазвичай подається як сума систематичної компоненти (середньої) та випадкових відхилень від неї:

$$y_t = f(t) + \varepsilon_t, (1{,}1)$$

Де: f (t) — невипадкова функція часу (детермінована частина);

ε_t — випадкова, недетермінована частина.

Мета розкладання часового ряду полягає у вивченні факторів, які впливають на його фактичні значення, відокремленні основних та випадкових(другорядних) компонент, а серед основних — виділенні еволюційних та періодичних, включаючи сезонні, елементів.

Еволюційні фактори визначають основний напрямок розвитку економічного показника, представляючи провідну тенденцію. Тенденція являє собою не випадкову складову динамічного ряду, що повільно змінюється, і може бути описана функцією $н_t$, відомою як функція тренду або просто тренд. За допомогою тренду відображають вплив постійних чинників на економічний показник, значущість яких накопичується з часом. У масштабному значенні, тренд визначає будь-який впорядкований процес, відмінний від випадкового, тобто функцію f(t) у вище зазначеній формулі. Іноді під трендом розуміють також зміну математичного сподівання в залежності від часу. Відносно v_t допускається, що це гладка функція, ступінь гладкості якої не відомий заздалегідь. Під ступенем гладкості мається на увазі мінімальний ступінь полінома, який найкраще згладжує компоненту $н_t$. На рис. 1.5 показано умовний динамічний ряд із лінійно зростаючою тенденцією.

Серед факторів, які впливають на постійно повторювані коливання ряду, можна виділити такі:

Сезонні чинники, що характеризуються періодичними або подібними до них коливаннями, які спостерігаються упродовж одного року. Прикладами можуть бути збільшення рівня безробіття у курортних містах під час зимового

сезону та зміни цін на сільгосппродукцію у різні сезони. Сезонні чинники можуть включати елементи, пов'язані з людською діяльністю, такі як релігійні традиції , свята, відпустки та інше. Наприклад, в квартальних рядах будуть присутні сезонні коливання з періодом 4, а в щомісячних даних з періодом - 12. Рисунок 1.6 ілюструє умовний часовий ряд, що містить виключно сезонну компоненту. Функцію, яка моделює результат впливу сезонних чинників, позначають як S_t.

Кон'юнктурні або циклічні коливання мають схожість з сезонними, але виявляються на більших інтервалах часу. Циклічні коливання пояснюються впливом тривалих циклів демографічної, астрофізичної або економічної природи. Наприклад, за багаторічними спостереженнями активність сонця має циклічність у 10,5—11 років, що впливає на репродуктивну властивість тварин та врожайність зернових культур тощо. Таким чином, динаміка показника містить характерні повторювані коливання з однаковою циклічністю. Результат впливу циклічних чинників моделюється функцією c_t.

Тренд, сезонну й циклічну компоненти називають **систематичними компонентами часового ряду,** тому що вони не є випадковими.

Випадкові чинники, хоч і не підлягають вимірюванню, супроводжують будь-який економічний процес, надаючи йому стохастичний характер. До випадкових чинників можна віднести випадкові збурення, помилки вимірювання та інші. В таких динамічних рядах, як стаціонарні, що не мають тренду та сезонної складової, кожен наступний рівень формується як сума середнього рівня ряду та випадкової (додатної або від'ємної) компоненти. Прикладом такого ряду є рисунок 1.7. Вплив випадкових чинників позначається як **випадкова компонента** ε_t, яку обчислюють як залишок або похибку, що залишається після вилучення систематичних компонент з часового ряду. Це не означає, що ця компонента не підлягає подальшому аналізу, оскільки вона представляє собою лише хаос.

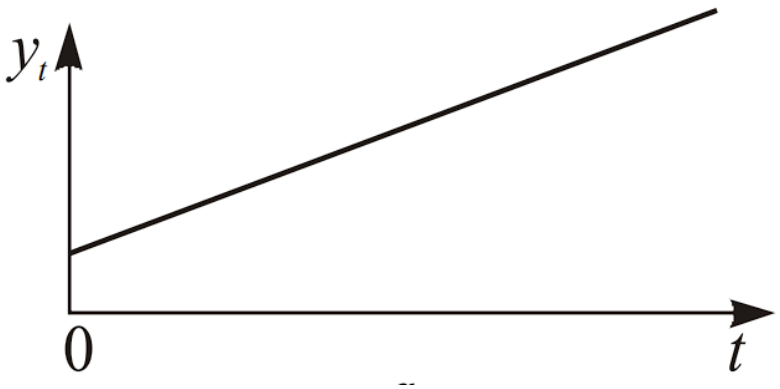


Рис. 1.5 тренд, що зростає

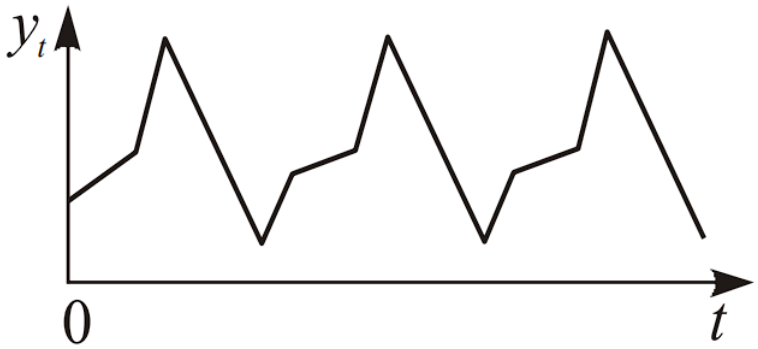


Рис. 1.6 сезонна компонента

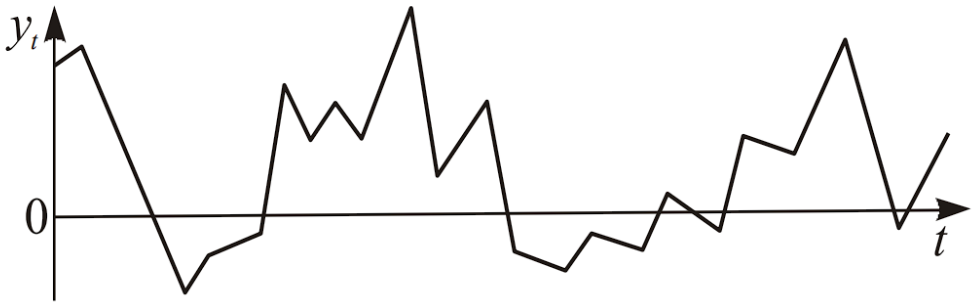


Рис. 1.7 випадкова компонента

Вокористовуючи **декомпозицію Вольда** суто недетермінований стаціонарний у широкому сенсі випадковий процес можна записати фомулою:

$$y_t - \mu = \varepsilon_t + \Psi_1 \varepsilon_{t-1} + \Psi_2 \varepsilon_{t-2} + \cdots = \sum_{\tau=0}^{\infty} \Psi_t * \varepsilon_{t-\tau} \quad (1{,}2)$$

Де: μ — детермінована складова або математичне сподівання цього процесу, ε_t — білий шум з обмеженими математичним сподіванням та дисперсією. Метод Вольда, також відомий під назвою “лінійний фільтр”, можна

розглядати як процес, у якому білий шум проходить через фільтр, що дозволяє зручно представити і вивчити стаціонарні процеси, не втрачаючи повноти.

Щоб вираз мав сенс, повинна виконуватися умова збіжності за ймовірністю, оскільки підсумовуються випадкові величини. Ця умова записується у вигляді $\sum_{\tau=0}^{\infty} \Psi_t^2 < \infty$. Припускається, що $\Psi_0 = 1$. Чим більший ваговий критерій Ψ_t, тим більший вплив випадкового збурення в момент $t - \tau$ на поточний момент *t*.

Вивчення випадкової компоненти виявляється ключовою інформативною складовою при аналізі динамічних рядів. Це пояснюється тим, що в короткостроковому та, до певної міри, середньостроковому прогнозуванні результати прогнозу напряму залежать від випадкової компоненти, в той час як у довгостроковому прогнозуванні основний акцент робиться на визначенні тенденцій та взаємозв'язків між факторами.

Очевидно, реальні дані цілковито не відповідають лише одній із наведених функцій, тож часовий ряд y_t, $t=1,2, \ldots, n$ можна уявити у вигляді розкладення:

$$y_t = н_t + s_t + c_t + e_t, t = 1,2,\cdots,n \ (1,3)$$

або різноманітних поєднань окремих функцій. Однак наявність випадкової складової завжди є обов'язковою. Декомпозиція (розкладення) часового ряду відбувається за такими варіантами моделей:

модель тренду:

$$y_t = н_t + e_t, \ t = 1,2,\cdots,n; \ (1,4)$$

модель сезонності:

$$y_t = s_t + e_t, \qquad t = 1,2,\cdots,n; \ (1,5)$$

Тренд-сезонна модель:

$$y_t = н_t + s_t + e_t, \qquad t = 1,2,\cdots,n; \ (1,6)$$

Моделі тренду й сезонності (тренд-сезонні) можуть відображати як відносно постійну сезонну хвилю (цикл), так і динамічно змінювану залежно від тренду. Формули зазначені вище належать до **адитивних**, формула

$$y_t = н_t * s_t * c_t * e_t, \ t = 1,2,\cdots,n; \ (1,7)$$

до **мультиплікативних** моделей.

Мультиплікативна модель базується на припущенні, що чотири компоненти динамічного ряду можуть взаємодіяти і впливати один на одного, у той час як адитивна модель передбачає їхню незалежність.

Моделі для урахування циклічних чинників будуються аналогічно до трендово-сезонних, за винятком введення циклічної, а не сезонної складової.

Процес окремого обчислення функцій H_t, c_t, St і ε_t відомий як фільтрація компонент динамічного ряду y_t. Процесс оцінювання детермінованої частини разом з усіма невипадковими компонентами називають процесом з**гладжування** часового ряду.

Успішне розв'язання завдань виявлення та моделювання впливу розглянутих складових є важливим кроком для розуміння процесу формування соціально-економічного механізму та його прогнозування.

Проте важливо пам'ятати, що операція розкладення ряду динаміки, яка є математично допустимою та корисною для моделювання динаміки зміни показників у часі, іноді може вводити в оману. Зокрема, припущення щодо незалежного впливу названих компонент та їхньої чіткої структури за такого підходу може бути дуже спрощеним.

1.4 Критерій вибору кращої моделі

Існують такі ситуації коли задовільними виявляються кілька моделей. В таких випадках потрібно використовувати правила вибору між ними. Бокс і Дженкінс запропонували принцип ощадливості, згідно з яким, маючи кілька адекватних моделей, треба обрати модель із найменшою кількістю параметрів. Для використання цього принципу треба формалізувати правило компромісу між кількістю її параметрів точністю пристосування моделі та точністю пристосування моделі. Існує кілька підходів до розв’язання цієї проблеми. Розглянемо підхід через числові критерії.

Числові критерії

Назва критерію	Формула підрахунку	Бажаний екстремум
Коефіцієнт детермінації	$R^2 = 1 - \frac{\sum_{t=1}^{n} e_t^2}{\sum_{t=1}^{n} (y_t - \bar{y})^2}$	1
Скоригований коефіцієнт детермінації	$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-(p+q)}$	1
Інформаційний критерій Акаїке (AIC)	$AIC = \ln(\frac{\sum_{t=1}^{n} e_t^2}{n}) + \frac{2(p+q)}{n}$	*Min*
Інформаційний критерій Шварца-Ріссанена (SIC)	$SIC = \ln\left(\frac{\sum_{t=1}^{n} e_t^2}{n}\right) + \frac{p+q}{n}\ln n$	*Min*
Критерій Ханнана-Квіна (HQ)	$HQ = \ln\left(\frac{\sum_{t=1}^{n} e_t^2}{n}\right) + \frac{c(p+q)}{n}\ln(\ln n), c \geq 2$	*Min*
Прогнозовий критерій (FC)	$FC = \frac{\sum_{t=1}^{n} e_t^2}{n-(p+q)}(1 + \frac{p+q}{n})$	*Min*

Рис. 1.8 Таблиця числових критеріїв адекватності моделей

Числові критерії. На відміну від попередніх тестувань, числові критерії лише дають певне значення, за яким можна судити про адекватність моделі. Загальну характеристику критеріїв наведено на рисунку 1.8.

Однією із пропозицій є обчислення коефіцієнта детермінації (R^2) та зваженого коефіцієнта детермінації ($\bar{R}^2$). Однак цей метод непридатний для деяких моделей з різницевої змінної z.

Інформаційні критерії ґрунтовані на мінімізації певних статистик, що мають стандартні розподіли.

Інформаційний критерій Акаїке (AIC) розглядає нелінійне компромісне співвідношення між дисперсією залишків і значенням загальної кількості оцінюваних параметрів (p + q), оскільки моделі з більшою кількістю оцінюваних параметрів можна віддати перевагу лише за пропорційно великого зменшення дисперсії залишків.

Інформаційний критерій Шварца-Ріссанена (SIC) надає більшої ваги (p + q) порівняно з AIC за n > 7, тобто зростання кількості оцінюваних параметрів потребує вагомішого зменшення дисперсії залишків для SIC, ніж для AIC.

Прогнозовий критерій (FC) використовує похибку передбачення.

Критерій Ханнана-Квіна (HQ). Тут вага при (p + q) є більшою за 2, якщо n > 15.

Вибір між цими критеріями є довільним, оскільки всі статистики змінюються в одному напрямі в разі збільшення кількості оцінюваних параметрів. На практиці користуються одним із них.

Критерій Дарбіна-Уотсона — це статистичний тест, який використовується для визначення наявності автокореляції в залишках (помилках) регресійної моделі. Цей критерій особливо корисний при аналізі часових рядів, де важливо враховувати можливу автокореляцію між послідовними спостереженнями.

Критерій Дарбіна-Уотсона (DW) приймає значення від 0 до 4. Значення близьке до 2 свідчить про відсутність автокореляції (нульове значення вказує на позитивну автокореляцію, а значення близьке до 4 — на негативну автокореляцію). Взагалі, критерій DW використовується для перевірки гіпотези про відсутність автокореляції в залишках моделі.

Основна формула для обчислення критерію Дарбіна-Уотсона є наступною:

$$DW = \frac{\sum_{t=2}^{T}(e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2} \quad (1,8)$$

Де:

e_t - залишки (помилки) моделі для кожного спостереження t.

T - кількість спостережень.

Значення DW порівнюється зі значеннями з таблиці, що визначає критичні значення для даного розміру вибірки та рівня значущості. Зазвичай, якщо DW близьке до 2, це вказує на відсутність значущої автокореляції. Зауважте, що цей тест має свої обмеження та може бути менш ефективним для виявлення автокореляції в наявності сезонності або інших складних структур часових рядів.

Висновки до розділу 1

Часові ряди можуть приймати різні форми та відображати різні патерни, такі як тренди, сезонність, циклічність та випадкові зміни.

Для аналізу та прогнозування часових рядів використовують різні методи, включаючи статистичні та машинні підходи. Моделі, які враховують попередні значення, автокореляцію та інші фактори, можуть бути ефективними для прогнозування.

Часові ряди використовуються в різних галузях, таких як фінанси, економіка, маркетинг, медицина, енергетика та інші. Прогнозування та аналіз часових рядів є важливим елементом стратегій прийняття рішень.

Для оцінки адекватності моделей часових рядів застосовуються різні статистичні тести. Критерій Дарбіна-Уотсона, тести на стаціонарність та інші допомагають визначити, наскільки добре модель враховує особливості даних.

Чим складніша модель, тим більше можливостей для відхилення від фактичних припущень моделі. Зі збільшенням параметрів моделі, ризик переобладнання також зростає. Відповідна модель часових рядів може дуже добре описувати дані навчання, але вона може виявитися непридатною для майбутнього прогнозування. Оскільки потенційне переоснащення впливає на

здатність моделі добре прогнозувати, економія часто використовується в якості орієнтира задля вирішення цієї проблеми.

Таким чином, слід приділяти увагу вибору найбільш скупої моделі серед всіх інших варіантів.

РОЗДІЛ 2. АЛГОРИТМ ПРОГНОЗУВАННЯ ОБ'ЄМІВ ПРОДАЖУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНІВ ВЗУТТЯ

2.1. Постановка задачі

Основною метою дослідницького проєкту є розробка алгоритму прогнозування об’єму продажу товару магазинів взуття, на базі адитивної або мультиплікативної моделей часового ряду. Алгоритм повинен включати в себе:

1. Вирівнювання вхідного часового ряду методом ковзної середньої.

Метод ковзного середнього заснований на властивості середньої погашати випадкові відхилення від загальної закономірності. Цей метод найкраще підходить для даних з невеликими випадковими відхиленнями від деякого постійного або повільно змінюваного значення.

У цьому методі середнє фіксованого числа n-останніх спостережень використовується для оцінки наступного значення рівня ряду. Значення прогнозу, отриманого методом простого ковзного середнього, завжди менше фактичного значення – якщо вихідні дані монотонно зростають, і навпаки більше фактичного значення -якщо вихідні дані монотонно спадають.

Недоліки методу ковзного середнього: при обчисленні прогнозу останні спостереження мають приблизно одну й ту ж значимість, що протирічить інтуїтивному висновку про більшу значимість для прогнозу останнього спостереження.

2. Розрахунок значень сезонної компоненти.

У загальному випадку часовий ряд можна представити з трьох компонентів:

- Сезонної компоненти (Позначається St, де t позначає момент часу)
- Тренд(Tt)
- Випадкова, нерегулярна компонента(Et)

Оцінка сезонної компоненти являє собою різницю між фактичними рівнями ряду і центрованим ковзним середнім. Ці оцінки використовуються для розрахунку сезонної компоненти S.

Особливість моделей з сезонною компонентою: в моделях з сезонною компонентою, як правило, сезонні компоненти взаємопогашаються, в адитивній моделі це виражається в тому, що сумма значень сезонної компоненти по всім кварталам =0. В мультипликативній моделі це виражається в тому, що сума значень сезонної компоненти по всім кварталам повинна бути рівна числу періодів в циклі. В даному алгоритмі число періодів одного цикла рівне 4.

Для знаходження скоригованої сезонної компоненти потрібно розрахувати коригуючий коефіцієнт-середнє арифметичне середньої оцінки сезонної компоненти. Потім цей коригуючий коефіцієнт віднімається від середнього значення оцінки сезонної компоненти для кожного з 4-х кварталів. Такмс чином знаходиться скоригована сезонна компонента для 4-х кварталів.

Далі слід виключити вплив сезонної компоненти з кожного рівня вихідного ряду. Ці значення містять тільки тенденцію і випадкову компоненту та розраховуються для кожного члена ряду T+E=Y-S для адитивної моделі та T*E=Y/S для мультиплікативної.

3. Пошук рівняння лінії тренду за допомогою методу найменших квадратів.

Метод найменших квадратів - це метод, який використовується для знаходження найкращого підгону лінії (або іншої функції) до набору даних, мінімізуючи суму квадратів відхилень між фактичними і передбаченими значеннями. Загальна ідея полягає в тому, щоб знаходити такі параметри моделі, які найкраще вписуються у ваші дані, мінімізуючи відхилення між фактичними і передбаченими значеннями.

Для визначення параметрів рівняння треба використати систему лінійних рівнянь отриману методом найменших квадратів. Система має вигляд:

$$\begin{cases} a_0 n + a_1 \sum t = \Sigma y \\ a_0 \Sigma t + a_1 \Sigma t^2 = \sum (y \cdot t) \end{cases} \quad (2{,}1)$$

Така система лінійних рівнянь може вирішуватися трьома способами: Метод Гауса, Метод Крамера, Матричний метод.

4. Аналітичне вирівнювання рівнів ряду з використанням отриманого рівняння тренду.

Загальне рівняння лінії тренду має вигляд:

$$T = bx + a \quad (2,2)$$

Де: a та b коефіцієнти лінії тренду, що знаходяться в результаті методу найменших квадратів.

x - це момент часу t (квартал).

Рівні тренду T знаходяться для кожного моменту часу.

5. Перевірка адекватності моделі шляхом пошуку коефіцієнту детермінації та критерію Дарбіна-Уотсона.

Формула коефіцієнту детермінації має вигляд:

$$R^2 = 1 - \frac{\sum_{t=1}^{n} e_t^2}{\sum_{t=1}^{n} (y_t - \bar{y})^2} \quad (2,3)$$

Де: *e* – це похибка, випадкова величина, що дорівнює різниці фактичного рівня часового ряду y_t та суми сезонної компоненти S_t і тренда T_t.

$\bar{y}$ – це середнє арифметичне всіх фактичних елементів динамічного ряду.

Залежно від отриманого значення коефіцієнта детермінації, можна поділити модель на три групи:

$0,8<R^2<1$ – модель є адекватною;

$0,5<R^2<0,8$ – модель прийнятної якості;

$0<R^2<0,5$ – модель є неадекватною;

Формула критерію Дарбіна-Уотсона має вигляд:

$$DW = \frac{\sum_{t=2}^{T} (e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2} (2,4)$$

Де: *e* – це випадкова величина (похибка), яка дорівнює різниці фактичного рівня часового ряду y_t та суми сезонної компоненти S_t і тренда T_t.

Значеня DW завжди знаходяться в діапазоні від 0 до 4. Якщо отримане значення наближене до 2, то це означає, що автокореляція випадкових величин відсутня і модель можна вважати адекватною.

6. Прогнозування майбутніх значень

Отримавши значення скоригованої сезонної компоненти S та значень тренду T, можна здійснити прогноз майбутніх значень згідно з моделлю.

$F_t=S_t+T_t$ для адитивної моделі;

$F_t=S_t*T_t$ для мультиплікативної моделі;

Де: F - прогнозоване значення, t - момент часу в майбутньому.

Для збереження точності прогнозу, доцільно виконувати прогноз максимум на 5 майбутніх кварталів.

Для оцінки точності прогнозу використовують середнє абсолютне відхилення (САО) та середнє відносних помилок (СООП)

$$CAO = \frac{\sum|e|}{N} \quad (2,5)$$

$$\text{СООП} = \frac{\sum \frac{|e|}{y_t} \cdot 100\%}{N} \quad (2,6)$$

Де: *N*- кількість моментів часу *t* (кварталів), *e*- випадкова величина (похибка), y_t- значення рівня динамічного ряду в момент часу *t*.

2.2. Реалізація алгоритму з реальними даними для адитивної моделі

Для виконання алгоритму було обрано даний часовий ряд, що симулює об’єм продажу умовного інтернет-магазину взуття, як експерементні дані:

Квартал	1	2	3	4	5	6	7	8	9	10	11	12	13
Об’єм продажу	239	201	182	297	324	278	257	384	401	360	335	462	481

Побудова адитивної моделі складається з наступних кроків:

1. Вирівнювання вхідного часового ряду методом ковзної середньої за 4-ма кварталами.

Квартал	Об'єм продажу	МКС	ЦКС	ОСК
1	239			
2	201			
3	182	229,75	240,375	-58,375
4	297	251	260,625	36,375
5	324	270,25	279,625	44,375
6	278	289	299,875	-21,875
7	257	310,75	320,375	-63,375
8	384	330	340,25	43,75
9	401	350,5	360,25	40,75
10	360	370	379,75	-19,75
11	335	389,5	399,5	-64,5
12	462	409,5		
13	481			

Рис. 2.1 Таблиця розрахунку значень методом ковзної середньої за 4-ма кварталами для адитивної моделі

Стовбець Методу ковзного середнього (МКС) знаходиться шляхом різниці суми значень об’єму продажу за квартал на кількість кварталів, наприклад:

229.75=(239+201+182+297)/4;

251=(201+182+297+324)/4;

……

409,5=(360+335+462+481)/4;

Значення стовбця центрованої ковзної середньої (ЦКС) дорівнюють середньому значенню двох ковзних середніх.

240,375=(229,75+251)/2;

260,625=(251+270,25)/2;

…….

399,5=(389,5+409,5)/2;

Оцінка сезонної компоненти (ОСК) отримується шляхом різниці фактичного значення рівня ряду (об’єм продажу) та центрованої ковзної середньої.

-58,375=182-240,375;

36,375=297-260,625;

…….

-64,5=335-399,5;

2. Пошук сезонної компоненти.

	Квартал						
	1	2	3	4			
Оцінка сезонної варіації	0	0	-58,375	36,375			
	44,375	-21,875	-63,375	43,75	Сума середніх значень		Корегуючий коефіцієнт
	40,75	-19,75	-64,5	0			
Середнє	42,5625	-20,8125	-62,0833	40,0625	-0,27083		-0,06771
Скоректована сезонна варіація	42,63021	-20,7448	-62,0156	40,13021	0		
					Сума значень сезонної компоненти повинна = 0		

Рис. 2.2 Таблиця розрахунку скорегованої сезонної варіації для адитивної моделі

Оцінка сезонної варіації береться з попередньої таблиці (ОСК) та рівномірно ділиться між 4-ма кварталами.

