Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «Дослідження часових рядів за критерієм часткової схожості»
за освітньою програмою **12 Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студент групи ПЗ2221: /Тімур КДИРОВ/

Керівник: / Віктор ШИНКАРЕНКО/

Нормоконтролер: /Світлана ВОЛКОВА /

Засвідчую, що у цій роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент

Дніпро – 2024 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note

to Master's Thesis

on the topic: «Research of time series based on the criteria of partial similarity»

according to educational curriculum **12 software engineering**
in the speciality: **121 software engineering**

Done by the student of the group PZ2221: student ____ / Timur KDYROV /
Scientific Supervisor: docent ____ /Viktor SHYNKARENKO/
Normative controller: docent ____ / Svitlana VOLKOVA/

Dnipro – 2024

Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем

Кафедра: Комп’ютерні інформаційні технології

Рівень вищої освіти: магістр

Освітня програма: Інженерія програмного забезпечення

Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ____КІТ

_____ Вадим ГОРЯЧКІН

_____ грудня 2023 р.

ЗАВДАННЯ

На кваліфікаційну роботу ____Магістр

студенту Кдирову Тімуру Айткалійовичу

1. Тема дипломної роботи: Дослідження часових рядів за критерієм часткової схожості

   Керівник роботи: Шинкаренко Віктор Іванович

   затверджені наказом 1196 ст від 05.12. 2022 року
2. Строк подання студентом роботи 23.12. 2023 року
3. Вихідні дані до дипломної роботи: поточні дані аналізу часових рядів
4. Зміст пояснювальної записки (перелік питань до розробки):
   - 4.1. Аналітична частина: вступ, опис предметної області аналізу часових рядів;
   - 4.2. Основна частина: алгоритмізація виконання аналізу часових рядів, методи аналізу часових рядів з використанням нейронних мереж, програмна реалізація алгоритму аналізу часових рядів, остаточні висновки, література;
5. Перелік демонстраційного матеріалу:
   - 5.1. презентація;
   - 5.2. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вступ	05.03.22 – 21.04.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	03.06.23 – 07.07.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	10.07.23 – 15.07.23	
4	Постановка задачі, технічне завдання	16.07.23 – 29.07.23	30%
5	Виконання досліджень	07.08.23 – 30.09.23	60%
6	Оформлення тез доповідей	30.10.23 – 05.12.23	
7	Оформлення пояснювальної записки	01.05.23 – 31.12.23	
8	Розробка демонстраційних матеріалів	05.01.24 – 14.01.24	100%
9	Подання кваліфікаційної роботи до кафедри	15.01.24 – 17.01.24	
10	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	23.01.24	

Студент _____ /Тімур Кдиров/

Керівник роботи _____ / Віктор ШИНКАРЕНКО /

РЕФЕРАТ

Пояснювальна записка складається зі вступу, 4 розділів, остаточних висновків, бібліографічного списку та 3 додатків:

– вступ – в даному розділі описується сутність розробки, її актуальність.

Складається з 2 сторінок;

– опис предметної області аналізу часових рядів – розділ надає основу для подальшого дослідження аналізу часових рядів, визначаючи ключові аспекти та напрямки в даній області.

Складається з 28 сторінок;

– алгоритмізація виконання аналізу часових рядів – у цьому розділі досліджуємо залежності та методи для розуміння та впровадження алгоритмів аналізу часових рядів з використанням систематизованого підходу.

Складається з 15 сторінок;

– методи аналізу часових рядів з використанням нейронних мереж – цей розділ відкриває можливості та переваги використання нейронних мереж у сучасному аналізі часових рядів.

Складається з 19 сторінок;

– програмна реалізація алгоритму аналізу часових рядів – розділ забезпечує повний огляд програмної реалізації, розкриваючи як програма обробляє та виводить результати аналізу часових рядів.

Складається з 19 сторінки;

– остаточні висновки. Складається з 1 сторінки;

– список літератури – включає в себе бібліографічний список використаної літератури. Складається з 6 сторінок;

– додатки – у додатках наведені технічне завдання із керівництвом користувача та текст програми.

Таблиць – 2, рисунків – 56, бібліографія – 67

Ключові слова: часові ряди, моделі, авторегресія, тренд.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ

СППР – система підтримки прийняття рішень

ОПР – особа, яка приймає рішення

СУБД – система управління базами даних

OLAP (від англ. Online analytical processing) – технологія обробки інформації, що дозволяє швидко отримувати відповіді на багатовимірні аналітичні запити

Data Mining – виявлення прихованих закономірностей або взаємозв'язків між змінними у великих масивах необроблених даних

DSS (від англ. Decision support system) – система підтримки прийняття рішень

ПЗ – програмне забезпечення

LCA (від англ. Life-cycle assessment) – оцінка життєвого циклу

IDE (від англ. Integrated Development Environment) – інтегроване середовище розробки

CSV (від англ. Comma-Separated Values) – текстовий формат, призначений для представлення табличних даних.