Середнє значення сезонної варіації рахується за кожний квартал:

42,5625=(44,375+40,75)/2;

-20,8125=(-21,275-19,75)/2;

-62,0833=(-58,375-63,375-64,5)/3;

40,0625=(36,375-43,75)/2;

Коригуючий коефіцієнт отримується шляхом ділення суми середніх значень сезонної варіації на кількість кварталів:

-0,06771=-0,2783/4;

Скорегована сезонна варіація(Si) дорівнює різниці середнього значення та коригуючого коефіцієнту:

42,63021=42,5625+0,06771;

-20,7448=-20,8125+0,06771;

-62,0156=-62,0833+0,06771;

40,13021=40,0625+0,06771;

Сума скорегованих сезонних варіацій в адитивній моделі завжди повинна дорівнювати 0!

3. Розрахунок показників лінії тренду методом найменших квадратів.

	t	y	t^2	y^2	ty
	1	196,37	1	38561,0951	196,37
	2	221,745	4	49170,7526	443,49
	3	244,016	9	59543,6252	732,047
	4	256,87	16	65982,0899	1027,48
	5	281,37	25	79168,9597	1406,85
	6	298,745	36	89248,4505	1792,47
	7	319,016	49	101770,969	2233,11
	8	343,87	64	118246,434	2750,96
	9	358,37	81	128428,908	3225,33
	10	380,745	100	144966,596	3807,45
	11	397,016	121	157621,406	4367,17
	12	421,87	144	177974,121	5062,44
	13	438,37	169	192168,074	5698,81
Сума:	91	4158,37	819	17292039,3	32744

Рис. 2.3 Таблиця квадратів

В ролі *y* в даній таблиці виступають значення об’єму продажу з вилученою сезонною компонентою Si.

196,37=239-42,6302;

221,745=201+20,745;

……...

438,37=481-42,6302;

Система лінійних рівнянь для пошуку коефіцієнтів:

$$\begin{cases} 13x_1 + 91x_2 = 4158{,}37 \\ 91x_1 + 819x_2 = 32744 \end{cases}$$

Метод Крамера для даної системи:

$$\Delta = \begin{vmatrix} 13 & 91 \\ 91 & 819 \end{vmatrix} = 13 \cdot 819 - 91 \cdot 91 = 10647 - 8281 = 2366$$

$$\Delta_1 = \begin{vmatrix} 4158.37 & 91 \\ 32744 & 819 \end{vmatrix} = 4158.37 \cdot 819 - 32744 \cdot 91 = 3405705.03 - 2979704 = 426001.03$$

$$\Delta_2 = \begin{vmatrix} 13 & 4158.37 \\ 91 & 32744 \end{vmatrix} = 13 \cdot 32744 - 91 \cdot 4158.37 = 425672 - 378411.67 = 47260.33$$

Рис. 2.4 Пошук визначників матриці за методом Крамера

$$x_1 = \frac{\Delta_1}{\Delta} = \frac{426001.03}{2366} = 180{,}052$$

$$x_2 = \frac{\Delta_2}{\Delta} = \frac{47260.33}{2366} = 19{,}9745$$

Після отримання коефіцієнтів рівняння лінії тренду має вигляд:

T=180,0504 + 19,9745 *t.

4. Аналітичне вирівнювання рівнів ряду з використанням отриманого рівняння тренду.

t	Yt	Si	Yt-Si	T	T+Si	e	|e|	e^2
1	239	42,63	196,37	200,03	242,66	-3,657	3,6571	13,374
2	201	-20,74	221,74	220	199,26	1,7434	1,7434	3,0394
3	182	-62,02	244,02	239,98	177,96	4,0397	4,0397	16,319
4	297	40,13	256,87	259,95	300,08	-3,081	3,0806	9,4901
5	324	42,63	281,37	279,92	322,56	1,4449	1,4449	2,0877
6	278	-20,74	298,74	299,9	279,15	-1,155	1,1546	1,3331
7	257	-62,02	319,02	319,87	257,86	-0,858	0,8583	0,7366
8	384	40,13	343,87	339,85	379,98	4,0214	4,0214	16,172
9	401	42,63	358,37	359,82	402,45	-1,453	1,4531	2,1115
10	360	-20,74	380,74	379,8	359,05	0,9474	0,9474	0,8976
11	335	-62,02	397,02	399,77	337,76	-2,756	2,7563	7,5971
12	462	40,13	421,87	419,75	459,88	2,1234	2,1234	4,5088
13	481	42,63	438,37	439,72	482,35	-1,351	1,3511	1,8255

Рис. 2.5 Таблиця адитивної моделі зі значеннями тренду(T) і помилок *e*

Значення тренду для кожного моменту часу t знаходиться шляхом підстановки порядкового номеру в рівняння лінії тренду:

200,03=180,0504 + 19,9745 *1;

220=180,0504 + 19,9745 *2;

……..

439,72=180,0504 + 19,9745 *13;

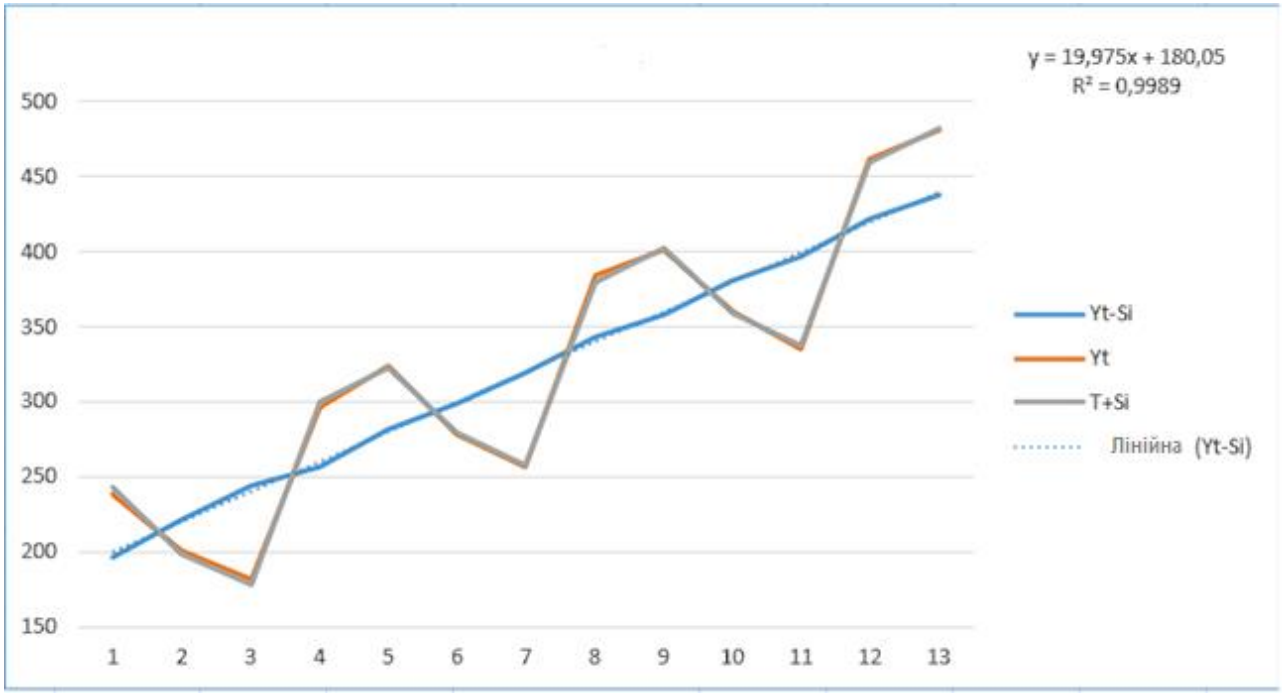


Рис. 2.6 Графік адитивної моделі з лінією тренду

5. Перевірка адекватності моделі.

За формулою коефіцієнта детермінації з розділу 2.1 R^2=0,996. Критерій Дарбіна-Уотсона для даної моделі дорівнює DW=2,7878. Модель з такими показниками можна назвати адекватною.

6. Прогнозування на основі адитивної моделі.

Прогноз буде здійснюватися на наступні 2 квартали, для збереження максимальної точності. В адитивній моделі для отримання значення прогнозу на момент часу в майбутньому потрібно додати значення скорегованої сезонної варіації Si та тренд T для прогнозуючого моменту часу.

$F_{14}=Si_{14}+T_{14}$= 459,6954 -20,7448= 438,9506;

$F_{15}=Si_{15}+T_{15}$= 479,6699 - -62,0156= 417,6543;

Залишилося лише перевірити точність прогнозу за допомогою середньої абсолютної похибки (САО) та середнього відносних помилок (СООП). Для даної моделі значення похибок:

САО=2,2;

СООП=0,77%;

Якщо значення СООП не перевищує 5% то прогноз можна вважати точним.

2.3. Реалізація алгоритму з реальними даними для мультиплікативної моделі

Для мультиплікативної моделі використовувався той самий експериментальний динамічний ряд, що й для адитивної моделі. На прикладі однакових значень вхідних даних можна ярко визначити особливості обох моделей.

Квартал	1	2	3	4	5	6	7	8	9	10	11	12	13
Об’єм продажу	239	201	182	297	324	278	257	384	401	360	335	462	481

Побудова мультиплікативної моделі покроково не відрізняється від адитивної, але вагома різниця є в розрахунках. Отже мультиплікативна модель включає в себе такі пункти:

1. Вирівнювання вхідного часового ряду методом ковзної середньої за 4-ма кварталами.

Квартал	Об'єм продажу	МКС	ЦКС	ОСК
1	239			
2	201			
3	182	229,75	240,375	0,75715
4	297	251	260,625	1,139568
5	324	270,25	279,625	1,158695
6	278	289	299,875	0,927053
7	257	310,75	320,375	0,802185
8	384	330	340,25	1,128582
9	401	350,5	360,25	1,113116
10	360	370	379,75	0,947992
11	335	389,5	399,5	0,838548
12	462	409,5		
13	481			

Рис. 2.7 Таблиця розрахунку значень методом ковзної середньої за 4-ма кварталами для мультиплікативної моделі

Стовбці методу ковзного середнього (МКС) та центрованої ковзної середньої (ЦКС) рахуються ідентично з алгоритмом для мультиплікативної моделі, а оцінка сезонної компоненти (ОСК) отримується шляхом ділення фактичного значення рівня ряду (об’єм продажу) на центровану ковзну середню.

0,75715=182/240,375;
1,139568=297/260,625;

…….

0,838548=335/399,5;

2. Пошук сезонної компоненти.

	1	2	3	4			
Оцінка сезонної варіації	0	0	0,75715	1,139568			
	1,158695	0,927053	0,802185	1,128582			
	1,113116	0,947992	0,838548	0	Сума середніх значень	Корегуючий коефіцієнт	
Середнє	1,135905	0,937523	0,799294	1,134075	4,006797	0,998304	
Скоректована сезонна компонента Si	1,133978	0,935932	0,797938	1,132151	4		
					Сума скорегованих сезонних компонент повинна дорівнювати кількості кварталів		

Рис. 2.8 Таблиця розрахунку скорегованої сезонної варіації для мультиплікативної моделі

Оцінка сезонної варіації береться з попередньої таблиці (ОСК) та рівномірно ділиться між 4-ма кварталами, ідентично адитивній моделі.

Середнє значення сезонної варіації рахується за кожний квартал:

1,135905=(1,158695+ 1,113116)/2;

0,937523=(0,927053+ 0,947992)/2;

0,799294=(0,75715+ 0,802185+0,838548)/3;

1,134075=(1,139568+ 1,128582)/2;

Коригуючий коефіцієнт в мультиплікативній моделі отримується шляхом ділення кількості кварталів на суму середніх значень:

0,998304=4/4,006797;

Скорегована сезонна варіація(Si) дорівнює добутку середнього значення та коригуючого коефіцієнту:

1,133978= 1,135905* 0,998304;

0,935932= 0,937523* 0,998304;

0,797938= 0,799294* 0,998304;

1,134075= 1,132151* 0,998304;

Сума скорегованих сезонних варіацій в мультиплікативній моделі завжди повинна дорівнювати 4, на відміну від адитивної!

3. Розрахунок показників лінії тренду методом найменших квадратів.

	t	y	t^2	y^2	ty
	1	210,7624	1	44420,8	210,7624
	2	214,7592	4	46121,51	429,5184
	3	228,0878	9	52024,02	684,2633
	4	262,3324	16	68818,31	1049,33
	5	285,7198	25	81635,78	1428,599
	6	297,0301	36	88226,89	1782,181
	7	322,0800	49	103735,5	2254,56
	8	339,1773	64	115041,2	2713,418
	9	353,6223	81	125048,7	3182,601
	10	384,6433	100	147950,5	3846,433
	11	419,8319	121	176258,8	4618,15
	12	408,0727	144	166523,3	4896,872
	13	424,1704	169	179920,5	5514,215
Сума	91	4150,290	819	1395726	32610,9

Рис. 2.9 Таблиця квадратів (2)

В ролі *y* в даній таблиці виступають значення об'єму продажу з вилученою сезонною компонентою Si, але їх значення вже відрізняється від адитивної моделі, та знаходиться шляхом ділення фактичного значення об'єму продажу на коригуючу сезонну компоненту.

210,7624=239/1,133978;

214,7592=201/0,935932;

…….....

424,1704=481/1,133978;

Система лінійних рівнянь для пошуку коефіцієнтів:

$$\begin{cases} 13x_1 + 91x_2 = 4150{,}29 \\ 91x_1 + 819x_2 = 32610{,}9 \end{cases}$$

Метод Крамера для даної системи:

$$\Delta = \begin{vmatrix} 13 & 91 \\ 91 & 819 \end{vmatrix} = 13 \cdot 819 - 91 \cdot 91 = 10647 - 8281 = 2366$$

$$\Delta_1 = \begin{vmatrix} 4150.29 & 91 \\ 32610.9 & 819 \end{vmatrix} = 4150.29 \cdot 819 - 32610.9 \cdot 91 = 3399087.51 - 2967591.9 = 431495.61$$

$$\Delta_2 = \begin{vmatrix} 13 & 4150.29 \\ 91 & 32610.9 \end{vmatrix} = 13 \cdot 32610.9 - 91 \cdot 4150.29 = 423941.7 - 377676.39 = 46265.31$$

Рис. 2.10 Пошук визначників матриці за методом Крамера (2)

$$x_1 = \frac{\Delta_1}{\Delta} = \frac{431495.61}{2366} = 182{,}373$$

$$x_2 = \frac{\Delta_2}{\Delta} = \frac{46265.31}{2366} = 19{,}554$$

Після отримання коефіцієнтів рівняння лінії тренду має вигляд:

T=182,373 + 19,554*t.

4. Аналітичне вирівнювання рівнів ряду з використанням отриманого рівняння тренду.

t	yt	Si	yt/Si	T	TxSi	E=Yt/(TxSi)	|E|	E^2
1	239	1,133978	210,7624	201,927	228,9808	10,019172	10,01917	100,3838
2	201	0,935932	214,7592	221,481	207,2912	-6,2911643	6,291164	39,57875
3	182	0,797938	228,0878	241,035	192,3311	-10,331104	10,3311	106,7317
4	297	1,132151	262,3324	260,589	295,0262	1,9738481	1,973848	3,896076
5	324	1,133978	285,7198	280,143	317,6761	6,3239291	6,323929	39,99208
6	278	0,935932	297,0301	299,697	280,496	-2,4960248	2,496025	6,23014
7	257	0,797938	322,08	319,251	254,7427	2,2573387	2,257339	5,095578
8	384	1,132151	339,1773	338,805	383,5785	0,421509	0,421509	0,17767
9	401	1,133978	353,6223	358,359	406,3713	-5,3713142	5,371314	28,85102
10	360	0,935932	384,6433	377,913	353,7009	6,2991147	6,299115	39,67885
11	335	0,797938	419,8319	397,467	317,1542	17,845782	17,84578	318,4719
12	462	1,132151	408,0727	417,021	472,1308	-10,13083	10,13083	102,6337
13	481	1,133978	424,1704	436,575	495,0666	-14,066558	14,06656	197,868

Рис. 2.11 Таблиця мультиплікативної моделі зі значеннями тренду(T) і помилок *e*

Значення тренду для кожного моменту часу t знаходиться шляхом підстановки порядкового номеру в рівняння лінії тренду, аналогічно з адитивною моделлю:

201,927=182,373 + 19,554*1;

221,481=182,373 + 19,554*2;

……..

436,575=182,373 + 19,554*13;

Рис. 2.12 Графік мультиплікативної моделі з лінією тренду

5. Перевірка адекватності моделі.

За формулою коефіцієнта детермінації з розділу 2.1 R^2=0,9905. Критерій Дарбіна-Уотсона для даної моделі дорівнює DW=1,7599. Модель з такими показниками можна назвати адекватною.

6. Прогнозування на основі адитивної моделі.

Прогноз буде здійснюватися на наступні 2 квартали, для збереження максимальної точності. В мультиплікативній моделі для отримання значення прогнозу на момент часу в майбутньому потрібно знайти добуток значення скорегованої сезонної варіації Si та тренду T для прогнозуючого моменту часу.

$F_{14}=Si_{14}*T_{14}$= 0,935932*456,129= 426,9057;

$F_{15}=Si_{15}*T_{15}$= 0,797938*475,683= 379,5658;

Так, як і в адитивній моделі залишилося лише перевірити точність прогнозу за допомогою середньої абсолютної похибки (САО) та середнього відносних помилок (СООП). Для даної моделі значення похибок:

САО=7,21;

СООП=2,38%;

Якщо значення СООП не перевищує 5% то прогноз можна вважати точним.

Висновки до розділу 2

Розроблено алгоритм, який дає змогу магазину або підприємству отримати дані, що показують як об’єм продажу змінюється залежно від кварталу за річний цикл. Такі дані дають змогу зрозуміти який товар і в який період року краще продавати для максимізації прибутку.

Також алгоритм дає змогу здійснити прогноз, на який об’єм продажу приблизно магазин може розраховувати в наступні квартали. Отримавши прогноз магазин може виготовити або замовити оптимальну кількість товару, що дозволяє зекономити на матеріалах та логістиці, та таким чином отримати максимальний прибуток.

Методом ковзного середнього здійснено вирівнювання вхідного часового ряду та отримано оцінку сезонної компоненти для адитивної та мультиплікативної моделей. Виконано аналітичне вирівнювання рівнів ряду та оцінено параметри моделей**.** Також здійснена перевірка адекватності адитивної та мультиплікативної моделей. З отриманих даних побудовано прогноз об’єму продажу умовного інтернет-магазину взуття.

Розроблений алгоритм сприяє полегшенню аналізу залежності об’єму продажу від сезону, а також дозволяє отримати приблизний прогноз об’єму продажу в майбутньому.

РОЗДІЛ 3. ВИБІР СЕРЕДОВИЩА ТА СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Вибір платформи, середовища та мови програмування для реалізації додатку

В рамках даної кваліфікаційної магістерської роботи було вирішено створити класичний віконний додаток з інтерфейсом користувача для операційної системи **Windows**, на мові **C#**. Проєкт розроблявся в інтегрованому середовищі розробки (IDE) **Visual Studio** версії “community 2022” на платформі **.NET Framework**.

C# (вимовляється як "C-sharp") - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона є однією з ключових мов програмування для платформи .NET і використовується для розробки різноманітних програм, від настільних додатків до веб-застосунків та мобільних додатків. Мова програмування C# володіє рядом переваг, які зробили її популярною серед розробників. Основні переваги цієї мови програмування:

1. **Синтаксис**: C# має синтаксис, що вивчається та схожий на інші мови програмування, такі як C, C++ та Java. Це робить його доступним для багатьох розробників.
2. **Інтеграція з .NET Framework/.NET Core/.NET 5**: C# є основною мовою для платформи .NET, яка надає багатофункціональний фреймворк для розробки різноманітних застосунків. Ця інтеграція дозволяє розробникам використовувати широкий спектр бібліотек і сервісів, що постачаються разом з .NET.
3. **Об'єктно-Орієнтований підхід**: C# використовує об'єктно-орієнтований підхід, що полегшує створення модульного та структурованого коду. Це сприяє підтримці сучасних архітектурних парадигм, таких як SOLID.
4. **Безпека та Керування Пам'яттю**: C# вбудовує систему керування пам'яттю, що дозволяє уникати багатьох типових помилок, пов'язаних із використанням пам'яті, і забезпечує високий рівень безпеки виконання.

5. **Багатозадачність та Асинхронність**:Мова підтримує асинхронне програмування, що дозволяє ефективно використовувати ресурси системи та підтримує многозадачність для паралельного виконання завдань.

6. **Багатофункціональні Бібліотеки та Фреймворки**: Разом з C# постачаються різні багатофункціональні бібліотеки та фреймворки, такі як ASP.NET для веб-розробки, Windows Forms для настільних додатків, WPF для розробки графічних інтерфейсів, Entity Framework для роботи з базами даних, та інші.

7. **Широке Використання в Розробці Ігор**: C# є популярним вибором для розробки відеоігор, зокрема завдяки фреймворку Unity, який використовується для створення ігор для різних платформ.

8. **Підтримка Крос-Платформеного Розвитку**: З виходом .NET Core та .NET 5, C# отримала підтримку крос-платформеного розвитку, що дозволяє розробникам створювати застосунки для різних операційних систем.

9. **Інтеграція з Іншими Мовами та Технологіями**: Мова C# добре інтегрується з іншими мовами програмування, такими як C++ та Visual Basic. Також існує підтримка інтеграції з різними технологіями, такими як COM та P/Invoke.

Сьогодні C# є однією з основних мов програмування для розробки застосунків на платформі Microsoft, включаючи різні види програм, від настільних до веб-сервісів та мобільних додатків.

В середині 1990-х років Microsoft визначила потребу в новому підході до розробки програмного забезпечення. Прагнучи створити більш безпечну, продуктивну та масштабовану платформу, компанія розпочала роботу над проєктом, який став відомий як ".NET". Робота над мовою C# офіційно розпочалася у 1999 році під керівництвом архітектора програмного забезпечення Андерса Гейлсберга (Anders Hejlsberg). Він вже був відомий своєю роботою над мовою Turbo Pascal та Delphi. Мова C# була представлена на конференції Microsoft Professional Developers Conference (PDC) в липні 2000 року. Її цільовою аудиторією були розробники, які працюють з платформою Windows. З часом ця

мова програмування отримала ряд оновлень і покращень, включаючи нові функції, підтримку асинхронного програмування, введення LINQ (Language Integrated Query) та інші.

Visual Studio - це інтегроване середовище розробки (Integrated Development Environment, IDE) від Microsoft, яке надає зручний та потужний набір інструментів для розробки програмного забезпечення. Що стосується актуальності, Visual Studio залишається однією з провідних IDE для роботи з платформами Microsoft, такими як .NET Framework, .NET Core, Azure, Xamarin і багато інших. Основні переваги IDE Visual Studio:

Підтримка Багатьох Мов: Visual Studio підтримує розробку на різних мовах програмування, таких як C#, VB.NET, F#, C++, Python, JavaScript, та інші. Це робить його універсальним середовищем для різноманітних типів розробки.

Інтеграція з .NET: Враховуючи, що Visual Studio розроблено Microsoft, воно має потужну інтеграцію з технологіями .NET Framework та .NET Core. Це дозволяє легко створювати веб-застосунки, настільні програми, служби та інші проєкти на основі .NET.

Інструменти для Швидкого Розгортання: Visual Studio дозволяє легко розгортати застосунки в хмарні сервіси Azure або на інші сервери. Інтеграція з Azure DevOps дозволяє автоматизувати процеси розгортання та CI/CD.

Інтеграція з Visual Studio Code: Visual Studio і Visual Studio Code (легка, крос-платформена версія) взаємодіють, що дозволяє розробникам використовувати обидва середовища в залежності від їхніх потреб.

Підтримка Останніх Технологій: Visual Studio регулярно оновлюється для підтримки останніх версій мов програмування, фреймворків та технологій. Наприклад, вона має повну підтримку .NET 6, який є однією з найновіших версій платформи .NET.

Windows Forms та WPF Розробка: Visual Studio надає інструменти для розробки настільних додатків за допомогою Windows Forms і Windows Presentation Foundation (WPF), що спрощує створення графічно насичених та ефективних застосунків для Windows.

ASP.NET Розробка: Для веб-розробки Visual Studio має вбудовану підтримку для створення веб-застосунків на платформі ASP.NET. Ви можете легко розробляти та тестувати веб-застосунки прямо у середовищі Visual Studio.

Розширені Засоби Відлагодження (Debugging) та Профілювання: Visual Studio надає розширені засоби для відлагодження коду, включаючи можливість крокування по коду, встановлення точок зупинки, перегляд стеку викликів та інші корисні функції. Профілювання коду дозволяє виявляти його ефективність та виявляти можливі точки оптимізації.

Вбудовані Інструменти Роботи З Кодом (Code Analysis): Visual Studio має інтегровані інструменти для аналізу коду, виявлення помилок, підказок та автоматичних виправлень. Це полегшує підтримку кодової якості та стилістичних правил.

Інструменти для Колективної Розробки: Visual Studio Team Services надає інструменти для колективної розробки, такі як система керування версіями, засоби спільної роботи, засоби для автоматизації CI/CD та інші.

Ідеї та технології, які визначають Visual Studio, є актуальними для розробки програмного забезпечення на платформах Microsoft, і вона залишається популярною серед розробників у багатьох галузях.

Сама компнія Microsoft виділяє такі особливості свого продукту:

Розробка коду. Інтегроване середовище розробки Visual Studio надає безліч функцій, які спрощують написання та управління кодом з впевненістю. Наприклад, завдяки засобам розробки на основі штучного інтелекту, таким як GitHub Copilot та IntelliCode, код можна швидко та точно покращити. Лампочки пропонують дії для розширення коду, а також можна згортати або розгортати блоки коду для структурування. Впорядкування та огляд коду стає можливим завдяки Обозрювачу рішень, що включає код, впорядкований за файлами або представленням класів.

Збірка додатку. Ви можете компілювати та створювати додатки, щоб відразу створювати збірки та тестувати їх у відлагоджувачі. Ви можете запускати збірки з декількома процесорами для проєктів на C++ та C#. Visual Studio також

надає кілька налаштовуваних варіантів для створення додатків. Ви можете створити власну настроювану конфігурацію збірки, додатково до вбудованих конфігурацій, приховати певні попередження або збільшити вивідні дані збірки.

Відладка коду. Вбудована відладка в Visual Studio дозволяє легко виконувати відладку, профілювання та діагностику. Ви виконуєте пошагову навігацію по коду, переглядаєте значення, збережені в змінних, встановлюєте контроль над змінними, щоб бачити, коли значення змінюються, перевіряєте шлях виконання коду та інші методи відладки коду під час його виконання.

Тестування коду. Ви можете написати високоякісний код завдяки розширеним засобам тестування в Visual Studio. Модульні тести дозволяють розробникам та тестувальникам швидко виявляти помилки логіки в коді. Ви можете аналізувати, скільки коду ви тестуєте, і бачити миттєві результати у наборі тестів або знати вплив кожної зміни, яку ви вносите, завдяки розширеним можливостям, які тестують код під час введення.

Управління версіями. За допомогою інтегрованих засобів Git у Visual Studio ви можете клонувати, створювати або відкривати власні репозиторії. В вікні інструментів Git є все необхідне для фіксації та відправлення змін у коді, управління гілками та вирішення конфліктів злиття. Якщо у вас є обліковий запис GitHub, ви можете керувати цими репозиторіями безпосередньо в Visual Studio.

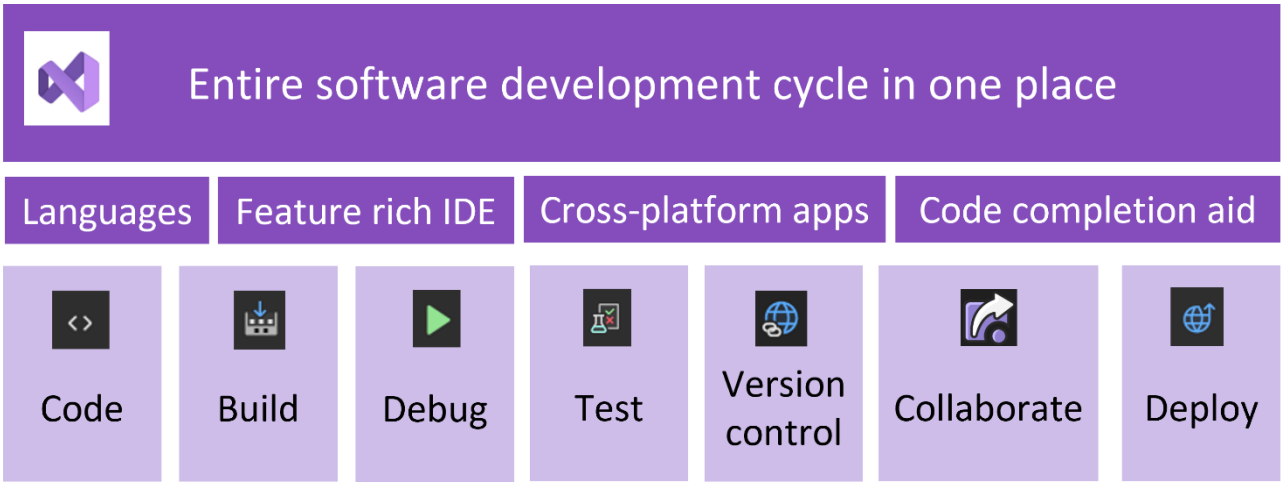


Рис. 3.1 Ідейне зображення IDE Visual Studio від Microsoft

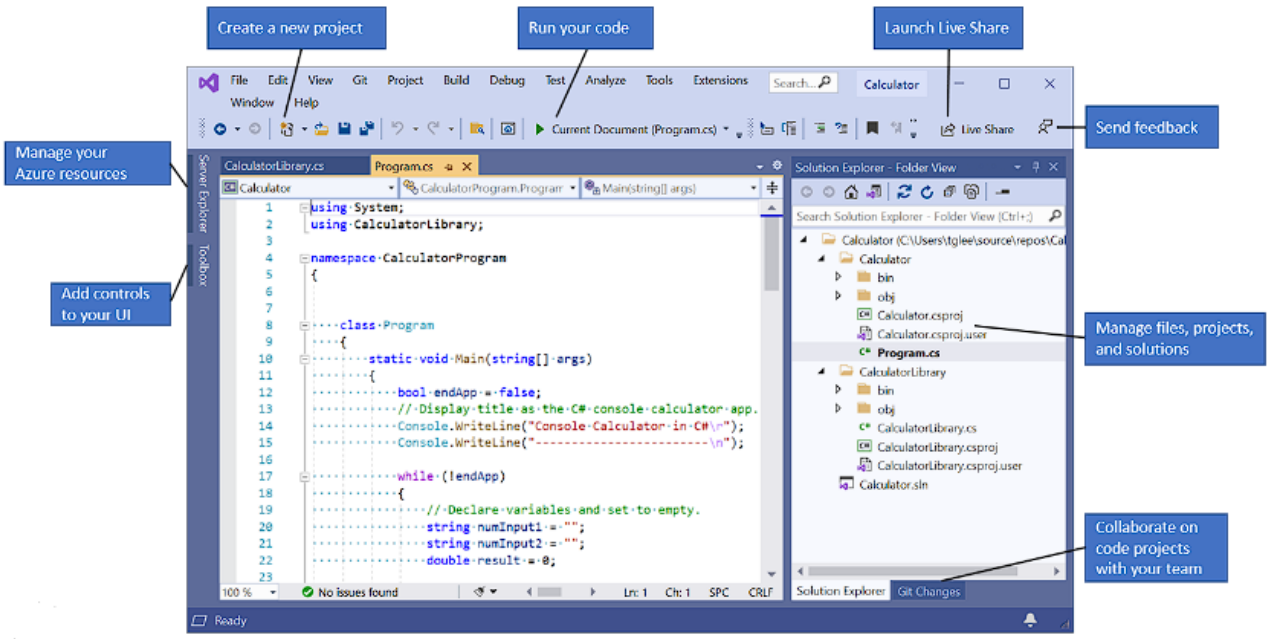


Рис. 3.2 Огляд інтерфейсу IDE

Перша випущена версія Visual Studio 97 з’явилася в 1997 році, в якій вперше були об'єднані різні засоби розробки програмного забезпечення. Вона була випущена у двох версіях — Professional та Enterprise, і включала в себе Visual Basic 5.0, Visual C++ 5.0, Visual J++ 1.1, Visual FoxPro 5.0, а також вперше з'явилася середа розробки ASP — Visual InterDev. Visual Studio 97 була першою спробою Microsoft створити єдине середовище для розробки на різних мовах програмування: Visual C++, Visual J++, Visual InterDev та MSDN використовували одне середовище, яке називалося Developer Studio. Visual Basic і Visual FoxPro використовували окремі середи розробки.

Visual Studio отримує оновлення та розвивається і по сьогодні. Остання версія вийшла в 2022 році.

.NET Framework - це платформа розробки програмного забезпечення, яка була розроблена компанією Microsoft. Вона надає середовище для виконання, розгортання та управління програмами на мовах програмування, таких як C#, Visual Basic, інших мовах .NET та навіть деяких мовах сторонніх розробників.

Основні компоненти та характеристики .NET Framework включають:

Common Language Runtime (CLR) - це виконавче середовище, яке виконує код, написаний на різних мовах програмування (.NET-сумісних), у байт-коді.

Воно відповідає за автоматизацію управління пам'яттю, обробку виключень, зборку сміття та інші операції, що допомагають розробникам створювати надійне та ефективне програмне забезпечення.

Бібліотека класів .NET (BCL) - це колекція готових до використання класів та бібліотек, які надають функціональність для великої кількості завдань, включаючи роботу з мережами, базами даних, роботу з XML, роботу з файлами, шифрування та багато іншого.

Мови програмування: .NET підтримує різні мови програмування, такі як C#, VB.NET, F#, C++ та інші. Вони всі компілюються в спільний мовний інтерфейс (Common Intermediate Language - CIL), який CLR потім виконує.

ASP.NET - це фреймворк для розробки веб-застосунків, який забезпечує спосіб створення веб-сайтів, веб-служб та інших веб-застосунків.

Windows Forms та WPF: Для розробки настільних застосунків .NET пропонує Windows Forms (для традиційних десктопних застосунків) та Windows Presentation Foundation (WPF - для більш сучасних, гнучких та графічно насичених застосунків).

Entity Framework - це ORM (Object-Relational Mapping) фреймворк, який дозволяє робити взаємодію з базами даних за допомогою об'єктно-орієнтованого коду.

LINQ (Language Integrated Query) - це мова виразів запитань, яка дозволяє виконувати запити до даних безпосередньо в коді, що значно полегшує роботу з колекціями та даними.

.NET Framework використовується для розробки різноманітних програм, від маленьких веб-сайтів до великих корпоративних застосунків. Важливо зазначити, що на початку 2020 року Microsoft анонсувала, що .NET Framework не буде розвиватися далі, і її майбутнє належить .NET 5 і вище, які представляють собою більш єдину та масштабовану платформу для розробки.

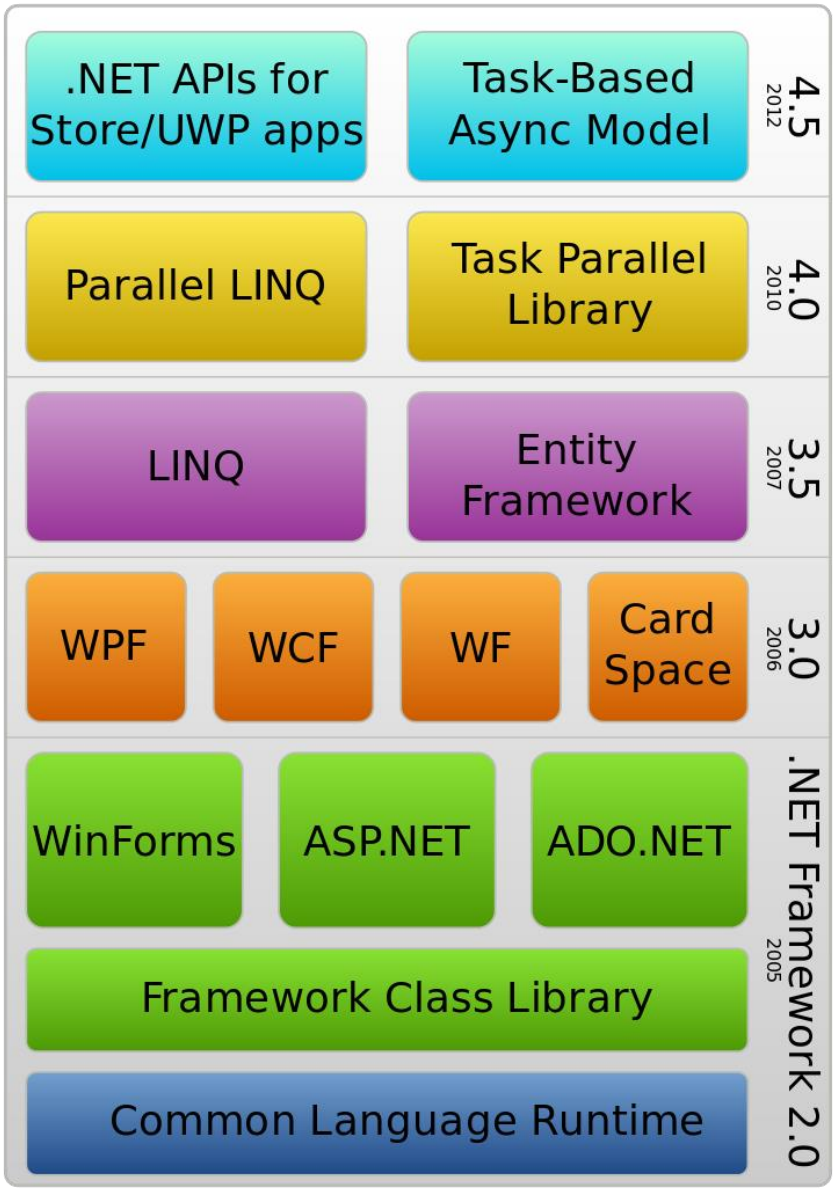


Рис. 3.3 Компоненти .NET Framework до версії 4.5 включно

3.2. Структура проєкту

Реалізований в Visual Studio 2020 проєкт має в своїй структурі створений автоматично обов'язковий клас Program, та створені мною головна форма MainScreen всередині якої відбуваються головні розрахунки, тека для дочірніх форм, що містить 4 форми для відображення табличних значень та форму WelcomeScreen. Також було створено 3 користувальницьких класів TimeLines – зберігає та надає доступ до розрахункових даних з любого місця програми, ThemeColor – забезпечує налаштування кольорів для елементів інтерфейсу та ExcelHelper – класс, що забезпечує зв'язок з зовнішньою програмою Microsoft Excel, для створення зручного звіту результатів обчислення.

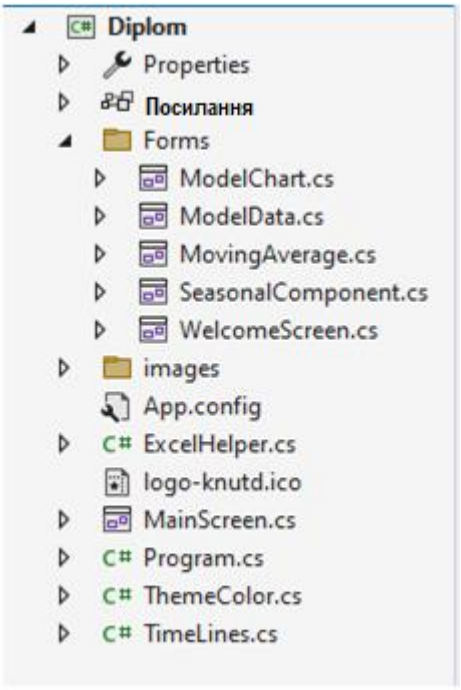


Рис. 3.4 Структура проєкту

3.3. Програмна реалізація алгоритму прогнозу

Головним завданням цієї кваліфікаційної магістерської роботи є програмна реалізація алгоритму прогнозування об'ємів продажу для інтернет-магазинів взуття розглянутого в другому розділі роботи. Дотримуючись кроків описаних вище, були запрограмовані такі методи розрахунку даних:

```
Ссылок: 2
static (int salesIndex, int[] salesIndexes, double[] salesValues) InputIformation(string timeLineRow)
{
    string[] subStringsOfSaleValues = timeLineRow.Split(' ');
    double[] salesValues = new double[subStringsOfSaleValues.Length];
    for (int i = 0; i < subStringsOfSaleValues.Length; i++)
    {
        salesValues[i] = Convert.ToDouble(subStringsOfSaleValues[i]);
    }
    int salesIndex = salesValues.Length;
    int[] salesIndexes = new int[salesIndex];
    for (int i = 0; i < salesIndex; i++)
    {
        salesIndexes[i] = i + 1;
    }
    return (salesIndex, salesIndexes, salesValues);

}
```

Рис. 3.5 Метод InputInformation

Метод InputInformation отримує в вигляді аргументу рядок timeLineRow, що являє собою фактичні рівні часового ряду введені користувачем на формі MainScreen. Після розрахунків метод повертає ціле число salesIndex, що дорівнює спільній кількості спостережень моментів часу(кварталів), масив чисел

SalesIndexes, що містить в собі порядкові номери рівнів ряду t та масив SalesValues що містить в собі фактичні значення рівнів ряду y_t.

Медот MovingAverageForAdditiveModel програмно реалізує метод ковної середньої за 4-ма кварталами для адитивної моделі, приймаючи вже описані змінні salesIndex, SalesIndexes та SalesValues, як аргументи та вертаючи значення ковзного середнього, центрованої ковзної середньої та оцінки сезонної компоненти в масивах MovingAverageForFourQuarters, CentralMovingAverageValues, MovingAverageEstimation.

```
static (double[] movingAverageForFourQuarters, double[] centralMovingAverageValues, double[] movingAverageEstimation)
Ссылок: 1
MovingAverageForAdditiveModel(int SalesIndex, int[] SalesIndexes, double[] SalesVal)
{
    double[] SalesValues = new double[SalesIndex];
    SalesValues = SalesVal;
    double[] MovingAverageForFourQuarters = new double[SalesIndex];
    double SumOfQuarters;
    const int QUARTERS_NUMBER = 4;

    for (int i = 0; i <= SalesIndex - QUARTERS_NUMBER; i++)
    {
        SumOfQuarters = SalesValues[i] + SalesValues[i + 1] + SalesValues[i + 2] + SalesValues[i + 3];
        MovingAverageForFourQuarters[i + 2] = SumOfQuarters / QUARTERS_NUMBER;
    }

    double[] CentralMovingAverageValues = new double[SalesIndex];
    int NumberOfQuartersForCentralMovingAverage = 2;
    double SumOfCentralMovingAverageValues;
    for (int i = 2; i <= MovingAverageForFourQuarters.Length - 3; i++)
    {
        SumOfCentralMovingAverageValues = (MovingAverageForFourQuarters[i] +
        MovingAverageForFourQuarters[i + 1]);
        CentralMovingAverageValues[i] = SumOfCentralMovingAverageValues
             / NumberOfQuartersForCentralMovingAverage;
    }
    double[] MovingAverageEstimation = new double[SalesIndex];
    for (int i = 2; i <= MovingAverageForFourQuarters.Length - 3; i++)
    {
        MovingAverageEstimation[i] = SalesValues[i] - CentralMovingAverageValues[i];
    }
    return (MovingAverageForFourQuarters, CentralMovingAverageValues, MovingAverageEstimation);

}
```

Рис. 3.6 Метод MovingAverageForAdditiveModel

Метод MovingAverageForMultiplicativeModel приймає, та повертає аналогічні значення, виконуючи розрахунок для мультиплікативної моделі.

```
static (double[] movingAverageForFourQuarters, double[] centralMovingAverageValues, double[] movingAverageEstimation)
    Ссылок: 1
    MovingAverageForMultiplicativeModel(int SalesIndex, int[] SalesIndexes, double[] SalesVal)
{
    double[] SalesValues = new double[SalesIndex];
    SalesValues = SalesVal;
    double[] MovingAverageForFourQuarters = new double[SalesIndex];
    double SumOfQuarters;
    const int QUARTERS_NUMBER = 4;

    for (int i = 0; i <= SalesIndex - QUARTERS_NUMBER; i++)
    {
        SumOfQuarters = SalesValues[i] + SalesValues[i + 1] + SalesValues[i + 2] + SalesValues[i + 3];
        MovingAverageForFourQuarters[i + 2] = SumOfQuarters / QUARTERS_NUMBER;
    }

    double[] CentralMovingAverageValues = new double[SalesIndex];
    int NumberOfQuartersForCentralMovingAverage = 2;
    double SumOfCentralMovingAverageValues;
```

```
        for (int i = 2; i <= MovingAverageForFourQuarters.Length - 3; i++)
        {
            SumOfCentralMovingAverageValues = (MovingAverageForFourQuarters[i] +
            MovingAverageForFourQuarters[i + 1]);
            CentralMovingAverageValues[i] = SumOfCentralMovingAverageValues
                / NumberOfQuartersForCentralMovingAverage;
        }
        double[] MovingAverageEstimation = new double[SalesIndex];
        for (int i = 2; i <= MovingAverageForFourQuarters.Length - 3; i++)
        {
            MovingAverageEstimation[i] = SalesValues[i] / CentralMovingAverageValues[i];
        }
        return (MovingAverageForFourQuarters, CentralMovingAverageValues, MovingAverageEstimation);

    }
```

Рис. 3.7 Медот MovingAverageForMultiplicativeModel

Методи SeaosonalComponentCalculationForAdditiveModel і SeaosonalComponentCalculationForMultiplicativeModel розраховують скореговану сезонну варіацію для своїх моделей.

```
static (double[,] seasonalComponent, double[] averageValueOfSeasonalComponents, double sumOfaverageValueOfSeasonalComponents,
    double correctiveFactor, double[] correctedSeasonalVariation)
    Ссылок: 1
    SeasonalComponentCalculationForAdditiveModel(double[] MAE)
{
    double[] MovingAverageEstimation = MAE;
    const int QUARTERS_NUMBER = 4;
    int quartersRepeat;
    int test = MovingAverageEstimation.Length % QUARTERS_NUMBER;
    if (MovingAverageEstimation.Length % QUARTERS_NUMBER == 0 || MovingAverageEstimation.Length % QUARTERS_NUMBER == 1)
    {
        quartersRepeat = MovingAverageEstimation.Length / QUARTERS_NUMBER;
    }
    else quartersRepeat = MovingAverageEstimation.Length / QUARTERS_NUMBER + 1;

    double[,] seasonalComponent = new double[quartersRepeat, QUARTERS_NUMBER];
    int counter = 0;
    for (int i = 0; i < quartersRepeat; i++)
    {
        for (int j = 0; j < QUARTERS_NUMBER; j++)
        {
            if (counter < MovingAverageEstimation.Length)
            {
                seasonalComponent[i, j] = MovingAverageEstimation[counter];
            }
            else seasonalComponent[i, j] = 0;
            counter++;
        }
    }
    double[] sumOfSeasonalComponents = new double[QUARTERS_NUMBER];
    double[] averageValueOfSeasonalComponents = new double[QUARTERS_NUMBER];
    double sumOfaverageValueOfSeasonalComponents = 0;

    int averageCounter = 0;
    for (int i = 0; i < QUARTERS_NUMBER; i++)
    {
        for (int j = 0; j < quartersRepeat; j++)
        {
            if (seasonalComponent[j, i] != 0)
            {
                sumOfSeasonalComponents[i] += seasonalComponent[j, i];
                averageCounter++;
            }
        }
        averageValueOfSeasonalComponents[i] = sumOfSeasonalComponents[i] / averageCounter;
        averageCounter = 0;
        sumOfaverageValueOfSeasonalComponents += averageValueOfSeasonalComponents[i];

    }
    double correctiveFactor = sumOfaverageValueOfSeasonalComponents / QUARTERS_NUMBER;
    double[] correctedSeasonalVariation = new double[QUARTERS_NUMBER];
    for (int i = 0; i < QUARTERS_NUMBER; i++)
    {
        correctedSeasonalVariation[i] = averageValueOfSeasonalComponents[i] - correctiveFactor;
    }
    return (seasonalComponent, averageValueOfSeasonalComponents, sumOfaverageValueOfSeasonalComponents,
        correctiveFactor, correctedSeasonalVariation);
```