ВСТУП

У фінансовій сфері часто виникає завдання прогнозування короткострокових тенденцій часового ряду без безпосереднього розрахунку прогнозних значень. Це випливає з того, що фінансові часові ряди часто проявляють нестійкий характер коливань, які можуть наближатися до випадкових. У таких випадках розробляються моделі, які спроможні передбачати знаки приростів значень часового ряду на одну точку вперед з необхідною максимальною точністю. Особливістю таких моделей і методів є їхня здатність знаходити стійку залежність поточного спостереження від попередніх.

На основі цієї залежності розраховуються прогнози знаків приростів на наступну точку, тобто з періодом 1. Такі моделі часто використовуються для визначення напрямку руху часових рядів валютних пар і можуть служити для визначення точок зміни тенденцій. При прогнозуванні знаків приростів використовуються специфічні моделі та методи. Важливо відзначити, що перед впровадженням таких моделей необхідно провести аналіз вхідного фінансового часового ряду на наявність абсолютної випадковості.

Цей аналіз може бути здійснений на основі фрактального аналізу або використання критеріїв поворотних точок, розподілу довжини фази та інших методів.

При розробці прогнозу в даному випадку основні етапи включають:

1. Прогнозна ретроспекція:

- Встановлення об’єкту прогнозування, включаючи передпрогнозний аналіз об’єкту та оцінку його параметрів.

2. Вибір методів прогнозування та побудова моделі:

- Вибір ефективних методів прогнозування та створення моделі прогнозування, її формалізація.

3. Побудова проспекції:

- Розрахунок прогнозу на визначений період, орієнтований на результати побудованої моделі.

4. Оцінка прогнозу і верифікація:

- Проведення оцінки прогнозу та його верифікація для визначення його точності та відповідності реальним даним.

Зазвичай традиційні математичні моделі не є достатньо ефективними для точного прогнозування фінансових часових рядів, які мають випадковий характер коливань. Тому виникає необхідність розробки моделей та методів, спрямованих на прогнозування знаків приростів у таких рядах. Дослідження в цьому напрямку спрямоване на створення методу, який забезпечив би максимальну точність прогнозу знаку приросту часового ряду. Для цього можуть використовуватися методи фрактального аналізу інформації, інтелектуального аналізу часових рядів, такі як індексація за методами найближчого сусіда та K-найближчих сусідів тощо.

Об'єкт дослідження – розробка програмного коду та desktop-додатку візуалізації даних для дослідження часових рядів за критерієм часткової схожості.

Предмет дослідження – математичні методи візуалізації даних та скорингу.

Мета роботи - розглянути методи побудови дослідження часових рядів за критерієм часткової схожості.

Розділ 1 Опис предметної області аналізу часових рядів

1.1 Постановка задачі аналізу часових рядів

Моделювання та аналіз часових рядів є науковими способами прогнозування майбутнього. При роботі з емпіричними даними часових рядів, як правило, приходять до класичних книг Бокса і методології Дженкінса для моделей часових рядів у 1970-х роках, в яких вони ввели моделі авторегресії з інтегрованим ковзним середнім (ARIMA) для прогнозування майбутньої поведінки часового ряду. Однак моделі такі як: пуассонівські процеси, марківські процеси, авторегресія (AR), ковзний середній (MA), авторегресія з ковзним середнім (ARMA) і самі процеси ARIMA дозволяють захоплювати лише близьку залежність (short-range dependence (SRD)). Вони належать до звичайних цілочисельних моделей [42].

В аналізі часових рядів іншим традиційним припущенням є те, що зв’язок між значеннями в різні моменти часу швидко зменшується, коли різниця в часі або відстані зростає. Залежність великої дальності (long-range dependence (LRD)), яка також називається довготривалою пам’яттю (long memory) або довготривалою стійкістю (long-range persistence), є явищем, яке може виникнути при аналізі даних просторового або часового ряду. LRD вперше було висвітлено у гідрологічних даних британським гідрологом H. E. Hurst, а потім інші статистичні дані в економетриці, мережевому трафіку, лінгвістиці та науках про Землю тощо. Ефект зв’язку між значеннями при різних часових розділеннях. Таким чином LRD також вказує на те, що розпад автокореляційної функції (АКФ) є алгебраїчним і більш повільним, ніж експоненціальний розпад, так що площа під кривою функції є нескінченною. Таку поведінку можна також назвати зворотною затримкою силового закону [43].

На відміну від аналітичних результатів лінійних диференціальних рівнянь цілочисельного порядку, які представлені поєднанням експоненціальних функцій, аналітичні результати лінійних диференціальних

рівнянь дробового порядку представлені функцією Міттага-Лефлера (Mittag-Leffler), яка по суті володіє силовим законом і асимптотичною поведінкою [38].