Рис. 3.8 Метод SeaosonalComponentCalculationForAdditiveModel

```
static (double[,] seasonalComponent, double[] averageValueOfSeasonalComponents, double sumOfaverageValueOfSeasonalComponents,
    double correctiveFactor, double[] correctedSeasonalVariation)
    Ссылок: 1
    SeasonalComponentCalculationForMultiplicativeModel(double[] MAE)
{
    double[] MovingAverageEstimation = MAE;
    const int QUARTERS_NUMBER = 4;
    int quartersRepeat = MovingAverageEstimation.Length / QUARTERS_NUMBER;
    double[,] seasonalComponent = new double[quartersRepeat, QUARTERS_NUMBER];
    int counter = 0;
    for (int i = 0; i < quartersRepeat; i++)
    {
        for (int j = 0; j < QUARTERS_NUMBER; j++)
        {

            seasonalComponent[i, j] = MovingAverageEstimation[counter];
            counter++;
        }
    }
    double[] sumOfSeasonalComponents = new double[QUARTERS_NUMBER];
    double[] averageValueOfSeasonalComponents = new double[QUARTERS_NUMBER];
    double sumOfaverageValueOfSeasonalComponents = 0;

    int averageCounter = 0;
    for (int i = 0; i < QUARTERS_NUMBER; i++)
    {
        for (int j = 0; j < quartersRepeat; j++)
        {
            if (seasonalComponent[j, i] != 0)
            {
                sumOfSeasonalComponents[i] += seasonalComponent[j, i];
                averageCounter++;
            }
        }
        averageValueOfSeasonalComponents[i] = sumOfSeasonalComponents[i] / averageCounter;
        averageCounter = 0;
        sumOfaverageValueOfSeasonalComponents += averageValueOfSeasonalComponents[i];

    }
    double correctiveFactor = QUARTERS_NUMBER / sumOfaverageValueOfSeasonalComponents;
    double[] correctedSeasonalVariation = new double[QUARTERS_NUMBER];
    for (int i = 0; i < QUARTERS_NUMBER; i++)
    {
        correctedSeasonalVariation[i] = averageValueOfSeasonalComponents[i] * correctiveFactor;
    }
    return (seasonalComponent, averageValueOfSeasonalComponents, sumOfaverageValueOfSeasonalComponents,
        correctiveFactor, correctedSeasonalVariation);
}
```

Рис. 3.9 Метод SeaosonalComponentCalculationForMultiplicativeModel

Методи AdditiveModel (рис. 3.14) і MultiplicativeModel (рис. 3.15) групують в собі всі попередні дані, знаходять значення лінії тренду T та здійснюють прогнози для відповідних моделей. В тілі цих методів також викликаються інші розрахункові методи такі як: TrendLineCalculation для знаходження коефіцієнтів рівняння тренду методами найменших квадратів та Крамера, DeterminationCoefficientCalculation для пошуку коефіцієнта детермінації R^2, DurbinWatsonStatisticCalculation для знаходження критерію Дарбіна-Уотсона DW, AverageAbsoluleDebiatioCalculation для пошуку САО та AverageRelativeErrorCalculation для пошуку СООП.

```
Ссылок: 2
static (double, double) TrendLineCalculation(int salesIndex, int[] SalesIndexes, double[] valuesWOSeasonalVariation)
{
    int sumOfIndexes = 0;
    for (int i = 0; i < salesIndex; i++)
    {
        sumOfIndexes += SalesIndexes[i];
    }
    double sumOfValues = 0;
    for (int i = 0; i < salesIndex; i++)
    {
        sumOfValues += valuesWOSeasonalVariation[i];
    }
    int[] squareOfIndexes = new int[salesIndex];
    int sumOfSquareOfIndexes = 0;
    for (int i = 0; i < salesIndex; i++)
    {
        squareOfIndexes[i] = (int)Math.Pow(SalesIndexes[i], 2);
        sumOfSquareOfIndexes += squareOfIndexes[i];
    }
    double[] productOfIndexesAndValues = new double[salesIndex];
    double sumOfProducts = 0;
    for (int i = 0; i < salesIndex; i++)
    {
        productOfIndexesAndValues[i] = SalesIndexes[i] * valuesWOSeasonalVariation[i];
        sumOfProducts += productOfIndexesAndValues[i];
    }
    double[,] linearEquationMatrix = { { salesIndex, sumOfIndexes, sumOfValues }, { sumOfIndexes, sumOfSquareOfIndexes, sumOfProducts } };
    double matrixDeterminant1 = linearEquationMatrix[0, 0] * linearEquationMatrix[1, 1] - linearEquationMatrix[1, 0] * linearEquationMatrix[0, 1];
    double matrixDeterminant2 = linearEquationMatrix[0, 2] * linearEquationMatrix[1, 1] - linearEquationMatrix[1, 2] * linearEquationMatrix[0, 1];
    double matrixDeterminant3 = linearEquationMatrix[0, 0] * linearEquationMatrix[1, 2] - linearEquationMatrix[1, 0] * linearEquationMatrix[0, 2];
    double coefficient1 = matrixDeterminant2 / matrixDeterminant1;
    double coefficient2 = matrixDeterminant3 / matrixDeterminant1;
    return (coefficient1, coefficient2);

}
```

Рис. 3.10 Метод TrendLineCalculation

```
Ссылок: 2
static double DeterminationCoefficientCalculation(int salesIndex, double[] salesValues, double[] trendValues, double[] errorsSquare)
{
    double[] salesValuesMinusTrendValues = new double[salesIndex];
    double[] squareOfSalesValuesMinusTrendValues = new double[salesIndex];
    double sumOfSquares = 0;
    double sumOfErrorSquares = 0;
    for (int i = 0; i < salesIndex; i++)
    {
        salesValuesMinusTrendValues[i] = salesValues[i] - trendValues[i];
        squareOfSalesValuesMinusTrendValues[i] = Math.Pow(salesValuesMinusTrendValues[i], 2);
        sumOfSquares += squareOfSalesValuesMinusTrendValues[i];
        sumOfErrorSquares += errorsSquare[i];
    }
    double determinantCoefficient = 1 - (sumOfErrorSquares / sumOfSquares);

    return determinantCoefficient;
}
Ссылок: 2
static double DurbinWatsonStatisticCalculation(int salesIndex, double[] valuesWOSeasonalVariation, double[] trendValues)
{
    double[] salesValuesMinusTrendValues = new double[salesIndex];
    double[] squareOfSalesValuesMinusTrendValues = new double[salesIndex];
    double sumOfSquares = 0;
    for (int i = 0; i < salesIndex; i++)
    {
        salesValuesMinusTrendValues[i] = valuesWOSeasonalVariation[i] - trendValues[i];
        squareOfSalesValuesMinusTrendValues[i] = Math.Pow(salesValuesMinusTrendValues[i], 2);
        sumOfSquares += squareOfSalesValuesMinusTrendValues[i];
    }
    double[] DWBufferValues = new double[salesIndex - 1];
    double[] squareOfDWBufferValues = new double[salesIndex - 1];
    double sumOfSquaresDWBufferValues = 0;
    for (int i = 1; i < salesIndex; i++)
    {
        DWBufferValues[i - 1] = salesValuesMinusTrendValues[i - 1] - salesValuesMinusTrendValues[i];
        squareOfDWBufferValues[i - 1] = Mat
        sumOfSquaresDWBufferValues += squar
    }
    double durbinWatsonStatistic = sumOfSquaresDWBufferValues / sumOfSquares;
    return durbinWatsonStatistic;
}
```

(локальная переменная) double[] salesValuesMinusTrendValues

Рис. 3.11 Методи DeterminationCoefficientCalculation і DurbinWatsonStatisticCalculation

```
Ссылок: 2
static double AverageAbsoluteDeviationCalculation(int salesIndex, double[] errorsModule)
{
    double averageAbsoluteDeviationtmp = 0;
    for (int i = 0; i < salesIndex; i++)
    {
        averageAbsoluteDeviationtmp += errorsModule[i];
    }
    double averageAbsoluteDeviation = averageAbsoluteDeviationtmp / salesIndex;
    return averageAbsoluteDeviation;
}
Ссылок: 2
```

Рис. 3.12 Метод AverageAbsoluleDebiatioCalculation

```
Ссылок: 2
static double AverageRelativeErrorCalculation(int salesIndex, double[] salesValues, double[] errorsModule)
{
    double averageRelativeErrortmp = 0;
    for (int i = 0; i < salesIndex; i++)
    {
        averageRelativeErrortmp += (errorsModule[i] / salesValues[i]) * 100;
    }
    double averageRelativeError = averageRelativeErrortmp / salesIndex;
    return averageRelativeError;
}
Ссылок: 1
```

Рис. 3.13 Метод AverageRelativeErrorCalculation

```
static (int numberOfForecastingValues, double[] forecastingValuesIndexes, double[] seasonalVariation,
    double[] valuesWOSeasonalVariation, double coeficient1, double coeficient2, double[] trendValues,
    double[] trendValuesWithSeasonalVariation, double[] errorsE,
    double[] errorsModule, double[] errorsSquare, double determinationCoefficient,
    double durbinWatsonStatistic, double averageAbsoluteDeviation,
    Ссылок: 1
    double averageRelativeError) AdditiveModel(int SalesIndex, int[] SalesIndexes, double[] SalesValues,
    double[] correctedSeasonalVariation, int forecastingNumber)
{
    int numberOfForecastingValues = SalesIndex + forecastingNumber;
    double[] forecastingValuesIndexes = new double[numberOfForecastingValues];
    for (int i = 0; i < numberOfForecastingValues; i++)
    {
        forecastingValuesIndexes[i] = i + 1;
    }
    double[] seasonalVariation = new double[numberOfForecastingValues];
    int seasonalVariatioCounter = 0;
    for (int i = 0; i < numberOfForecastingValues; i++)
    {
        if ((seasonalVariatioCounter) % correctedSeasonalVariation.Length == 0)
        {
            seasonalVariatioCounter = 0;
        }
        seasonalVariation[i] = correctedSeasonalVariation[seasonalVariatioCounter];
        seasonalVariatioCounter++;

    }
    double[] valuesWOSeasonalVariation = new double[SalesIndex];
    for (int i = 0; i < SalesIndex; i++)
    {
        valuesWOSeasonalVariation[i] = SalesValues[i] - seasonalVariation[i];
    }
    double[] trendValues = new double[numberOfForecastingValues];
    var coeficients = TrendLineCalculation(SalesIndex, SalesIndexes, valuesWOSeasonalVariation);
    double coeficient1 = coeficients.Item1;
    double coeficient2 = coeficients.Item2;
    for (int i = 0; i < numberOfForecastingValues; i++)
    {
        trendValues[i] = (coeficient2 * forecastingValuesIndexes[i]) + coeficient1;
    }
    double[] trendValuesWithSeasonalVariation = new double[numberOfForecastingValues];
    for (int i = 0; i < numberOfForecastingValues; i++)
    {
        trendValuesWithSeasonalVariation[i] = trendValues[i] + seasonalVariation[i];
    }
    double[] errorsE = new double[SalesIndex];
    double[] errorsModule = new double[SalesIndex];
    double[] errorsSquare = new double[SalesIndex];
    for (int i = 0; i < SalesIndex; i++)
    {
        errorsE[i] = SalesValues[i] - trendValuesWithSeasonalVariation[i];
        errorsModule[i] = Math.Abs(errorsE[i]);
        errorsSquare[i] = Math.Pow(errorsE[i], 2);
    }
    double determinationCoefficient = DeterminationCoefficientCalculation(SalesIndex, SalesValues, trendValues, errorsSquare);
    double durbinWatsonStatistic = DurbinWatsonStatisticCalculation(SalesIndex, valuesWOSeasonalVariation, trendValues);
    double averageAbsoluteDeviation = AverageAbsoluteDeviationCalculation(SalesIndex, errorsModule);
    double averageRelativeError = AverageRelativeErrorCalculation(SalesIndex, SalesValues, errorsModule);
    return (numberOfForecastingValues, forecastingValuesIndexes, seasonalVariation, valuesWOSeasonalVariation,
   coeficient1, coeficient2, trendValues, trendValuesWithSeasonalVariation, errorsE,
     errorsModule, errorsSquare, determinationCoefficient, durbinWatsonStatistic, averageAbsoluteDeviation,
    averageRelativeError);

}
```

Рис. 3.14 Метод AdditiveModel

```
static (int numberOfForecastingValues, double[] forecastingValuesIndexes, double[] seasonalVariation,
static (int numberOfForecastingValues, double[] forecastingValuesIndexes, double[] seasonalVariation,
    double[] valuesWOSeasonalVariation,double coeficient1, double coeficient2, double[] trendValues,
    double[] trendValuesWithSeasonalVariation, double[] errorsE,double[] errorsModule, double[] errorsSquare,
    double determinationCoefficient, double durbinWatsonStatistic, double averageAbsoluteDeviation,
    Ссылок: 1
    double averageRelativeError) MultiplicativeModel(int SalesIndex, int[] SalesIndexes,
    double[] SalesValues, double[] correctedSeasonalVariation, int forecastingNumber)
{
    int numberOfForecastingValues = SalesIndex + forecastingNumber;
    double[] forecastingValuesIndexes = new double[numberOfForecastingValues];
    for (int i = 0; i < numberOfForecastingValues; i++)
    {
        forecastingValuesIndexes[i] = i + 1;
    }
    double[] seasonalVariation = new double[numberOfForecastingValues];
    int seasonalVariatioCounter = 0;
    for (int i = 0; i < numberOfForecastingValues; i++)
    {
        if ((seasonalVariatioCounter) % correctedSeasonalVariation.Length == 0)
        {
            seasonalVariatioCounter = 0;
        }
        seasonalVariation[i] = correctedSeasonalVariation[seasonalVariatioCounter];
        seasonalVariatioCounter++;

    }
    double[] valuesWOSeasonalVariation = new double[SalesIndex];
    for (int i = 0; i < SalesIndex; i++)
    {
        valuesWOSeasonalVariation[i] = SalesValues[i] / seasonalVariation[i];
    }
    double[] trendValues = new double[numberOfForecastingValues];
    var coeficients = TrendLineCalculation(SalesIndex, SalesIndexes, valuesWOSeasonalVariation);
    double coeficient1 = coeficients.Item1;
    double coeficient2 = coeficients.Item2;
    for (int i = 0; i < numberOfForecastingValues; i++)
    {
        trendValues[i] = (coeficient2 * forecastingValuesIndexes[i]) + coeficient1;
    }
    double[] trendValuesWithSeasonalVariation = new double[numberOfForecastingValues];
    for (int i = 0; i < numberOfForecastingValues; i++)
    {
        trendValuesWithSeasonalVariation[i] = trendValues[i] * seasonalVariation[i];
    }
    double[] errorsE = new double[SalesIndex];
    double[] errorsModule = new double[SalesIndex];
    double[] errorsSquare = new double[SalesIndex];
    for (int i = 0; i < SalesIndex; i++)
    {
        errorsE[i] = SalesValues[i] - trendValuesWithSeasonalVariation[i];
        errorsModule[i] = Math.Abs(errorsE[i]);
        errorsSquare[i] = Math.Pow(errorsE[i], 2);
    }
    double determinationCoefficient = DeterminationCoefficientCalculation(SalesIndex, SalesValues, trendValues, errorsSquare);
    double durbinWatsonStatistic = DurbinWatsonStatisticCalculation(SalesIndex, valuesWOSeasonalVariation, trendValues);
    double averageAbsoluteDeviation = AverageAbsoluteDeviationCalculation(SalesIndex, errorsModule);
    double averageRelativeError = AverageRelativeErrorCalculation(SalesIndex, SalesValues, errorsModule);
    return (numberOfForecastingValues, forecastingValuesIndexes, seasonalVariation, valuesWOSeasonalVariation,
   coeficient1, coeficient2, trendValues, trendValuesWithSeasonalVariation, errorsE,
     errorsModule, errorsSquare, determinationCoefficient, durbinWatsonStatistic, averageAbsoluteDeviation,
    averageRelativeError);

}
```

Рис. 3.15 Метод MultiplicativeModel

3.4. Інтерфейс користувача та функціонал інтерактивних елементів

Для класних віконних додатків невід'ємним атрибутом є гарно пропрацьований інтерфейс користувача.

Інтерфейс користувача (ІК) - це спосіб взаємодії користувача з комп'ютерною системою чи програмою через візуальні та звукові елементи. Основною метою інтерфейсу користувача є забезпечення зручного та ефективного використання програми чи системи.

Основні принципи інтерфейсу користувача включають:

1. **Простота та зрозумілість**: ІК повинен бути простим у використанні та зрозумілим для широкого кола користувачів.

2. **Консистентність**: Елементи інтерфейсу повинні мати однаковий вигляд і використовувати аналогічні команди для подібних завдань, щоб зменшити плутанину.

3. **Зручність**: ІК повинен бути зручним для використання, сприяючи продуктивності та комфорту користувача.

4. **Наявність зворотного зв'язку**: Система повинна надавати користувачеві зворотний зв'язок про результати його дій.

5. **Можливість адаптації**: ІК повинен бути гнучким і надавати можливості адаптації до потреб різних користувачів.

Інтерфейси користувача можуть бути графічними (GUI), текстовими (CLI), аудіо- або таціональними, залежно від типу програми чи пристрою. GUI, наприклад, включає графічні елементи, такі як вікна, кнопки та меню, для полегшення взаємодії користувача з програмою.

Слідуючи цим принципам, я розробив унікальний графічний інтерфейс користувача для додатку. Вмикаючи програму користувача зустрічає вітальне вікно, в якому зазначено те кваліфікаційної магістерської роботи та емблема університету, з інтерактивних елементів на тілі вітального розміщена кнопка “Почати роботу”, натискання якої перенесе користувача на головну форму. Система панель вікна була перероблена. Кольори використані в дизайні програми належать палітрі під назвою “ягідна синьова”.

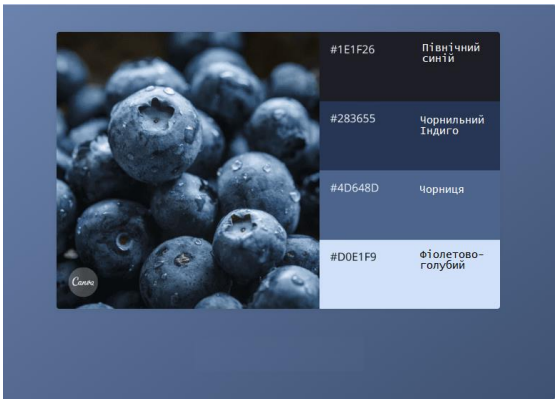


Рис. 3.16 палітра кольорів використаних в дизайні програми

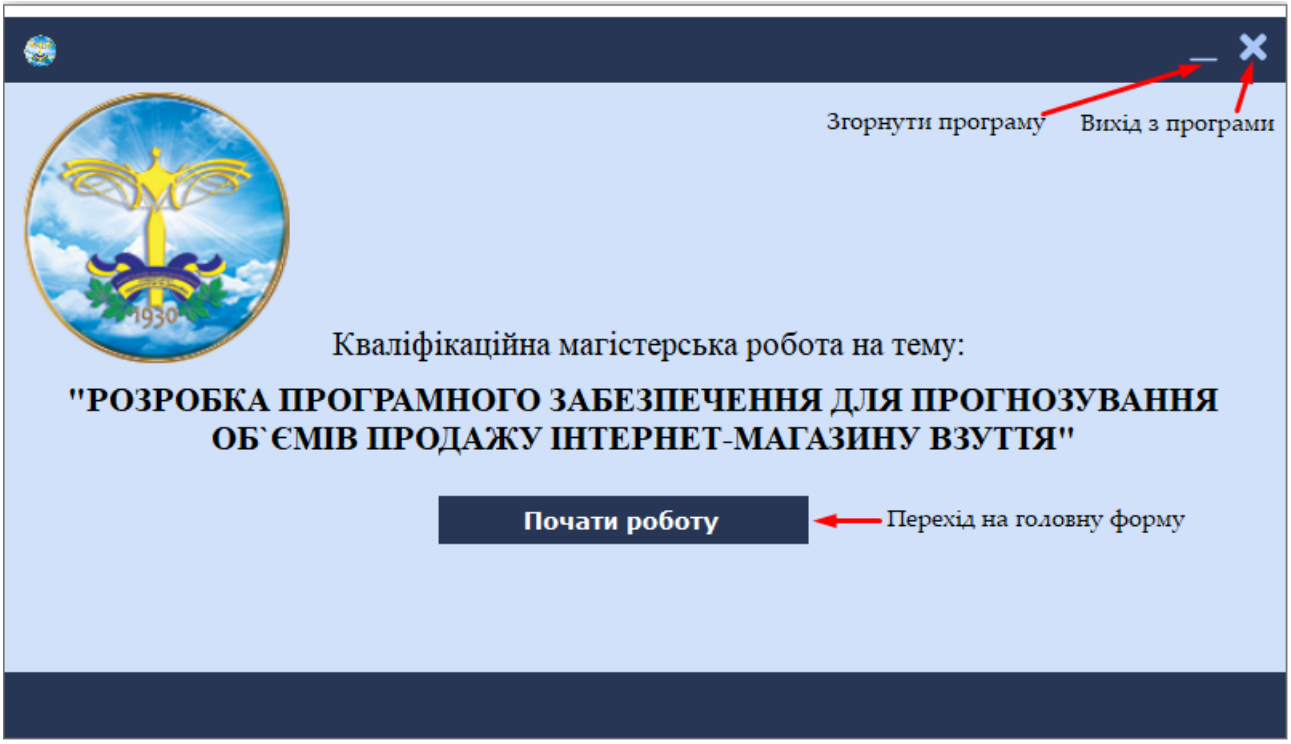


Рис. 3.17 Вітальний екран програми

Натиснувши на кнопку “Почати роботу” перед користувачем відкривається головна форма на якій розміщені всі інтерактивні елементи для виконання розрахунків. В першу чергу користувач повинен ввести експериментальні дані (рівні часового ряду) в текстовий бокс під назвою “Часовий ряд”, обрати одну з двох моделей, поставити кількість прогнозованих значень та натиснути на кнопку “Прогнозувати”.

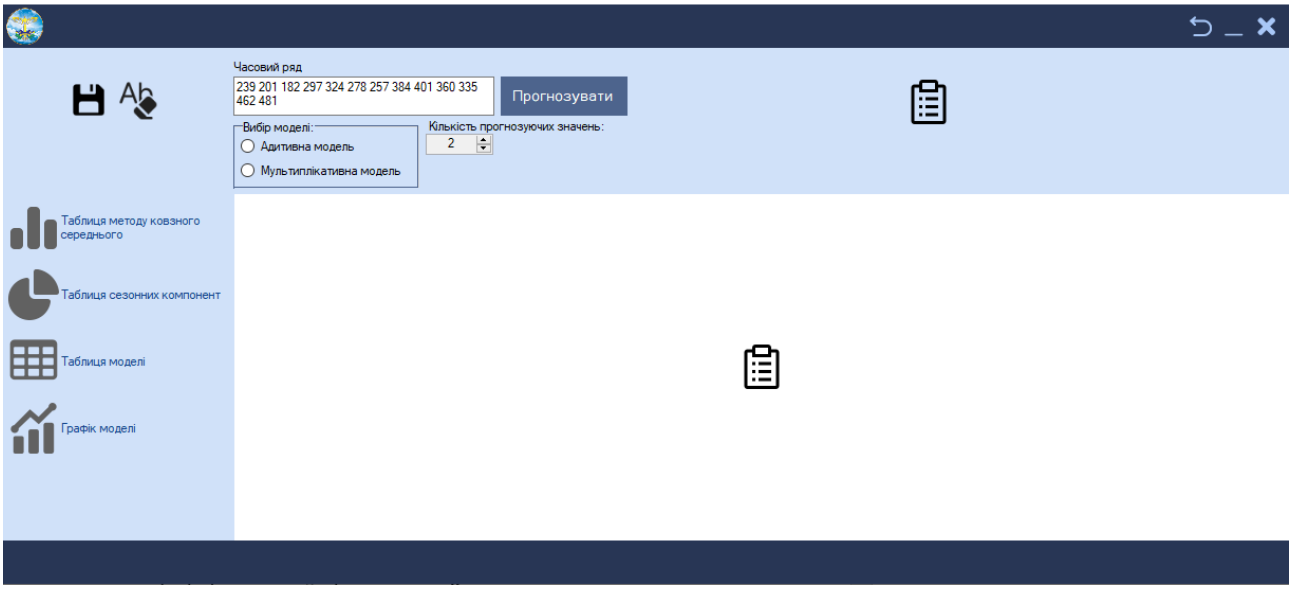


Рис. 3.18 Головна форма програми

В програмі наявний захист вводу інформації, якщо користувач буде вводити некоректну інформацію, або забуде обрати модель, програма про це

повідомить шляхом з’явлення tooltip повідомлення в місці помилки. Коректним вважається ряд чисел, який містить мінімум 8 значень написаних через пробіл, без використання любих інших символів.

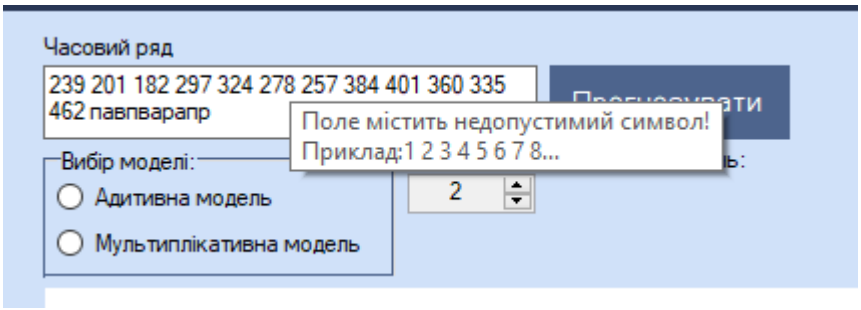


Рис. 3.19 Приклад некоректного вводу інформації

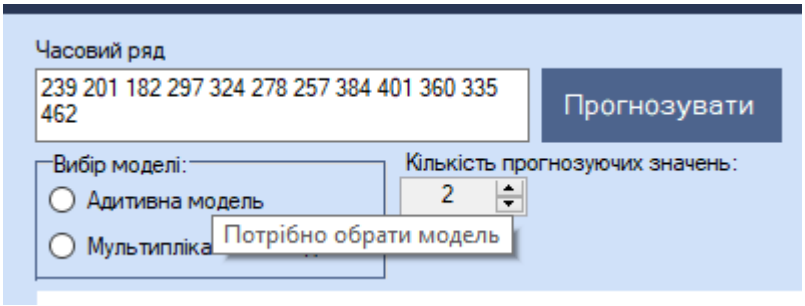


Рис. 3.20 Повідомлення при відсутності вибраної моделі

Якщо введені дані являються коректними, при натисканні кнопки “Прогнозувати” будуть здійснені всі необхідні розрахунки та на формі з’являться таблиці з результатами.

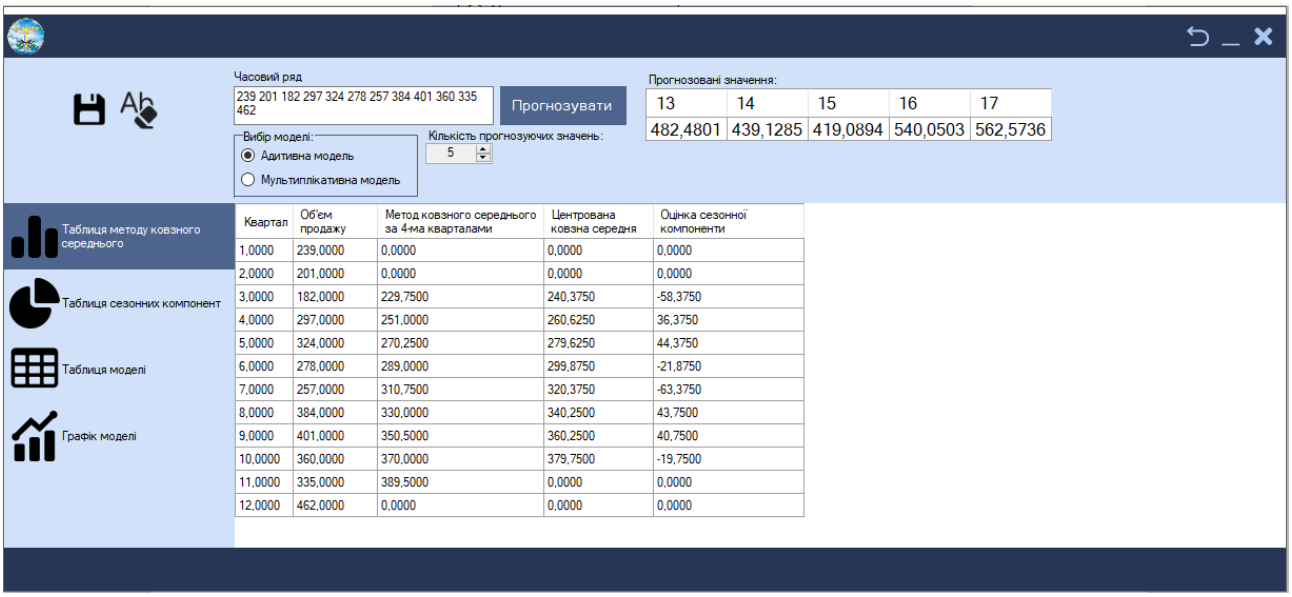


Рис. 3.21 Зміна форми після розрахунків

Щоб уникнути нагромадження інформації, в лівій частині програми було реалізовано можливість перемикатися між вкладками. Кожна вкладка містить в

собі інформацію зазначену в своїй назві. Слід зазначити, що можливість переходу між вкладками програми з’являється лише після здійснення розрахунків.

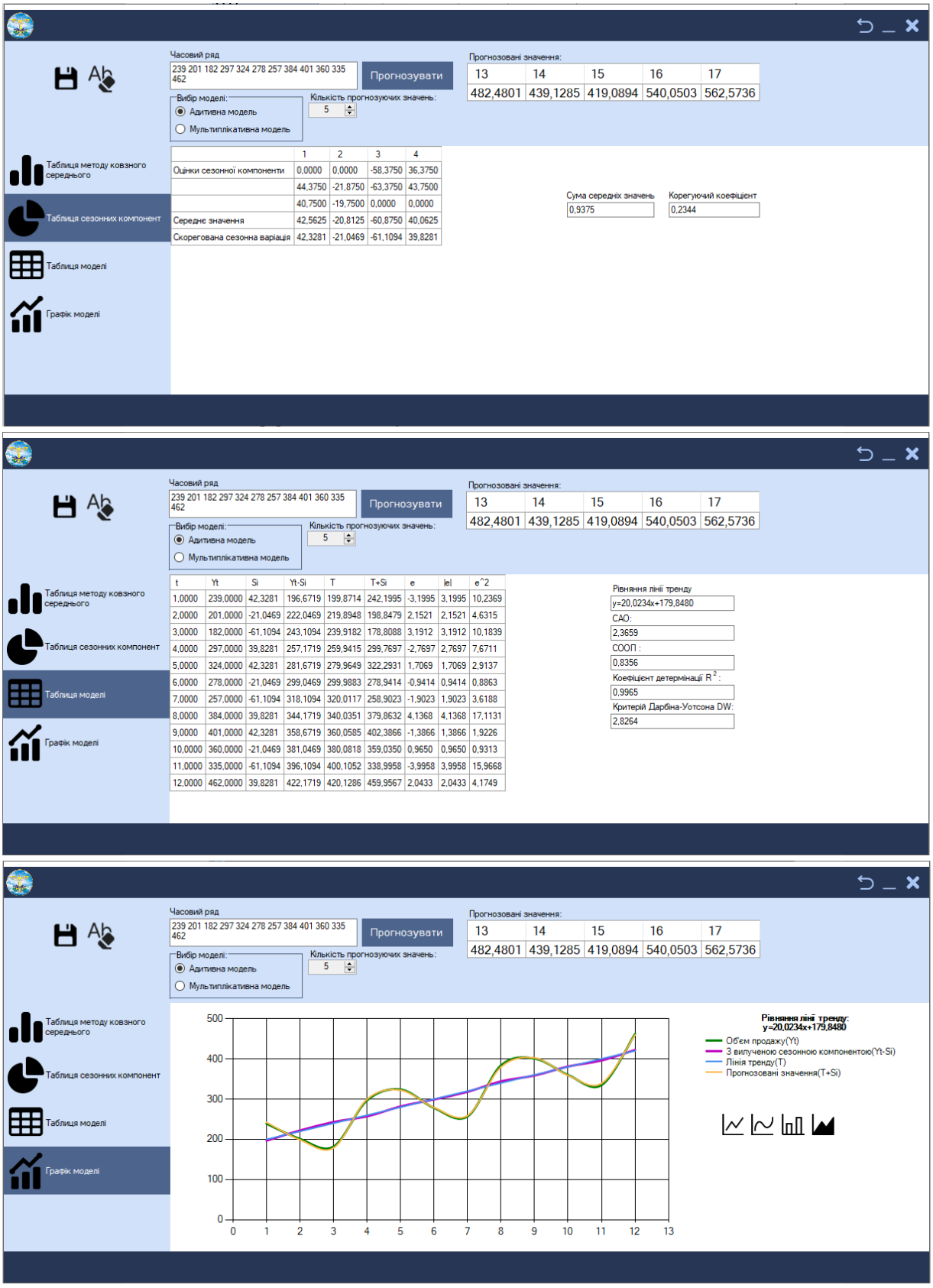


Рис. 3.22 Перехід між вкладками та їх контент

На вкладці “Графік моделі” у користувача, окрім того, що може дослідити дані графічно, є можливість міняти стиль графіку, на його смак.

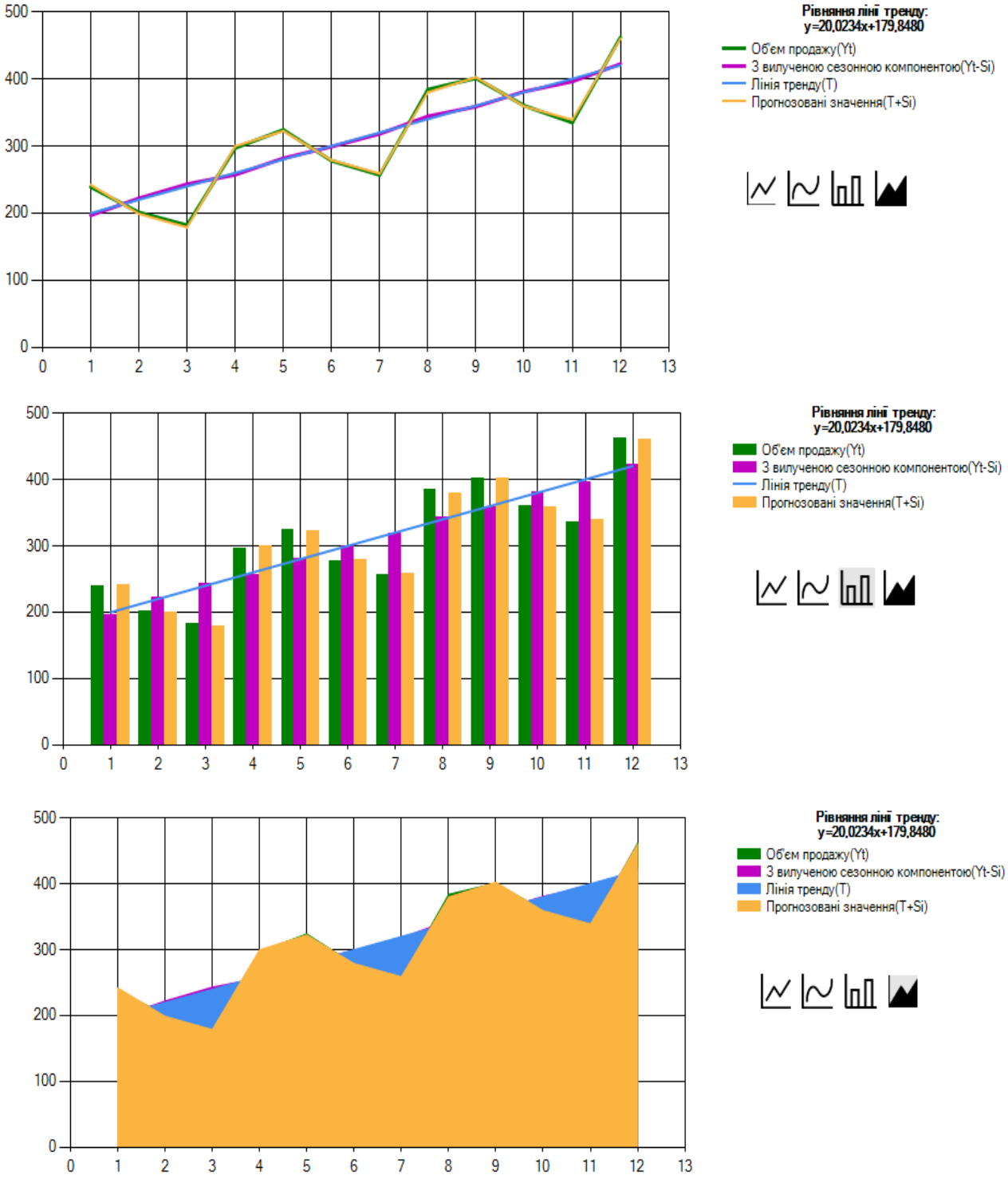


Рис. 3.23 Приклади зміни стилю графіку

Після того як користувач дослідив значення та отримав прогноз, в нього є можливість згенерувати excel звіт, що містить всі результати розрахунків та особистий графік. Також він може повністю очистити форму та продовжити роботу або закрити програму.

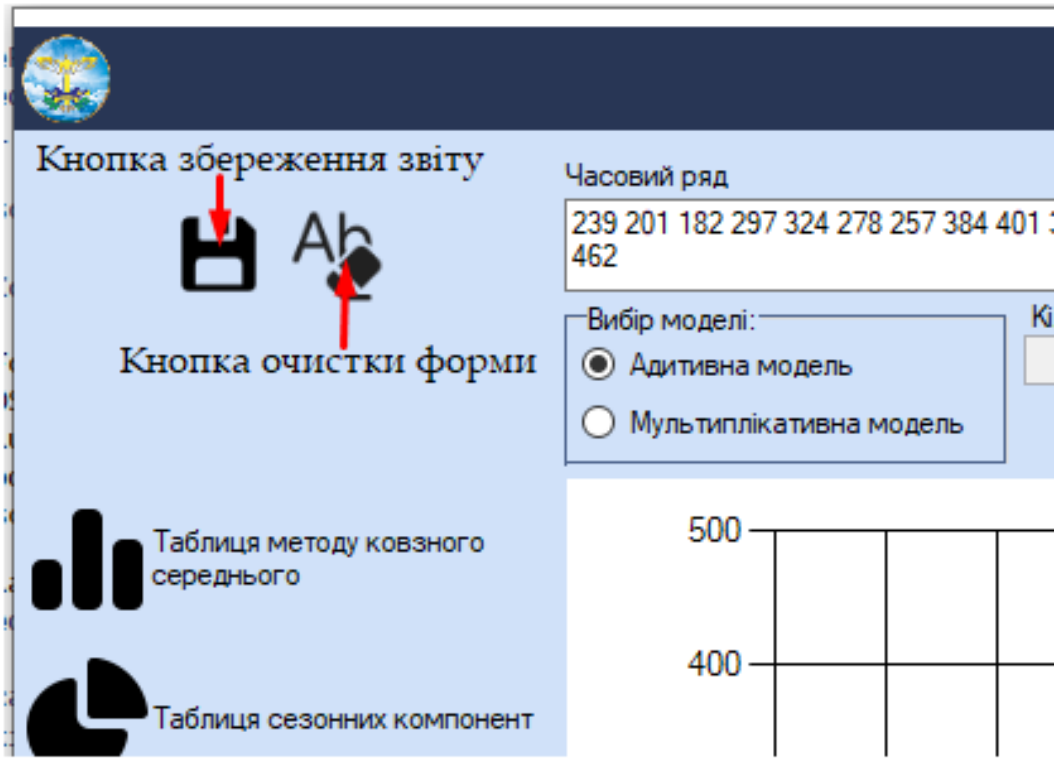


Рис. 3.24 Розташування кнопок

В випадку натискання на кнопку “Збереження звіту”, перед користувачем з’явиться діалогове вікно в якому він може назвати звіт та вказати шлях для його збереження.

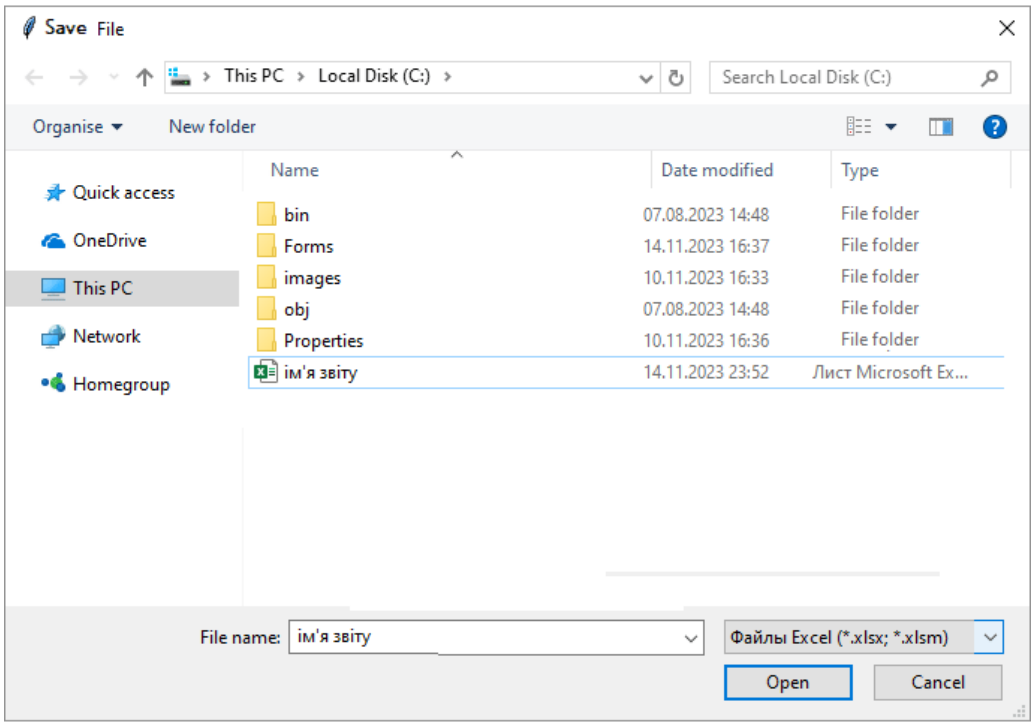
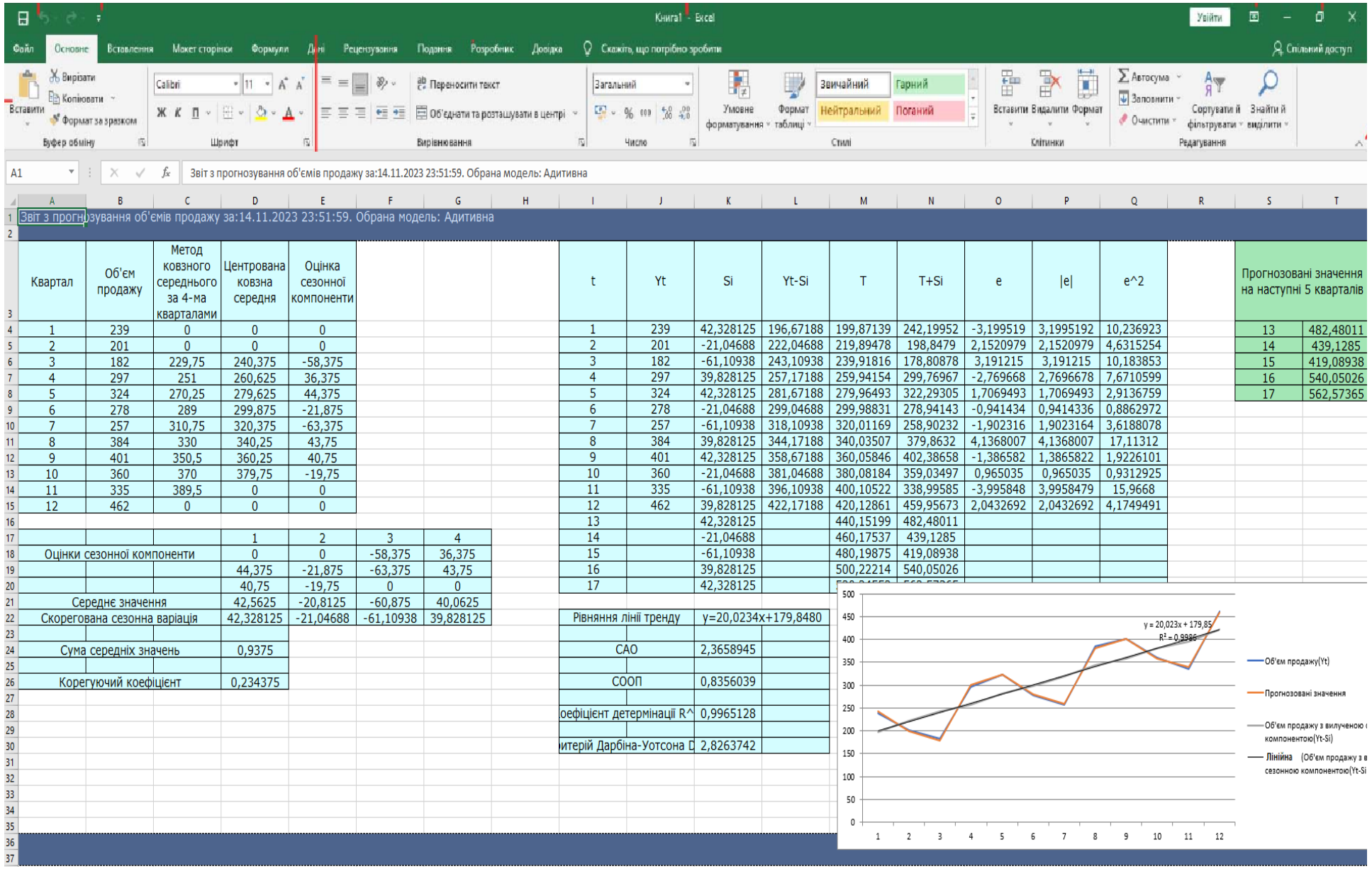


Рис. 3.25 Діалогове вікно для збереження звіту



Квартал	Об'єм продажу	Метод ковзного середнього за 4-ма кварталами	Центрована ковзна середня	Оцінка сезонної компоненти
1	239	0	0	0
2	201	0	0	0
3	182	229,75	240,375	-58,375
4	297	251	260,625	36,375
5	324	270,25	279,625	44,375
6	278	289	299,875	-21,875
7	257	310,75	320,375	-63,375
8	384	330	340,25	43,75
9	401	350,5	360,25	40,75
10	360	370	379,75	-19,75
11	335	389,5	0	0
12	462	0	0	0

	1	2	3	4
Оцінки сезонної компоненти	0	0	-58,375	36,375
	44,375	-21,875	-63,375	43,75
	40,75	-19,75	0	0
Середнє значення	42,5625	-20,8125	-60,875	40,0625
Скорегована сезонна варіація	42,328125	-21,04688	-61,10938	39,828125
Сума середніх значень	0,9375			
Корегуючий коефіцієнт	0,234375			

t	Yt	Si	Yt-Si	T	T+Si	e	|e|	e^2
1	239	42,328125	196,67188	199,87139	242,19952	-3,199519	3,1995192	10,236923
2	201	-21,04688	222,04688	219,89478	198,8479	2,1520979	2,1520979	4,6315254
3	182	-61,10938	243,10938	239,91816	178,80878	3,191215	3,191215	10,183853
4	297	39,828125	257,17188	259,94154	299,76967	-2,769668	2,7696678	7,6710599
5	324	42,328125	281,67188	279,96493	322,29305	1,7069493	1,7069493	2,9136759
6	278	-21,04688	299,04688	299,98831	278,94143	-0,941434	0,9414336	0,8862972
7	257	-61,10938	318,10938	320,01169	258,90232	-1,902316	1,9023164	3,6188078
8	384	39,828125	344,17188	340,03507	379,8632	4,1368007	4,1368007	17,11312
9	401	42,328125	358,67188	360,05846	402,38658	-1,386582	1,3865822	1,9226101
10	360	-21,04688	381,04688	380,08184	359,03497	0,965035	0,965035	0,9312925
11	335	-61,10938	396,10938	400,10522	338,99585	-3,995848	3,9958479	15,9668
12	462	39,828125	422,17188	420,12861	459,95673	2,0432692	2,0432692	4,1749491
13		42,328125		440,15199	482,48011			
14		-21,04688		460,17537	439,1285			
15		-61,10938		480,19875	419,08938			
16		39,828125		500,22214	540,05026			
17		42,328125						

Прогнозовані значення на наступні 5 кварталів	
13	482,48011
14	439,1285
15	419,08938
16	540,05026
17	562,57365

Рівняння лінії тренду	y=20,0234x+179,8480
САО	2,3658945
СООП	0,8356039
оефіцієнт детермінації R^	0,9965128
итерій Дарбіна-Уотсона D	2,8263742

Рис. 3.26 Приклад Excel звіту

Генерація Excel з кастомними даними, стилями та графіком здійснюються за допомогою бібліотеки взаємодій з об’єктами Office Microsoft Office.Interop.Excel та класу ExcelHelper.

Системна панель додатку була перероблена під стиль програми, всі кнопки на панелі створені та запрограмовані вручну.

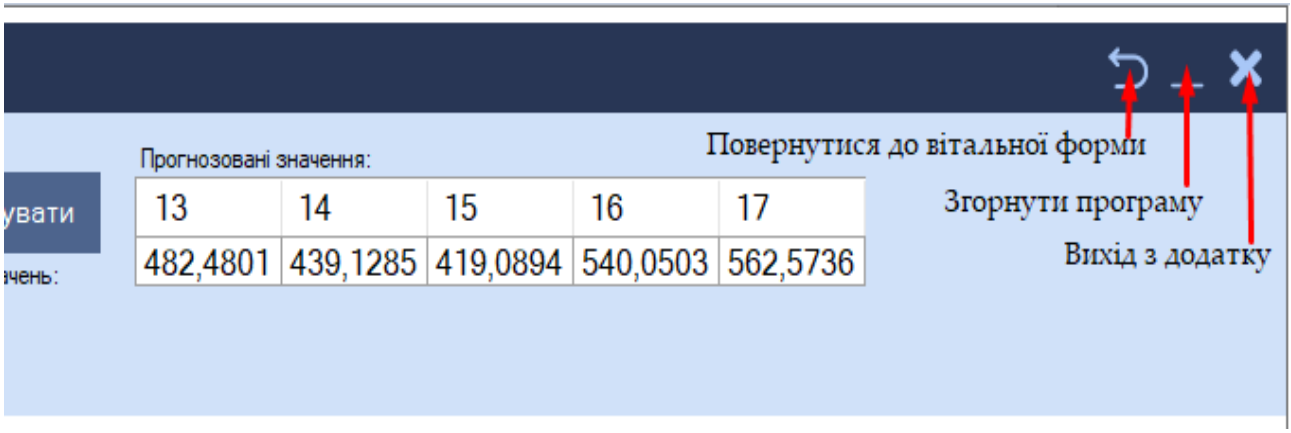


Рис. 3.27 Системна панель головної форми

Висновки до розділу 3

Для програмної реалізації алгоритму прогнозування об'ємів продажу інтернет-магазинів взуття було обрано середовище розробки від Microsoft – Visual Studio 2022 Community edition. Проєкт реалізований мові програмування C# та платформі .NET Framework. Ця платформа представляє розробнику колекцію класів та бібліотек корисних в великій кількості різних завдань. .NET Framework використовується для широкого спектру різноманітних програм, від корпоративних застосунків до маленьких сайтів.

В додатку реалізовано алгоритм побудови прогнозу в вигляді методів, що покроково розраховують етапи побудови моделей для прогнозування.

Додаток представляє собою віконний застосунок з сучасним графічним інтерфейсом користувача. Інтерфейс включає в себе наявність вікон, кнопок, форм та графічних елементів з особистим дизайном.

Також була реалізована функція, що дозволяє генерувати звіт-Excel таблицю, стилізовану під програму із застосуванням бібліотеки microsoft.office.interop.excel.

ВИСНОВКИ

В результаті виконаного дослідження було реалізовано та успішно виконано всі поставлені задачі. Для реалізації поставленої задачі дослідження було вивчено та проаналізовано теорію часових рядів.

Проаналізовано вплив часових рядів на економічні процеси, а саме процеси інтернет-торгівлі та торгівлі взуттям.

В ході дослідження розроблено алгоритм прогнозування об’ємів продажу інтернет-магазинів взуття в майбутній момент часу. Даний алгоритм сприяє оптимізації процесів фінансових та матеріальних витрат та позитивно впливає на прибуток.

Створено програмне забезпечення на мові програмування C#. Розроблений додаток включає в себе унікальний графічний інтерфейс користувача, що дозволяє потенційному користувачу зручно і приємно виконувати алгоритм прогнозування. Вся інформація представлена в вигляді таблиць, текстових полів та графіків з можливістю зміни стилю. Після завершення роботи з алгоритмом, у користувача є можливість зробити звіт, очистити форму та продовжити працювати, або просто завершити роботу.

Під час виконання даного проєкту були набуті нові знання та навички в сфері об’єктно орієнтованого програмування та розробки інтерфейсів користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Касьяненко В.О., Старченко Л.В. “ Моделювання та прогнозування економічних процесів.”, Київ, рік: 2023, 184 с.

2. Андерс Хейлсберг, “Мова програмування C#. 4-е вид.” 2012. 411 с.

3. Гончаренко Т. П. “Оцінювання сезонності в системі маркетингу промислових підприємств”, Суми, рік: 2014, 370 с.

4. Стів Макконнелл, “Ідельний код”, Україна 896 с.

5. Microsoft Visual Studio [Електронний ресурс] - https://learn.microsoft.com/ en-en /visualstudio/get-started/visual-studio-ide?view=vs-2022#discover-visual-studio.

6. “Introductory Time Series with R (Use R!)”, aul S.P. Cowpertwait and Andrew V. Metcalfe

7. Динаміка світових цін на нафту [Електронний ресурс] https://studfile.net/preview/9182666/page:4

8. Ціна алюмінію [Електронний ресурс] https://biz.liga.net/ua/ekonomika/all/novosti/tsena-alyuminiya-dostigla-14-letnego-maksimuma-iz-zaolimpiady-i-ugroz-rossii

9. Автокориляція в залишках[Електронний ресурс] https://epi.cc.ua/ avtokorelyatsiya-v-zalishkah-makroekonomichne-planuvannya-i-prognozuvannya.html

10. Віктор Юхимович Клепко, “Вища математика в прикладах і задачах”, Видавництво: Центр учбової літератури

11. Лінійна алгебра та аналітична геометрія [Електронний ресурс] https://matan.kpi.ua/public/files/Posibnyk%20LA+AG.pdf

12. Класи C#[Електронний ресурс] - https://learn.microsoft.com/en-en/dotnet/csharp/fundamentals/types/classes

13. “Time Series Analysis”, James Douglas Hamilton

14. Методи C# [Електронний ресурс] - https://learn.microsoft.com/ en-en /dotnet/csharp/programming-guide/classes-and-structs/methods

15. Діагностика побудованої моделі [Електронний ресурс] - https://studfile.net/preview/7878597/page:2/

16. Генератор палітри кольорів [Електронний ресурс] https:// www.canva.com/en_en/obuchenie/100-cvetovyx-sochetnij/

17. Microsoft .NET Framework [Електронний ресурс] - https://flexberry .github.io/en/gbt_dotnet.html

18. Практичний аналіз часових рядів. Прогнозування зі статистикою і машинне навчання. Ейлін Нільсен , 532 с.

19. 7 видів графічного дизайну для просування вашого бізнесу [Електронний ресурс] - https://blog.depositphotos.com/ua/vydy-grafichnogo-dyzajnu.html

20. Graphic, UI, Web Design – різниця [Електронний ресурс] - http://www.itschool.vn.ua/graphic-design-ui-design-web-design/

21. “Introductory Time Series with R (Use R!)”, aul S.P. Cowpertwait and Andrew V. Metcalfe

22. “The Analysis of Time Series: An Introduction”, Chris Chatfield

23. “Forecasting: Principles and Practice”, Rob J. Hyndman and George Athanasopoulos

24. Прогнозування та аналіз часових рядів [Електронний ресурс] - http://ir.stu.cn.ua/bitstream/handle/123456789/16992Прогнозув.%20та%20аналіз%20часових%20рядів.pdf?sequence=1&isAllowed

25. “Introduction to Time Series Analysis and Forecasting”, Douglas C. Montgomery, Cheryl L. Jennings, and Murat Kulahci

26. Algorithms + Data Structures = Programs, Nicolaus Wirth

27. Cracking the Coding Interview: 189 Programming Questions and Solutions 6th Edition, Г. Лакман Макдауелл, 687 с.

28. Refactoring: Improving the Design of Existing Code (2nd Edition), Martin Fowler, 448 с.

29. Head First Design Patterns: A Brain-Friendly Guide, Автори: Eric Freeman, Bert Bates, Kathy Sierra, Elisabett Robson, 692 с.

30. Introduction to Algorithms, 3rd Edition, Автори: Thomas H Corner, Charles E Leiserson, Ronald L Rivest, Clifford Stein, 1292 с.

31. Walkthrough: Office Programming in C# [Електронний ресурс] https://learn.microsoft.com/en-us/dotnet/csharp/advanced-topics/interop/walkthrough-office-programming

ДОДАТКИ

Додаток А

Повний код програми

```
using Diplom.Forms;
using Excel=Microsoft.Office.Interop.Excel;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.VisualStyles;


namespace Diplom
{
   public partial class MainScreen : Form
   {
      private Button currentButton;
      private Form activeForm;


      public MainScreen()
      {
         InitializeComponent();
         this.Text = string.Empty;
         this.ControlBox = false;
         this.MaximizedBounds =
Screen.FromHandle(this.Handle).WorkingArea;

      }
      private void ActivateButton(object btnSender)
      {
         if (btnSender != null)
         {
            if (currentButton != (Button)btnSender)
            {
               DisableButton();
```

```
                Color color = Color.FromArgb(77, 100, 141);
                currentButton = (Button)btnSender;
                currentButton.BackColor = color;
                currentButton.ForeColor = Color.White;
                currentButton.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

                ThemeColor.PrimaryColor = color;
                ThemeColor.SecondaryColor =
ThemeColor.ChangeColorBrightness(color, -0.3);

            }
        }
    }
    [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
    private extern static void ReleaseCapture();
    [DllImport("user32.DLL", EntryPoint = "SendMessage")]
    private extern static void SendMessage(System.IntPtr hWnd, int wMsg,
int wParam, int lParam);
    private void DisableButton()
    {
        foreach (Control previousBtn in panelMenu.Controls)
        {
            if (previousBtn.GetType() == typeof(Button))
            {
                previousBtn.BackColor = Color.FromArgb(208, 225, 249);
                previousBtn.ForeColor = Color.Black;
                previousBtn.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            }
        }
    }
    private void OpenChildForm(Form childForm, object btnSender)
    {
        if (activeForm != null)
        {
            activeForm.Close();
        }

        ActivateButton(btnSender);
        activeForm = childForm;
        childForm.TopLevel = false;
        childForm.FormBorderStyle = FormBorderStyle.None;
```

```
            childForm.Dock = DockStyle.Fill;
            this.panelDesktopPanel.Controls.Add(childForm);
            this.panelDesktopPanel.Tag = childForm;
            childForm.BringToFront();
            childForm.Show();

        }
        private void ClearForm()
        {
            dataGridView_ForecastingValues.Visible = false;
            btn_MovingAverage.Enabled = false;
            btn_SeasolanComponent.Enabled = false;
            btn_ModelData.Enabled = false;
            btn_ModelChart.Enabled = false;
            activeForm.Close();
            radioButton1.Checked = false;
            radioButton2.Checked = false;
            numericUpDown1.Value = 2;
            label3.Visible = false;
            btn_Clear.Enabled = false;
            DisableButton();
        }

        private bool ErrorsChecks()
        {

            if (this.textBox1.Text == string.Empty)
            {
                toolTip1.Show("Поле обов'язкове для заповнення!\nПриклад:1 2 3
4 5 6 7 8...", textBox1, 1000);
                return false;
            }
            char[] stringToChar = textBox1.Text.ToCharArray();
            bool invalidSymbolsCheck;
            for (int i = 0; i < stringToChar.Length; i++)
            {
                invalidSymbolsCheck = char.IsNumber(stringToChar[i]);
                if (invalidSymbolsCheck != true && stringToChar[i] != ' ')
                {
                    toolTip1.Show("Поле містить недопустимий
символ!\nПриклад:1 2 3 4 5 6 7 8...", textBox1, 1000);
                    return false;
                }
                if (stringToChar[i] == ' ' && stringToChar[i + 1] == ' ')
                {
```

```
                toolTip1.Show("Поле містить зайві відступи!\nПриклад:1 2 3 4
5 6 7 8...", textBox1, 1000);
                return false;
            }
            if (stringToChar[i] == ',')
            {
                if (char.IsNumber(stringToChar[i - 1]) != true ||
char.IsNumber(stringToChar[i + 1]) != true)
                {
                    toolTip1.Show("Некоректно розташована
кома!\nНедопустимо: 1, 2; 1 , 2...\nДопустимо: 1,2... ", textBox1, 1000);
                    return false;
                }
            }
        }
        if (char.IsNumber(stringToChar[0]) != true)
        {
            toolTip1.Show("Вхідні дані починаються з некоректного
символу!\nПриклад:1 2 3 4 5 6 7 8...", textBox1, 1000);
            return false;
        }
        var inputInformation = InputIformation(textBox1.Text);
        double[] dataForErrorsCheck = inputInformation.salesValues;
        if (dataForErrorsCheck.Length < 8)
        {
            toolTip1.Show("Кількість вхідних елементів повинна
дорівнювати мінімум 8!\nПриклад:1 2 3 4 5 6 7 8...", textBox1, 1000);
            return false;
        }

        return true;
    }
    static (int salesIndex, int[] salesIndexes, double[] salesValues)
InputIformation(string timeLineRow)
    {
        string[] subStringsOfSaleValues = timeLineRow.Split(' ');
        double[] salesValues = new double[subStringsOfSaleValues.Length];
        for (int i = 0; i < subStringsOfSaleValues.Length; i++)
        {
            salesValues[i] = Convert.ToDouble(subStringsOfSaleValues[i]);
        }
        int salesIndex = salesValues.Length;
        int[] salesIndexes = new int[salesIndex];
        for (int i = 0; i < salesIndex; i++)
        {
```

```
            salesIndexes[i] = i + 1;
        }
        return (salesIndex, salesIndexes, salesValues);

    }
    static (double[] movingAverageForFourQuarters, double[]
centralMovingAverageValues, double[] movingAverageEstimation)
MovingAverageForAdditiveModel(int SalesIndex, int[] SalesIndexes, double[]
SalesVal)
    {
        double[] SalesValues = new double[SalesIndex];
        SalesValues = SalesVal;
        double[] MovingAverageForFourQuarters = new double[SalesIndex];
        double SumOfQuarters;
        const int QUARTERS_NUMBER = 4;

        for (int i = 0; i <= SalesIndex - QUARTERS_NUMBER; i++)
        {
            SumOfQuarters = SalesValues[i] + SalesValues[i + 1] +
SalesValues[i + 2] + SalesValues[i + 3];
            MovingAverageForFourQuarters[i + 2] = SumOfQuarters /
QUARTERS_NUMBER;
        }

        double[] CentralMovingAverageValues = new double[SalesIndex];
        int NumberOfQuartersForCentralMovingAverage = 2;
        double SumOfCentralMovingAverageValues;
        for (int i = 2; i <= MovingAverageForFourQuarters.Length - 3; i++)
        {
            SumOfCentralMovingAverageValues =
(MovingAverageForFourQuarters[i] +
            MovingAverageForFourQuarters[i + 1]);
            CentralMovingAverageValues[i] =
SumOfCentralMovingAverageValues
                / NumberOfQuartersForCentralMovingAverage;
        }
        double[] MovingAverageEstimation = new double[SalesIndex];
        for (int i = 2; i <= MovingAverageForFourQuarters.Length - 3; i++)
        {
            MovingAverageEstimation[i] = SalesValues[i] -
CentralMovingAverageValues[i];
        }
        return (MovingAverageForFourQuarters,
CentralMovingAverageValues, MovingAverageEstimation);
```

```
        }
        static (double[,] seasonalComponent, double[]
averageValueOfSeasonalComponents, double
sumOfaverageValueOfSeasonalComponents, double correctiveFactor, double[]
correctedSeasonalVariation)
            SeasonalComponentCalculationForAdditiveModel(double[] MAE)
        {
            double[] MovingAverageEstimation = MAE;
            const int QUARTERS_NUMBER = 4;
            int quartersRepeat;
            int test = MovingAverageEstimation.Length %
QUARTERS_NUMBER;
            if (MovingAverageEstimation.Length % QUARTERS_NUMBER == 0
|| MovingAverageEstimation.Length % QUARTERS_NUMBER == 1)
            {
                quartersRepeat = MovingAverageEstimation.Length /
QUARTERS_NUMBER;
            }
            else quartersRepeat = MovingAverageEstimation.Length /
QUARTERS_NUMBER + 1;

            double[,] seasonalComponent = new double[quartersRepeat,
QUARTERS_NUMBER];
            int counter = 0;
            for (int i = 0; i < quartersRepeat; i++)
            {
                for (int j = 0; j < QUARTERS_NUMBER; j++)
                {
                    if (counter < MovingAverageEstimation.Length)
                    {
                        seasonalComponent[i, j] = MovingAverageEstimation[counter];
                    }
                    else seasonalComponent[i, j] = 0;
                    counter++;
                }
            }
            double[] sumOfSeasonalComponents = new
double[QUARTERS_NUMBER];
            double[] averageValueOfSeasonalComponents = new
double[QUARTERS_NUMBER];
            double sumOfaverageValueOfSeasonalComponents = 0;

            int averageCounter = 0;
            for (int i = 0; i < QUARTERS_NUMBER; i++)
            {
```

```
            for (int j = 0; j < quartersRepeat; j++)
            {
               if (seasonalComponent[j, i] != 0)
               {
                  sumOfSeasonalComponents[i] += seasonalComponent[j, i];
                  averageCounter++;
               }
            }
            averageValueOfSeasonalComponents[i] =
sumOfSeasonalComponents[i] / averageCounter;
            averageCounter = 0;
            sumOfaverageValueOfSeasonalComponents +=
averageValueOfSeasonalComponents[i];

         }
         double correctiveFactor = sumOfaverageValueOfSeasonalComponents
/ QUARTERS_NUMBER;
         double[] correctedSeasonalVariation = new
double[QUARTERS_NUMBER];
         for (int i = 0; i < QUARTERS_NUMBER; i++)
         {
            correctedSeasonalVariation[i] =
averageValueOfSeasonalComponents[i] - correctiveFactor;
         }
         return (seasonalComponent, averageValueOfSeasonalComponents,
sumOfaverageValueOfSeasonalComponents, correctiveFactor,
correctedSeasonalVariation);
      }
      static (int numberOfForecastingValues, double[]
forecastingValuesIndexes, double[] seasonalVariation,
         double[] valuesWOSeasonalVariation, double coeficient1, double
coeficient2, double[] trendValues,
         double[] trendValuesWithSeasonalVariation, double[] errorsE,
         double[] errorsModule, double[] errorsSquare, double
determinationCoefficient,
         double durbinWatsonStatistic, double averageAbsoluteDeviation,
         double averageRelativeError) AdditiveModel(int SalesIndex, int[]
SalesIndexes, double[] SalesValues,
         double[] correctedSeasonalVariation, int forecastingNumber)
      {
         int numberOfForecastingValues = SalesIndex + forecastingNumber;
         double[] forecastingValuesIndexes = new
double[numberOfForecastingValues];
         for (int i = 0; i < numberOfForecastingValues; i++)
         {
```

```
            forecastingValuesIndexes[i] = i + 1;
        }
        double[] seasonalVariation = new double[numberOfForecastingValues];
        int seasonalVariatioCounter = 0;
        for (int i = 0; i < numberOfForecastingValues; i++)
        {
            if ((seasonalVariatioCounter) % correctedSeasonalVariation.Length
== 0)
            {
                seasonalVariatioCounter = 0;
            }
            seasonalVariation[i] =
correctedSeasonalVariation[seasonalVariatioCounter];
            seasonalVariatioCounter++;

        }
        double[] valuesWOSeasonalVariation = new double[SalesIndex];
        for (int i = 0; i < SalesIndex; i++)
        {
            valuesWOSeasonalVariation[i] = SalesValues[i] -
seasonalVariation[i];
        }
        double[] trendValues = new double[numberOfForecastingValues];
        var coeficients = TrendLineCalculation(SalesIndex, SalesIndexes,
valuesWOSeasonalVariation);
        double coeficient1 = coeficients.Item1;
        double coeficient2 = coeficients.Item2;
        for (int i = 0; i < numberOfForecastingValues; i++)
        {
            trendValues[i] = (coeficient2 * forecastingValuesIndexes[i]) +
coeficient1;
        }
        double[] trendValuesWithSeasonalVariation = new
double[numberOfForecastingValues];
        for (int i = 0; i < numberOfForecastingValues; i++)
        {
            trendValuesWithSeasonalVariation[i] = trendValues[i] +
seasonalVariation[i];
        }
        double[] errorsE = new double[SalesIndex];
        double[] errorsModule = new double[SalesIndex];
        double[] errorsSquare = new double[SalesIndex];
        for (int i = 0; i < SalesIndex; i++)
        {
            errorsE[i] = SalesValues[i] - trendValuesWithSeasonalVariation[i];
```

```
            errorsModule[i] = Math.Abs(errorsE[i]);
            errorsSquare[i] = Math.Pow(errorsE[i], 2);
        }
        double determinationCoefficient =
DeterminationCoefficientCalculation(SalesIndex, SalesValues, trendValues,
errorsSquare);
        double durbinWatsonStatistic =
DurbinWatsonStatisticCalculation(SalesIndex, valuesWOSeasonalVariation,
trendValues);
        double averageAbsoluteDeviation =
AverageAbsoluteDeviationCalculation(SalesIndex, errorsModule);
        double averageRelativeError =
AverageRelativeErrorCalculation(SalesIndex, SalesValues, errorsModule);
        return (numberOfForecastingValues, forecastingValuesIndexes,
seasonalVariation, valuesWOSeasonalVariation,
       coeficient1, coeficient2, trendValues,
trendValuesWithSeasonalVariation, errorsE,
         errorsModule, errorsSquare, determinationCoefficient,
durbinWatsonStatistic, averageAbsoluteDeviation,
        averageRelativeError);

    }
    static (double[] movingAverageForFourQuarters, double[]
centralMovingAverageValues, double[] movingAverageEstimation)
MovingAverageForMultiplicativeModel(int SalesIndex, int[] SalesIndexes,
double[] SalesVal)
    {
        double[] SalesValues = new double[SalesIndex];
        SalesValues = SalesVal;
        double[] MovingAverageForFourQuarters = new double[SalesIndex];
        double SumOfQuarters;
        const int QUARTERS_NUMBER = 4;

        for (int i = 0; i <= SalesIndex - QUARTERS_NUMBER; i++)
        {
            SumOfQuarters = SalesValues[i] + SalesValues[i + 1] +
SalesValues[i + 2] + SalesValues[i + 3];
            MovingAverageForFourQuarters[i + 2] = SumOfQuarters /
QUARTERS_NUMBER;
        }

        double[] CentralMovingAverageValues = new double[SalesIndex];
        int NumberOfQuartersForCentralMovingAverage = 2;
        double SumOfCentralMovingAverageValues;
        for (int i = 2; i <= MovingAverageForFourQuarters.Length - 3; i++)
```

```
        {
            SumOfCentralMovingAverageValues =
(MovingAverageForFourQuarters[i] +
            MovingAverageForFourQuarters[i + 1]);
            CentralMovingAverageValues[i] =
SumOfCentralMovingAverageValues
                / NumberOfQuartersForCentralMovingAverage;
        }
        double[] MovingAverageEstimation = new double[SalesIndex];
        for (int i = 2; i <= MovingAverageForFourQuarters.Length - 3; i++)
        {
            MovingAverageEstimation[i] = SalesValues[i] /
CentralMovingAverageValues[i];
        }
        return (MovingAverageForFourQuarters,
CentralMovingAverageValues, MovingAverageEstimation);

    }
    static (double[,] seasonalComponent, double[]
averageValueOfSeasonalComponents,
        double sumOfaverageValueOfSeasonalComponents,
        double correctiveFactor, double[] correctedSeasonalVariation)
        SeasonalComponentCalculationForMultiplicativeModel(double[]
MAE)
    {
        double[] MovingAverageEstimation = MAE;
        const int QUARTERS_NUMBER = 4;
        int quartersRepeat = MovingAverageEstimation.Length /
QUARTERS_NUMBER;
        double[,] seasonalComponent = new double[quartersRepeat,
QUARTERS_NUMBER];
        int counter = 0;
        for (int i = 0; i < quartersRepeat; i++)
        {
            for (int j = 0; j < QUARTERS_NUMBER; j++)
            {

                seasonalComponent[i, j] = MovingAverageEstimation[counter];
                counter++;
            }
        }
        double[] sumOfSeasonalComponents = new
double[QUARTERS_NUMBER];
        double[] averageValueOfSeasonalComponents = new
double[QUARTERS_NUMBER];
```

```
        double sumOfaverageValueOfSeasonalComponents = 0;

        int averageCounter = 0;
        for (int i = 0; i < QUARTERS_NUMBER; i++)
        {
           for (int j = 0; j < quartersRepeat; j++)
           {
              if (seasonalComponent[j, i] != 0)
              {
                 sumOfSeasonalComponents[i] += seasonalComponent[j, i];
                 averageCounter++;
              }
           }
           averageValueOfSeasonalComponents[i] =
sumOfSeasonalComponents[i] / averageCounter;
           averageCounter = 0;
           sumOfaverageValueOfSeasonalComponents +=
averageValueOfSeasonalComponents[i];

        }
        double correctiveFactor = QUARTERS_NUMBER /
sumOfaverageValueOfSeasonalComponents;
        double[] correctedSeasonalVariation = new
double[QUARTERS_NUMBER];
        for (int i = 0; i < QUARTERS_NUMBER; i++)
        {
           correctedSeasonalVariation[i] =
averageValueOfSeasonalComponents[i] * correctiveFactor;
        }
        return (seasonalComponent, averageValueOfSeasonalComponents,
sumOfaverageValueOfSeasonalComponents, correctiveFactor,
correctedSeasonalVariation);
     }
     static (int numberOfForecastingValues, double[]
forecastingValuesIndexes, double[] seasonalVariation,
        double[] valuesWOSeasonalVariation,double coeficient1, double
coeficient2, double[] trendValues,
        double[] trendValuesWithSeasonalVariation, double[] errorsE,double[]
errorsModule, double[] errorsSquare,
        double determinationCoefficient, double durbinWatsonStatistic, double
averageAbsoluteDeviation,
        double averageRelativeError) MultiplicativeModel(int SalesIndex, int[]
SalesIndexes,
        double[] SalesValues, double[] correctedSeasonalVariation, int
forecastingNumber)
```

```
        {
            int numberOfForecastingValues = SalesIndex + forecastingNumber;
            double[] forecastingValuesIndexes = new
double[numberOfForecastingValues];
            for (int i = 0; i < numberOfForecastingValues; i++)
            {
                forecastingValuesIndexes[i] = i + 1;
            }
            double[] seasonalVariation = new double[numberOfForecastingValues];
            int seasonalVariatioCounter = 0;
            for (int i = 0; i < numberOfForecastingValues; i++)
            {
                if ((seasonalVariatioCounter) % correctedSeasonalVariation.Length
== 0)
                {
                    seasonalVariatioCounter = 0;
                }
                seasonalVariation[i] =
correctedSeasonalVariation[seasonalVariatioCounter];
                seasonalVariatioCounter++;

            }
            double[] valuesWOSeasonalVariation = new double[SalesIndex];
            for (int i = 0; i < SalesIndex; i++)
            {
                valuesWOSeasonalVariation[i] = SalesValues[i] /
seasonalVariation[i];
            }
            double[] trendValues = new double[numberOfForecastingValues];
            var coeficients = TrendLineCalculation(SalesIndex, SalesIndexes,
valuesWOSeasonalVariation);
            double coeficient1 = coeficients.Item1;
            double coeficient2 = coeficients.Item2;
            for (int i = 0; i < numberOfForecastingValues; i++)
            {
                trendValues[i] = (coeficient2 * forecastingValuesIndexes[i]) +
coeficient1;
            }
            double[] trendValuesWithSeasonalVariation = new
double[numberOfForecastingValues];
            for (int i = 0; i < numberOfForecastingValues; i++)
            {
                trendValuesWithSeasonalVariation[i] = trendValues[i] *
seasonalVariation[i];
            }
```

```
            double[] errorsE = new double[SalesIndex];
            double[] errorsModule = new double[SalesIndex];
            double[] errorsSquare = new double[SalesIndex];
            for (int i = 0; i < SalesIndex; i++)
            {
                errorsE[i] = SalesValues[i] - trendValuesWithSeasonalVariation[i];
                errorsModule[i] = Math.Abs(errorsE[i]);
                errorsSquare[i] = Math.Pow(errorsE[i], 2);
            }
            double determinationCoefficient =
DeterminationCoefficientCalculation(SalesIndex, SalesValues, trendValues,
errorsSquare);
            double durbinWatsonStatistic =
DurbinWatsonStatisticCalculation(SalesIndex, valuesWOSeasonalVariation,
trendValues);
            double averageAbsoluteDeviation =
AverageAbsoluteDeviationCalculation(SalesIndex, errorsModule);
            double averageRelativeError =
AverageRelativeErrorCalculation(SalesIndex, SalesValues, errorsModule);
            return (numberOfForecastingValues, forecastingValuesIndexes,
seasonalVariation, valuesWOSeasonalVariation,
           coeficient1, coeficient2, trendValues,
trendValuesWithSeasonalVariation, errorsE,
             errorsModule, errorsSquare, determinationCoefficient,
durbinWatsonStatistic, averageAbsoluteDeviation,
            averageRelativeError);

        }
        static (double, double) TrendLineCalculation(int salesIndex, int[]
SalesIndexes, double[] valuesWOSeasonalVariation)
        {
            int sumOfIndexes = 0;
            for (int i = 0; i < salesIndex; i++)
            {
                sumOfIndexes += SalesIndexes[i];
            }
            double sumOfValues = 0;
            for (int i = 0; i < salesIndex; i++)
            {
                sumOfValues += valuesWOSeasonalVariation[i];
            }
            int[] squareOfIndexes = new int[salesIndex];
            int sumOfSquareOfIndexes = 0;
            for (int i = 0; i < salesIndex; i++)
            {
```

```
            squareOfIndexes[i] = (int)Math.Pow(SalesIndexes[i], 2);
            sumOfSquareOfIndexes += squareOfIndexes[i];
        }
        double[] productOfIndexesAndValues = new double[salesIndex];
        double sumOfProducts = 0;
        for (int i = 0; i < salesIndex; i++)
        {
            productOfIndexesAndValues[i] = SalesIndexes[i] *
valuesWOSeasonalVariation[i];
            sumOfProducts += productOfIndexesAndValues[i];
        }
        double[,] linearEquationMatrix = { { salesIndex, sumOfIndexes,
sumOfValues }, { sumOfIndexes, sumOfSquareOfIndexes, sumOfProducts } };
        double matrixDeterminant1 = linearEquationMatrix[0, 0] *
linearEquationMatrix[1, 1] - linearEquationMatrix[1, 0] *
linearEquationMatrix[0, 1];
        double matrixDeterminant2 = linearEquationMatrix[0, 2] *
linearEquationMatrix[1, 1] - linearEquationMatrix[1, 2] *
linearEquationMatrix[0, 1];
        double matrixDeterminant3 = linearEquationMatrix[0, 0] *
linearEquationMatrix[1, 2] - linearEquationMatrix[1, 0] *
linearEquationMatrix[0, 2];
        double coefficient1 = matrixDeterminant2 / matrixDeterminant1;
        double coefficient2 = matrixDeterminant3 / matrixDeterminant1;
        return (coefficient1, coefficient2);

    }
    static double DeterminationCoefficientCalculation(int salesIndex,
double[] salesValues, double[] trendValues, double[] errorsSquare)
    {
        double[] salesValuesMinusTrendValues = new double[salesIndex];
        double[] squareOfSalesValuesMinusTrendValues = new
double[salesIndex];
        double sumOfSquares = 0;
        double sumOfErrorSquares = 0;
        for (int i = 0; i < salesIndex; i++)
        {
            salesValuesMinusTrendValues[i] = salesValues[i] - trendValues[i];
            squareOfSalesValuesMinusTrendValues[i] =
Math.Pow(salesValuesMinusTrendValues[i], 2);
            sumOfSquares += squareOfSalesValuesMinusTrendValues[i];
            sumOfErrorSquares += errorsSquare[i];
        }
        double determinantCoefficient = 1 - (sumOfErrorSquares /
sumOfSquares);
```

```
            return determinantCoefficient;
        }
        static double DurbinWatsonStatisticCalculation(int salesIndex, double[]
valuesWOSeasonalVariation, double[] trendValues)
        {
            double[] salesValuesMinusTrendValues = new double[salesIndex];
            double[] squareOfSalesValuesMinusTrendValues = new
double[salesIndex];
            double sumOfSquares = 0;
            for (int i = 0; i < salesIndex; i++)
            {
                salesValuesMinusTrendValues[i] = valuesWOSeasonalVariation[i] -
trendValues[i];
                squareOfSalesValuesMinusTrendValues[i] =
Math.Pow(salesValuesMinusTrendValues[i], 2);
                sumOfSquares += squareOfSalesValuesMinusTrendValues[i];
            }
            double[] DWBufferValues = new double[salesIndex - 1];
            double[] squareOfDWBufferValues = new double[salesIndex - 1];
            double sumOfSquaresDWBufferValues = 0;
            for (int i = 1; i < salesIndex; i++)
            {
                DWBufferValues[i - 1] = salesValuesMinusTrendValues[i - 1] -
salesValuesMinusTrendValues[i];
                squareOfDWBufferValues[i - 1] = Math.Pow(DWBufferValues[i - 1],
2);
                sumOfSquaresDWBufferValues += squareOfDWBufferValues[i - 1];
            }
            double durbinWatsonStatistic = sumOfSquaresDWBufferValues /
sumOfSquares;
            return durbinWatsonStatistic;
        }
        static double AverageAbsoluteDeviationCalculation(int salesIndex,
double[] errorsModule)
        {
            double averageAbsoluteDeviationtmp = 0;
            for (int i = 0; i < salesIndex; i++)
            {
                averageAbsoluteDeviationtmp += errorsModule[i];
            }
            double averageAbsoluteDeviation = averageAbsoluteDeviationtmp /
salesIndex;
            return averageAbsoluteDeviation;
        }
```

```
        static double AverageRelativeErrorCalculation(int salesIndex, double[]
salesValues, double[] errorsModule)
        {
            double averageRelativeErrortmp = 0;
            for (int i = 0; i < salesIndex; i++)
            {
                averageRelativeErrortmp += (errorsModule[i] / salesValues[i]) *
100;
            }
            double averageRelativeError = averageRelativeErrortmp / salesIndex;
            return averageRelativeError;
        }
        public void CreateForecastingValuesTable(int salesIndex, int
numberOfForecastingValues)
        {
            if (dataGridView_ForecastingValues.ColumnCount <
numberOfForecastingValues)
            {
                for (int i = salesIndex; i < salesIndex+numberOfForecastingValues;
i++)
                {
                    dataGridView_ForecastingValues.Columns.Add($"col{i + 1}",
$"{i + 1}");
                }


                dataGridView_ForecastingValues.RowHeadersVisible = false;
                for (int i = 0; i <numberOfForecastingValues-salesIndex; i++)
                {
                    dataGridView_ForecastingValues.Columns[i].SortMode =
DataGridViewColumnSortMode.NotSortable;
                }
            }
            else return;



        }
        public void ForecastingValuesTableSetValue(int col, int row, double
value)
        {

            dataGridView_ForecastingValues.Rows[row].Cells[col].Value =
$"{value:f4}";
        }
```

```
        private void ForecastingValuesTableFill(int salesIndex, int
numberOfForecastingValues, double[] trendValuesWithSeasonalVariation)
        {
            dataGridView_ForecastingValues.Columns.Clear();

CreateForecastingValuesTable(TimeLines.SalesIndex,Convert.ToInt32(numeri
cUpDown1.Value));
            dataGridView_ForecastingValues.RowCount = 1;

            for (int i = 0; i < numberOfForecastingValues-salesIndex; i++)
            {

                ForecastingValuesTableSetValue(i, 0,
trendValuesWithSeasonalVariation[salesIndex + i]);
                dataGridView_ForecastingValues.Columns[i].SortMode =
DataGridViewColumnSortMode.NotSortable;

            }
            dataGridView_ForecastingValues.ClearSelection();
        }


        private void BackToWelcomeScreen_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void MainScreen_Load(object sender, EventArgs e)
        {
            textBox1.Text = "239 201 182 297 324 278 257 384 401 360 335 462
481";
            dataGridView_ForecastingValues.Visible = false;
            btn_MovingAverage.Enabled = false;
            btn_SeasolanComponent.Enabled = false;
            btn_ModelData.Enabled = false;
            btn_ModelChart.Enabled = false;
            label3.Visible = false;
            btn_Clear.Enabled = false;
        }
```

```
        private void btn_MovingAverage_Click(object sender, EventArgs e)
        {
            OpenChildForm(new Forms.MovingAverage(), sender) ;
        }

        private void btn_SeasolanComponent_Click(object sender, EventArgs e)
        {
            OpenChildForm(new Forms.SeasonalComponent(), sender);
        }

        private void btn_ModelChart_Click(object sender, EventArgs e)
        {
            OpenChildForm(new Forms.ModelChart(), sender);
        }

        private void btn_ModelData_Click(object sender, EventArgs e)
        {
            OpenChildForm(new Forms.ModelData(), sender);
        }

        private void panel1_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }

        private void btn_Calculate_Click(object sender, EventArgs e)
        {
            bool errorsCheck = ErrorsChecks();

            if (errorsCheck == false)
            {
                return;
            }
            if (errorsCheck == true)
            {
                var input = InputIformation(this.textBox1.Text);
                var movingAverage =
MovingAverageForAdditiveModel(input.salesIndex, input.salesIndexes,
input.salesValues);
```

```
            var seasonalComponent =
SeasonalComponentCalculationForAdditiveModel(movingAverage.movingAv
erageEstimation);
            var additiveModel = AdditiveModel(input.salesIndex,
input.salesIndexes, input.salesValues,
seasonalComponent.correctedSeasonalVariation,
Convert.ToInt32(numericUpDown1.Value));

            if (radioButton1.Checked == false && radioButton2.Checked ==
false)
            {
               toolTip1.Show("Потрібно обрати модель", groupBox1, 1000);
               return;
            }

            if (radioButton1.Checked == true)
            {
               TimeLines.isModelChosen = true;
               TimeLines.SalesIndex = input.salesIndex;
               TimeLines.SalesIndexes = input.salesIndexes;
               TimeLines.SalesValues = input.salesValues;
               TimeLines.MovingAverageForFourQuarters =
movingAverage.movingAverageForFourQuarters;
               TimeLines.CentralMovingAverageValues =
movingAverage.centralMovingAverageValues;
               TimeLines.MovingAverageEstimation =
movingAverage.movingAverageEstimation;
               TimeLines.SeasonalComponent =
seasonalComponent.seasonalComponent;
               TimeLines.averageValueOfSeasonalComponents =
seasonalComponent.averageValueOfSeasonalComponents;
               TimeLines.sumOfaverageValueOfSeasonalComponents =
seasonalComponent.sumOfaverageValueOfSeasonalComponents;
               TimeLines.correctiveFactor =
seasonalComponent.correctiveFactor;
               TimeLines.correctedSeasonalVariation =
seasonalComponent.correctedSeasonalVariation;
               TimeLines.numberOfForecastingValues =
additiveModel.numberOfForecastingValues;
               TimeLines.forecastingValuesIndexes =
additiveModel.forecastingValuesIndexes;
               TimeLines.seasonalVariation = additiveModel.seasonalVariation;
               TimeLines.valuesWOSeasonalVariation =
additiveModel.valuesWOSeasonalVariation;
               TimeLines.coeficient1 = additiveModel.coeficient1;
```

```
                TimeLines.coeficient2 = additiveModel.coeficient2;
                TimeLines.trendValues = additiveModel.trendValues;
                TimeLines.trendValuesWithSeasonalVariation =
additiveModel.trendValuesWithSeasonalVariation;
                TimeLines.errorsE = additiveModel.errorsE;
                TimeLines.errorsModule = additiveModel.errorsModule;
                TimeLines.errorsSquare = additiveModel.errorsSquare;
                TimeLines.determinationCoefficient =
additiveModel.determinationCoefficient;
                TimeLines.durbinWatsonStatistic =
additiveModel.durbinWatsonStatistic;
                TimeLines.averageAbsoluteDeviation =
additiveModel.averageAbsoluteDeviation;
                TimeLines.averageRelativeError =
additiveModel.averageRelativeError;
            }
            if (radioButton2.Checked == true)
            {
                TimeLines.isModelChosen = false;

                movingAverage =
MovingAverageForMultiplicativeModel(input.salesIndex, input.salesIndexes,
input.salesValues);
                seasonalComponent =
SeasonalComponentCalculationForMultiplicativeModel(movingAverage.movi
ngAverageEstimation);
                additiveModel = MultiplicativeModel(input.salesIndex,
input.salesIndexes, input.salesValues,
seasonalComponent.correctedSeasonalVariation,
Convert.ToInt32(numericUpDown1.Value));
                TimeLines.SalesIndex = input.salesIndex;
                TimeLines.SalesIndexes = input.salesIndexes;
                TimeLines.SalesValues = input.salesValues;
                TimeLines.MovingAverageForFourQuarters =
movingAverage.movingAverageForFourQuarters;
                TimeLines.CentralMovingAverageValues =
movingAverage.centralMovingAverageValues;
                TimeLines.MovingAverageEstimation =
movingAverage.movingAverageEstimation;
                TimeLines.SeasonalComponent =
seasonalComponent.seasonalComponent;
                TimeLines.averageValueOfSeasonalComponents =
seasonalComponent.averageValueOfSeasonalComponents;
                TimeLines.sumOfaverageValueOfSeasonalComponents =
seasonalComponent.sumOfaverageValueOfSeasonalComponents;
```

```
                TimeLines.correctiveFactor =
seasonalComponent.correctiveFactor;
                TimeLines.correctedSeasonalVariation =
seasonalComponent.correctedSeasonalVariation;
                TimeLines.numberOfForecastingValues =
additiveModel.numberOfForecastingValues;
                TimeLines.forecastingValuesIndexes =
additiveModel.forecastingValuesIndexes;
                TimeLines.seasonalVariation = additiveModel.seasonalVariation;
                TimeLines.valuesWOSeasonalVariation =
additiveModel.valuesWOSeasonalVariation;
                TimeLines.coeficient1 = additiveModel.coeficient1;
                TimeLines.coeficient2 = additiveModel.coeficient2;
                TimeLines.trendValues = additiveModel.trendValues;
                TimeLines.trendValuesWithSeasonalVariation =
additiveModel.trendValuesWithSeasonalVariation;
                TimeLines.errorsE = additiveModel.errorsE;
                TimeLines.errorsModule = additiveModel.errorsModule;
                TimeLines.errorsSquare = additiveModel.errorsSquare;
                TimeLines.determinationCoefficient =
additiveModel.determinationCoefficient;
                TimeLines.durbinWatsonStatistic =
additiveModel.durbinWatsonStatistic;
                TimeLines.averageAbsoluteDeviation =
additiveModel.averageAbsoluteDeviation;
                TimeLines.averageRelativeError =
additiveModel.averageRelativeError;
            }
            OpenChildForm(new Forms.MovingAverage(),
btn_MovingAverage);
            ForecastingValuesTableFill(TimeLines.SalesIndex,
TimeLines.numberOfForecastingValues,
TimeLines.trendValuesWithSeasonalVariation);
            dataGridView_ForecastingValues.Visible=true;
            btn_MovingAverage.Enabled = true;
            btn_SeasolanComponent.Enabled = true;
            btn_ModelData.Enabled = true;
            btn_ModelChart.Enabled = true;
            label3.Visible=true;
            btn_Clear.Enabled = true;
        }
    }

    private void groupBox1_Enter(object sender, EventArgs e)
    {
```

```
        }

        private void groupBox1_Paint(object sender, PaintEventArgs e)
        {
            Graphics gfx = e.Graphics;
            Pen p = new Pen(Color.FromArgb(77, 100, 141), 1);
            gfx.DrawLine(p, 0, 5, 0, e.ClipRectangle.Height + 25);
            gfx.DrawLine(p, 0, 5, 8, 5);
            gfx.DrawLine(p, 77, 5, e.ClipRectangle.Width - 2, 5);
            gfx.DrawLine(p, e.ClipRectangle.Width - 2, 5, e.ClipRectangle.Width -
2, e.ClipRectangle.Height - 2);
            gfx.DrawLine(p, e.ClipRectangle.Width - 2, e.ClipRectangle.Height -
2, 0, e.ClipRectangle.Height - 2);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();

            WelcomeScreen welcomeScreen = new WelcomeScreen();

            welcomeScreen.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }

        private void panel2_Paint(object sender, PaintEventArgs e)
        {

        }

        private void btn_Clear_Click(object sender, EventArgs e)
        {
            ClearForm();
        }

        private void Save_btn_Click(object sender, EventArgs e)
        {
            MovingAverage movingAverage = new MovingAverage();
            SaveFileDialog saveFileDialog = new SaveFileDialog();
```

```
        saveFileDialog.Filter = "Файлы Excel (*.xlsx; *.xlsm) | *.xlsx;
*.xlsm";
        if (saveFileDialog.ShowDialog() == DialogResult.OK)
        {
           try
           {
              using (var helper = new ExcelHelper())
              {
                 if (helper.Open(filePath: saveFileDialog.FileName))
                 {
                    helper.style(stylename: "Top Border Style",
Color.FromArgb(208, 225, 249), Color.FromArgb(77, 100, 141), $"A{1}",
$"AD{2}", borderAroundLineStyle: Excel.XlLineStyle.xlDash,
                    borderWeight: Excel.XlBorderWeight.xlThick,
borderLineStyle: Type.Missing, verticalAlignment:
Excel.XlVAlign.xlVAlignCenter, Excel.XlHAlign.xlHAlignLeft,isWrapText:
false) ;
                    string modelName;
                    if (TimeLines.isModelChosen == true)
                    {
                       modelName = "Адитивна";
                    }
                    else
                    {
                       modelName = "Мультиплікативна";
                    }
                    helper.Set(column: 1, row: 1, data: $"Звіт з прогнозування
об'ємів продажу за:{DateTime.Now}. Обрана модель: {modelName}") ;
                    helper.style(stylename: "Moving Average Data Style",
Color.FromArgb(0, 0, 0), Color.FromArgb(212, 253, 255), $"A{3}",
$"E{TimeLines.SalesIndexes.Length + 3}", borderAroundLineStyle:
Type.Missing, borderWeight: Excel.XlBorderWeight.xlThin, borderLineStyle:
Excel.XlLineStyle.xlContinuous, verticalAlignment:
Excel.XlVAlign.xlVAlignCenter,
Excel.XlHAlign.xlHAlignCenterAcrossSelection, isWrapText: true) ;
                    helper.Set(column: 1, row: 3, data: "Квартал");
                    helper.Set(column: 2, row: 3, data: "Об'єм продажу");
                    helper.Set(column: 3, row: 3, data: "Метод ковзного
середнього за 4-ма кварталами");
                    helper.Set(column: 4, row: 3, data: "Центрована ковзна
середня");
                    helper.Set(column: 5, row: 3, data: "Оцінка сезонної
компоненти");
                    for (int i = 0; i <= TimeLines.SalesIndexes.Length-1; i++)
                    {
```

```
                    helper.Set(column:  1, row: i+4, data:
TimeLines.SalesIndexes[i]);
                }
                for (int i = 0; i <= TimeLines.SalesValues.Length - 1; i++)
                {
                    helper.Set(column: 2, row: i + 4, data:
TimeLines.SalesValues[i]);
                }
                for (int i = 0; i <=
TimeLines.MovingAverageForFourQuarters.Length - 1; i++)
                {
                    helper.Set(column: 3, row: i + 4, data:
TimeLines.MovingAverageForFourQuarters[i]);
                }
                for (int i = 0; i <=
TimeLines.CentralMovingAverageValues.Length - 1; i++)
                {
                    helper.Set(column: 4, row: i + 4, data:
TimeLines.CentralMovingAverageValues[i]);
                }
                for (int i = 0; i <=
TimeLines.MovingAverageEstimation.Length - 1; i++)
                {
                    helper.Set(column: 5, row: i + 4, data:
TimeLines.MovingAverageEstimation[i]);
                }
                helper.style(stylename: "Seasonal Component Data Style",
Color.FromArgb(0, 0, 0), Color.FromArgb(212, 253, 255), $"A{3 +
TimeLines.SalesValues.Length + 2}", $"G{3 + TimeLines.SalesValues.Length
+ 3 + TimeLines.SeasonalComponent.GetLength(0)+1}",
                    borderAroundLineStyle: Type.Missing, borderWeight:
Excel.XlBorderWeight.xlThin, borderLineStyle:
Excel.XlLineStyle.xlContinuous, verticalAlignment:
Excel.XlVAlign.xlVAlignCenter,
                    Excel.XlHAlign.xlHAlignCenterAcrossSelection,
isWrapText: true);
                helper.Set(column: 4, row: 3+
TimeLines.SalesValues.Length+2, data: "1");
                helper.Set(column: 5, row: 3 +
TimeLines.SalesValues.Length + 2, data: "2");
                helper.Set(column: 6, row: 3 +
TimeLines.SalesValues.Length + 2, data: "3");
                helper.Set(column: 7, row: 3 +
TimeLines.SalesValues.Length + 2, data: "4");
```

```
                    helper.Set(column: 1, row: 3 +
TimeLines.SalesValues.Length + 3, data: "Оцінки сезонної компоненти");
                    helper.merge($"A{3 + TimeLines.SalesValues.Length +
3}",$"C{3 + TimeLines.SalesValues.Length + 3}");
                    helper.Set(column: 1, row: 3 +
TimeLines.SalesValues.Length +
3+TimeLines.SeasonalComponent.GetLength(0), data: "Середнє значення");
                    helper.merge($"A{3 + TimeLines.SalesValues.Length + 3 +
TimeLines.SeasonalComponent.GetLength(0)}", $"C{3 +
TimeLines.SalesValues.Length + 3 +
TimeLines.SeasonalComponent.GetLength(0)}");
                    helper.Set(column: 1, row: 3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0), data: "Скорегована сезонна
варіація");
                    helper.merge($"A{3 + TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0)}", $"C{3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0)}");
                    for (int i = 0; i <=
TimeLines.SeasonalComponent.GetLength(0) - 1; i++)
                    {
                        for (int j = 0; j <=
TimeLines.SeasonalComponent.GetLength(1)-1; j++)
                        {
                            helper.Set(column: 4+j, row: 3 +
TimeLines.SalesValues.Length + 3+i, data:
TimeLines.SeasonalComponent[i,j]);
                        }
                    }
                    for (int i = 0; i <=
TimeLines.averageValueOfSeasonalComponents.Length - 1; i++)
                    {
                        helper.Set(column: 4+i, row: 3 +
TimeLines.SalesValues.Length + 3 +
TimeLines.SeasonalComponent.GetLength(0), data:
TimeLines.averageValueOfSeasonalComponents[i]);
                    }
                    for (int i = 0; i <=
TimeLines.correctedSeasonalVariation.Length - 1; i++)
                    {
                        helper.Set(column: 4 + i, row: 3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0), data:
TimeLines.correctedSeasonalVariation[i]);
```

```
                }
                helper.style(stylename: "Seasonal Component Data Style2",
Color.FromArgb(0, 0, 0), Color.FromArgb(212, 253, 255), $"A{3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0)+1}", $"D{3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0) + 4}",
                    borderAroundLineStyle: Type.Missing, borderWeight:
Excel.XlBorderWeight.xlThin, borderLineStyle:
Excel.XlLineStyle.xlContinuous, verticalAlignment:
Excel.XlVAlign.xlVAlignTop,
                    Excel.XlHAlign.xlHAlignCenter, isWrapText: false);
                helper.Set(column: 1, row: 3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0)+2, data: "Сума середніх
значень");
                helper.merge($"A{3 + TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0) + 2}", $"C{3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0) + 2}");
                helper.Set(column: 1, row: 3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0)+4, data: "Корегуючий
коефіцієнт");
                helper.merge($"A{3 + TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0) + 4}", $"C{3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0) + 4}");
                helper.Set(column: 4, row: 3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0)+2, data:
TimeLines.sumOfaverageValueOfSeasonalComponents);
                helper.Set(column: 4, row: 3 +
TimeLines.SalesValues.Length + 4 +
TimeLines.SeasonalComponent.GetLength(0) + 4, data:
TimeLines.correctiveFactor);
                helper.style(stylename: "Model Table Data Style",
Color.FromArgb(0, 0, 0), Color.FromArgb(212, 253, 255), $"I{3}", $"Q{3 +
TimeLines.forecastingValuesIndexes.Length}",
                    borderAroundLineStyle: Type.Missing, borderWeight:
Excel.XlBorderWeight.xlThin, borderLineStyle:
Excel.XlLineStyle.xlContinuous, verticalAlignment:
Excel.XlVAlign.xlVAlignCenter,
                    Excel.XlHAlign.xlHAlignCenter, isWrapText: false);
                helper.Set(column: 9, row: 3, data: "t");
```

```
                    for (int i = 0; i <=
TimeLines.forecastingValuesIndexes.Length - 1; i++)
                    {
                        helper.Set(column: 9, row: 4 + i, data:
TimeLines.forecastingValuesIndexes[i]);
                    }
                    helper.Set(column: 10, row: 3, data: "Yt");
                    for (int i = 0; i <= TimeLines.SalesValues.Length - 1; i++)
                    {
                        helper.Set(column: 10, row: 4 + i, data:
TimeLines.SalesValues[i]);
                    }
                    helper.Set(column: 11, row: 3, data: "Si");
                    for (int i = 0; i <= TimeLines.seasonalVariation.Length - 1;
i++)
                    {
                        helper.Set(column: 11, row: 4 + i, data:
TimeLines.seasonalVariation[i]);
                    }
                    if (TimeLines.isModelChosen == true)
                    {
                        helper.Set(column: 12, row: 3, data: "Yt-Si");
                        helper.Set(column: 14, row: 3, data: "T+Si");
                    }
                    if (TimeLines.isModelChosen == false)
                    {
                        helper.Set(column: 12, row: 3, data: "Yt/Si");
                        helper.Set(column: 14, row: 3, data: "T*Si");
                    }

                    for (int i = 0; i <=
TimeLines.valuesWOSeasonalVariation.Length - 1; i++)
                    {
                        helper.Set(column: 12, row: 4 + i, data:
TimeLines.valuesWOSeasonalVariation[i]);
                    }
                    for (int i = 0; i <=
TimeLines.trendValuesWithSeasonalVariation.Length - 1; i++)
                    {
                        helper.Set(column: 14, row: 4 + i, data:
TimeLines.trendValuesWithSeasonalVariation[i]);
                    }
                    helper.Set(column: 13, row: 3, data: "T");
                    for (int i = 0; i <= TimeLines.trendValues.Length - 1; i++)
                    {
```

```
                    helper.Set(column: 13, row: 4 + i, data:
TimeLines.trendValues[i]);
                }
                helper.Set(column: 15, row: 3, data: "e");
                for (int i = 0; i <= TimeLines.errorsE.Length - 1; i++)
                {
                    helper.Set(column: 15, row: 4 + i, data:
TimeLines.errorsE[i]);
                }
                helper.Set(column: 16, row: 3, data: "|e|");
                for (int i = 0; i <= TimeLines.errorsModule.Length - 1; i++)
                {
                    helper.Set(column: 16, row: 4 + i, data:
TimeLines.errorsModule[i]);
                }
                helper.Set(column: 17, row: 3, data: "e^2");
                for (int i = 0; i <= TimeLines.errorsSquare.Length - 1; i++)
                {
                    helper.Set(column: 17, row: 4 + i, data:
TimeLines.errorsSquare[i]);
                }
                helper.style(stylename: "Model Table Data Style2",
Color.FromArgb(0, 0, 0), Color.FromArgb(212, 253, 255), $"I{3 +
TimeLines.forecastingValuesIndexes.Length+2}", $"L{3 +
TimeLines.forecastingValuesIndexes.Length + 10}",
                    borderAroundLineStyle: Type.Missing, borderWeight:
Excel.XlBorderWeight.xlThin, borderLineStyle:
Excel.XlLineStyle.xlContinuous, verticalAlignment:
Excel.XlVAlign.xlVAlignCenter,
                    Excel.XlHAlign.xlHAlignCenter, isWrapText: false);
                helper.Set(column: 9, row:
3+TimeLines.forecastingValuesIndexes.Length+2, data: "Рівняння лінії
тренду");
                helper.merge($"I{3 +
TimeLines.forecastingValuesIndexes.Length + 2}", $"J{3 +
TimeLines.forecastingValuesIndexes.Length + 2}");
                helper.Set(column: 9, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 4, data: "САО");
                helper.merge($"I{3 +
TimeLines.forecastingValuesIndexes.Length + 4}", $"J{3 +
TimeLines.forecastingValuesIndexes.Length + 4}");
                helper.Set(column: 9, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 6, data: "СООП");
```

```
                helper.merge($"I{3 +
TimeLines.forecastingValuesIndexes.Length + 6}", $"J{3 +
TimeLines.forecastingValuesIndexes.Length + 6}");
                helper.Set(column: 9, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 8, data: "Коефіцієнт
детермінації R^2");
                helper.merge($"I{3 +
TimeLines.forecastingValuesIndexes.Length + 8}", $"J{3 +
TimeLines.forecastingValuesIndexes.Length + 8}");
                helper.Set(column: 9, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 10, data: "Критерій Дарбіна-
Уотсона DW");
                helper.merge($"I{3 +
TimeLines.forecastingValuesIndexes.Length + 10}", $"J{3 +
TimeLines.forecastingValuesIndexes.Length + 10}");
                helper.Set(column: 11, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 2, data:
$"y={TimeLines.coeficient2:f4}x+{TimeLines.coeficient1:f4}");
                helper.merge($"K{3 +
TimeLines.forecastingValuesIndexes.Length + 2}", $"L{3 +
TimeLines.forecastingValuesIndexes.Length + 2}");
                helper.Set(column: 11, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 4, data:
TimeLines.averageAbsoluteDeviation);
                helper.Set(column: 11, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 6, data:
TimeLines.averageRelativeError);
                helper.Set(column: 11, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 8, data:
TimeLines.determinationCoefficient);
                helper.Set(column: 11, row: 3 +
TimeLines.forecastingValuesIndexes.Length + 10, data:
TimeLines.durbinWatsonStatistic);
                helper.style(stylename: "Forecasting Data Style",
Color.FromArgb(0, 0, 0), Color.FromArgb(163, 230, 181), $"S{3}", $"T{3 +
TimeLines.numberOfForecastingValues - TimeLines.SalesIndex}",
                    borderAroundLineStyle: Type.Missing, borderWeight:
Excel.XlBorderWeight.xlThin, borderLineStyle:
Excel.XlLineStyle.xlContinuous, verticalAlignment:
Excel.XlVAlign.xlVAlignCenter,
                    Excel.XlHAlign.xlHAlignCenterAcrossSelection,
isWrapText: true);
                helper.Set(column: 19, row: 3, data: $"Прогнозовані
значення на наступні {TimeLines.numberOfForecastingValues -
TimeLines.SalesIndex} кварталів");
```

```
                    helper.merge($"S{3}", $"T{3}");
                    for (int i = 0; i < TimeLines.numberOfForecastingValues -
TimeLines.SalesIndex; i++)
                    {
                        helper.Set(column: 19, row: 4 + i, data:
TimeLines.forecastingValuesIndexes[TimeLines.SalesIndex + i]);
                    }
                    for (int i = 0; i < TimeLines.numberOfForecastingValues -
TimeLines.SalesIndex; i++)
                    {
                        helper.Set(column: 20, row: 4 + i, data:
TimeLines.trendValuesWithSeasonalVariation[TimeLines.SalesIndex + i]);
                    }

                    helper.chart(4, 3+TimeLines.SalesValues.Length);
                    helper.style(stylename: "Bottom Border Style",
Color.FromArgb(208, 225, 249), Color.FromArgb(77, 100, 141),
$"A{4+TimeLines.SalesValues.Length+2+
TimeLines.SeasonalComponent.GetLength(0)+4+2+9}", $"AD{4 +
TimeLines.SalesValues.Length + 2 +
TimeLines.SeasonalComponent.GetLength(0) + 4 + 2 + 10}",
borderAroundLineStyle: Excel.XlLineStyle.xlDash,
                    borderWeight: Excel.XlBorderWeight.xlThick,
borderLineStyle: Type.Missing, verticalAlignment:
Excel.XlVAlign.xlVAlignCenter, Excel.XlHAlign.xlHAlignLeft, isWrapText:
false);
                    helper.Save();

                }
            }
        }
        catch (Exception ex) { Console.WriteLine(ex.Message); }
    }
  }
}
```

Додаток Б

Копії публікації з темою магістерської роботи

Інформаційні технології в науці, виробництві та підприємництві
Київський національний університет технологій та дизайну

ЛЕВКОВЕЦЬ Д. С., ДЕМКІВСЬКА Т. І.

ПРОГНОЗУВАННЯ ОБ`ЄМІВ ПРОДАЖУ МАГАЗИНІВ ВЗУТТЯ

LEVKOVETS D. S., DEMKIVSKA T. I.

SOFTWARE DEVELOPMENT FOR FORECASTING SALES VOLUMES OF SHOE STORES

Annotation – The article presents the idea of developing software for forecasting sales volumes of shoe stores based on additive and multiplicative time series models. The program should make it possible to analyze the impact of seasonal components on sales volumes for a certain quarter, obtain model estimates and build a forecast of sales volumes for several future quarters based on the best model.

As a result of the work, the input time series were smoothed using the moving average method and the seasonal component was estimated. The seasonal component was removed from the levels of the series and smoothed data were obtained. The parameters of the trend line equation were found using the method of least squares. The adequacy of the models was also checked, and on the basis of the best one, a forecast of sales volumes for the next few quarters was made.

Keywords: application development, additive model, multiplicative model, time series, least squares method, moving average method, trend, seasonal component, forecasting.

Вступ

Для кожного магазину чи підприємства головною метою є отримання прибутку, а ціллю - його максимізація. Програма, про яку йдеться в статті, дає змогу магазинові або підприємстві отримати дані, як об'єм продажу змінюється залежно від кварталу в році. Такі дані дають змогу зрозуміти який товар і в який період року краще продавати для максимізації прибутку.

Також програма дає змогу здійснити прогноз, на який об'єм продажу приблизно магазин може розраховувати в наступні квартали. Отримавши прогноз магазин може виготовити або замовити оптимальну кількість товару, що дозволяє зекономити на матеріалах та логістиці, та таким чином отримати максимальний прибуток.

Постановка завдання

Метою даного дослідження є розробка програмного забезпечення для прогнозування об'єму продажу товару магазинів взуття, на базі адитивної або мультиплікативної моделей часового ряду. Програма повинна виконати розрахунки для двох моделей, по отриманим даним обрати найкращу та виконати проноз даних за обраною моделлю.

Завданням дослідження є розробка додатку, який буде доступним для всіх популярних та широко використовуваних настільних та планшетних комп'ютерів, підтримуючих платформу .NET Framework.

Основна частина

В процесі даної роботи досліджено процес розробки, тестування та впровадження програмного додатку створеного на платформі .NET Framework, що підтримується різними операційними системами. Предметом дослідження є технології, засоби розробки й мови

програмування для реалізації програмного забезпечення на базі мультиплікативної або адитивної моделей часового ряду на платформі .NET Framework.

Візьмемо даний часовий ряд, як експерементні дані:

Квартал	1	2	3	4	5	6	7	8	9	10	11	12	13
Об'єм продажу	239	201	182	297	324	278	257	384	401	360	335	462	481

Побудова адитивної моделі складається з наступних кроків:

1. Вирівнювання вхідного часового ряду методом ковзної середньої.

Квартал	Об'єм продажу	МКС	ЦКС	ОСК
1	239			
2	201			
3	182	229,75	240,375	-58,375
4	297	251	260,625	36,375
5	324	270,25	279,625	44,375
6	278	289	299,875	-21,875
7	257	310,75	320,375	-63,375
8	384	330	340,25	43,75
9	401	350,5	360,25	40,75
10	360	370	379,75	-19,75
11	335	389,5	399,5	-64,5
12	462	409,5		
13	481			

Рис.1 Вирівнювання методом ковзної середньої

	Квартал						
	1	2	3	4			
Оцінка сезонної варіації	0	0	-58,38	36,375			
	44,375	-21,88	-63,38	43,75			
	40,75	-19,75	-64,5	0	Сума середніх значень	коефіцієнт	
Середнє	42,563	-20,81	-62,08	40,063	-0,2708	-0,068	
Скоректована сезонна компонента Si	42,63	-20,74	-62,02	40,13	0		
					Сума значень сезонної компоненти повинна = 0		

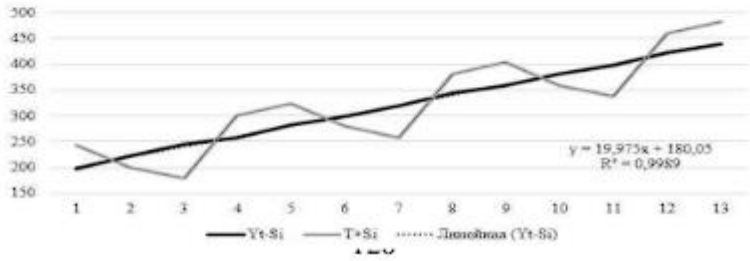
2. Розрахунок значення сезонної компоненти.

Рис.2 Розрахунок значень сезонної компоненти

Вилучаємо корегуючий коефіцієнт з середніх значень оцінки та отримуємо скореговану сезонну компоненту Si.

3. За допомогою методу найменших квадратів отрмуємо показники лінії тренду: a=180,0504, b= 19,9745.

4. Аналітичне вирівнювання рівнів ряду з використанням отриманого рівняння тренду. Рівняння тренду має наступний вигляд: T=180,0504 +



19,9745 *t.

t	Yt	Si	Yt-Si	T	T+Si	e	|e|	e^2
1	239	42,63	196,37	200,03	242,66	-3,657	3,6571	13,374
2	201	-20,74	221,74	220	199,26	1,7434	1,7434	3,0394
3	182	-62,02	244,02	239,98	177,96	4,0397	4,0397	16,319
4	297	40,13	256,87	259,95	300,08	-3,081	3,0806	9,4901
5	324	42,63	281,37	279,92	322,56	1,4449	1,4449	2,0877
6	278	-20,74	298,74	299,9	279,15	-1,155	1,1546	1,3331
7	257	-62,02	319,02	319,87	257,86	-0,858	0,8583	0,7366
8	384	40,13	343,87	339,85	379,98	4,0214	4,0214	16,172
9	401	42,63	358,37	359,82	402,45	-1,453	1,4531	2,1115
10	360	-20,74	380,74	379,8	359,05	0,9474	0,9474	0,8976
11	335	-62,02	397,02	399,77	337,76	-2,756	2,7563	7,5971
12	462	40,13	421,87	419,75	459,88	2,1234	2,1234	4,5088
13	481	42,63	438,37	439,72	482,35	-1,351	1,3511	1,8255

Рис.4 Графік прогнозу

Рис.5 Розрахунок вирівняних значень Т і помилок Е

5. Перевірка адекватності моделі шляхом пошуку коефіцієнту детермінації та критерію Дарбіна-Уотсона.

R^2=0,996, DW=1,9687.

Модель з такими показниками можна вважати адекватною.

Виконавши всі кроки для мультиплікативної моделі, отримаємо:

R^2=0,9905, DW=1,9497.

Мультиплікативну модель теж є адекватною.

Обрати кращу модель можна порівнявши їх середні абсолютні відхилення (САО) та середні відхилення відносно помилок(СООП).

Для адитивної моделі: САО=2,2; СООП=0,77%.

Для мультиплікативної моделі: САО=7,21; СООП=2,38%.

Обираємо Адитивну модель, через менше значення похибок. На її основі проведемо прогноз значень на 14 та 15 квартали.

$F_{14}=T_{14}+Si_{14}$=438,95; $F_{15}=T_{15}+Si_{15}$=417,6543

Висновок

Методом ковзного середнього здійснено вирівнювання вхідного часового ряду та отримано оцінку сезонної компоненти для адитивної та мультиплікативної моделей. Виконано аналітичне вирівнювання рівнів ряду та оцінено параметри моделей. Також здійснена перевірка адекватності адитивної та мультиплікативної моделей. З отриманих даних обрано кращу модель та на її основі побудовано прогноз об'єму продажу магазину взуття.

Розроблений програмний продукт сприяє полегшенню у аналізуванні залежності об'єму продажу від сезону, а також дозволяє отримати приблизний прогноз об'єму продажу в майбутньому.

Література

Додаток В

Розроблення програмного забезпечення для прогнозування об'ємів продажу інтернет-магазину взуття

Виконавець – Левковець Д. С.
Науковий керівник – Демківська Т. І.

Мета дослідження

Розробка програмного забезпечення для прогнозування об'єму продажу товару інтернет-магазину взуття на базі адитивної або мультиплікативної моделі часового ряду

Актуальність

Прогнозування часових рядів залишається дуже актуальною та важливою задачею на сучасний день. Серед областей, в яких досі використовують прогнозування часових рядів є: сфера фінансів, економіка, метеорологія, медицина, виробництво та логістика та соціальні науки.

Алгоритмічне забезпечення

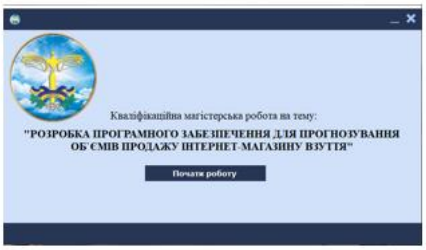
Алгоритм прогнозу для інтернет-магазину взуття складається з таких кроків:

1. Вирівнювання вхідного часового ряду методом ковзної середньої за 4-ма кварталами.
2. Пошук сезонної компоненти.
3. Розрахунок показників лінії тренду методом найменших квадратів.
4. Аналітичне вирівнювання рівнів ряду з використанням отриманого рівняння тренду.
5. Перевірка адекватності моделі.
6. Прогнозування майбутніх значень.

Мова програмування та середовище для реалізації

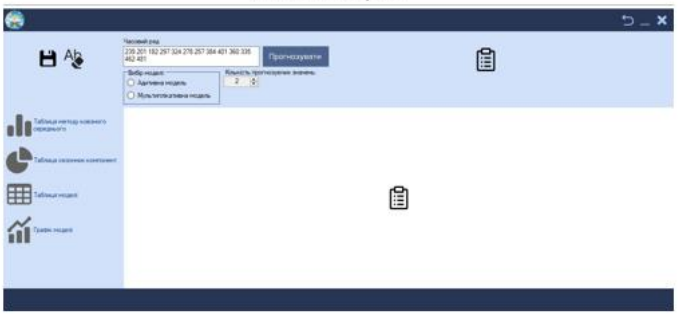
Було вирішено створити класичний віконний додаток з інтерфейсом користувача для операційної системи **Windows**, на мові **C#**. Проєкт розроблявся в інтегрованому середовищі розробки (IDE) **Visual Studio**

Демонстрація додатку

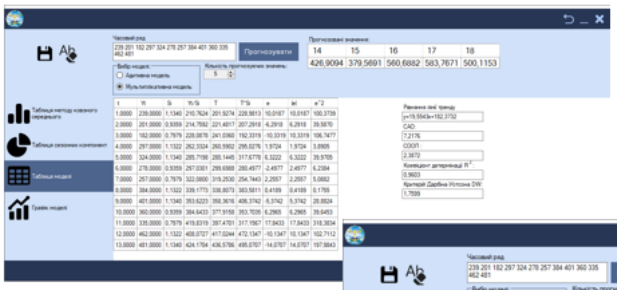


Вітальний екран

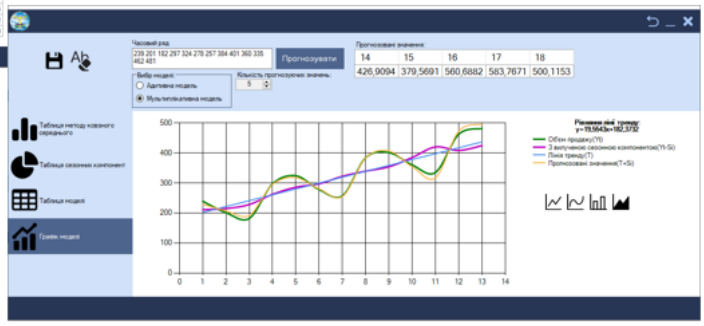
Головна форма



Відображення інформації



Таблична інформація



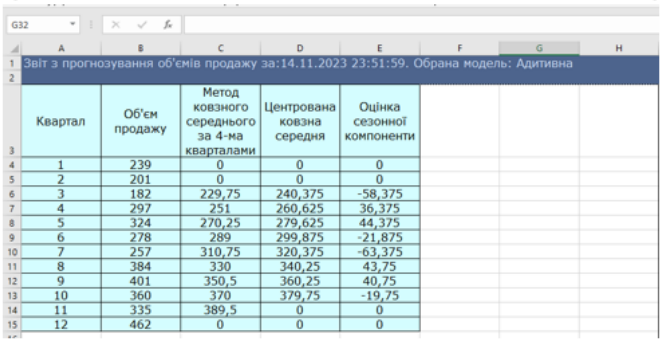
Графічна інформація

Можливості інтерфейсу програми

- Вся інформація зручно представлена в вигляді текстових боксів, таблиць та графіків;
- Користувач може зручно перемикатися між вкладками програми;
- Перепрограмована панель системних кнопок під спільний дизайн додатку;
- Система захисту вхідної інформації, користувач інформується про причину помилки;
- Можливість очистити форму та привести її до початкового стану;
- Користувач може зберегти результат дослідження, згенерувавши звіт-таблицю Excel;

Взаємодія з зовнішнім програмним забезпеченням

В проєкті реалізовано взаємодію додатку з професійним застосуванням Microsoft Excel. Додаток в фоновому режимі викликає excel, створює та зберігає звіт на комп'ютері користувача. Така можливість реалізована за допомогою бібліотеки microsoft.office.interop.excel.



Звіт з прогнозування об'ємів продажу за:14.11.2023 23:51:59. Обрана модель: Адитивна				
Квартал	Об'єм продажу	Метод ковзного середнього за 4-ма кварталами	Центрована ковзна середня	Оцінка сезонної компоненти
1	239	0	0	0
2	201	0	0	0
3	182	229,75	240,375	-58,375
4	297	251	260,625	36,375
5	324	270,25	279,625	44,375
6	278	289	299,875	-21,875
7	257	310,75	320,375	-63,375
8	384	330	340,25	43,75
9	401	350,5	360,25	40,75
10	360	370	379,75	-19,75
11	335	389,5	0	0
12	462	0	0	0

Частина звіту в вигляді excel-таблиці

Висновки

В результаті виконаного дослідження було реалізовано та успішно виконано всі поставлені задачі, а саме:

- Проаналізовано вплив часових рядів на економічні процеси, а саме процеси інтернет-торгівлі та торгівлі взуттям.
- Розроблено алгоритм прогнозування об'ємів продажу інтернет-магазинів взуття в майбутній момент часу.
- Створено програмне забезпечення на мові програмування C#, що реалізує алгоритм прогнозування.

Під час виконання даного проєкту були набуті нові знання та навички в сфері об'єктно орієнтованого програмування та розробки інтерфейсів користувача.