У зв'язку зі зростаючим попитом на моделювання та аналіз LRD та самоподібності у часових рядах, таких як фінансові дані про мережі зв'язку та підводний шум, технологія обробки сигналів дробового порядку (fractional order signal processing (FOSP)) стає бурхливою областю досліджень. Більш того, дробове перетворення Фур'є (the fast Fourier transform (FFT)), стало одним з найбільш цінних і часто використовуваних методів у частотній області систем дробового порядку. У порівнянні з звичайними цілочисельними моделями, модель ARFIMA дає краще пристосування і результат при роботі з даними, які мають властивість LRD [24].

1.2 Класифікація прогнозів при виконанні аналізу часових рядів

Якість моделі регресії пов'язують із адекватністю модічи спостерігається (емпіричним) даним. Перевірка адекватності (або відповідності) моделі регресії спостережуваним даним проводиться на основі аналізу залишків - εі. При побудові рівняння регресії ми можемо розбити значення у кожному спостереженні на 2 складові: $y_i = \hat{y}_i + \varepsilon_i$. Залишок є відхиленням фактичного значення залежної змінної від значення даної змінної, отримано розрахунковим шляхом: $\varepsilon_i = y_i - \hat{y}_i$. Якщо εі=0, то для всіх спостережень фактичні значення залежної змінної збігаються з розрахунковими (теоретичними) значеннями. Графічно це означає, що теоретична лінія регресії (лінія, побудована за функцієюу=a0+a1x) проходить через усі точки кореляційного поля, що можливо лише за строго функціонального зв'язку. Отже, результативна ознака повністю обумовлена впливом фактора х. Насправді, зазвичай, має місце деяке розсіювання точок кореляційного поля щодо теоретичної лінії регресії, тобто. відхилення емпіричних даних від теоретичних ($\varepsilon_i \neq 0$). Величина цих відхилень лежить в основі розрахунку показників якості (адекватності) рівняння [32].

Для оцінки якості регресійних моделей використовують також к-т множинної кореляції: $R = \sqrt{\frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}}$. Цей коефіцієнт є універсальним, т.к. він відображає тісноту зв'язку та точність моделі, а також може використовуватися за будь-якої форми зв'язку змінних. Коефіцієнт множинної кореляції, зведений у квадрат, називається коефіцієнтом детермінації: $R^2 = 1 - \frac{\sum e(i)^2}{\sum(y_i - \bar{y}_i)^2} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$.

Перейдемо до вихідних змінних x і y, виконавши потенціювання даного рівняння:

$\hat{y} = 10^{\hat{A}} \times x^b$

Отримаємо рівняння статечної моделі регресії:

$\hat{y} = a \times x^b$

Далі перевіряє якість моделі (індекс кореляції, к-т детермінації, F-критерій Фішера, середня відносна помилка) [37].

1.2.1 Побудова моделі як показової функції.

Рівняння показової моделі має вигляд: $\tilde{o} = \grave{a}b^x$

Для побудови цієї моделі необхідно провести лінеаризацію змінних. Для цього зробимо логарифмування обох частин рівняння:

$$\lg \tilde{o} = \lg a + x \lg b$$

Позначимо Y = $\lg \tilde{o}$, B = lg b, A = lg a

Отримаємо лінійне рівняння: Y = A + Bx.

Розрахуємо його параметри:

$\hat{A} = \frac{\overline{Y \times \tilde{o}} - \overline{Y} \times \overline{\tilde{o}}}{\overline{\tilde{o}^2} - \overline{\tilde{o}}^2}$

$\grave{A} = \overline{Y} - \hat{A} \times \overline{\tilde{o}}$

Перейдемо до вихідних змінних x і y, виконавши потенціювання даного рівняння:

$$\hat{y} = 10^{A} \times (10^{A})^{b} = a \times b^{b}$$

Далі перевіряє якість моделі (індекс кореляції, к-т детермінації, F-критерій Фішера, середня відносна помилка).

Розглянемо по черзі моделі, які використовуються при прогнозуванні G-методами. Наприклад об'єкти техноценозу є дискретним часовим рядом. $w_t = w_1, w_2, w_3, ..., w_n$ (Рис. 1.1)[1].

Авторрегресійна модель – це окремий випадок загальної лінійної моделі. У ній поточне значення процесу виражається як кінцева лінійна комбінація попередніх значень процесу білого шуму [3,4]. Авторегресійна модель записується так:

$$\tilde{w}_t = \phi_1 \cdot \tilde{w}_{t-1} + \phi_2 \cdot \tilde{w}_{t-2} + \ldots + \phi_p \cdot \tilde{w}_{t-p} + a_t$$

або

$$(1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p) \cdot \tilde{w}_t = \phi(B) \cdot \tilde{w}_t = a_t,$$

де $\tilde{w}_t = w_t - \overline{w}$ - відхилення процесу від вибіркового середнього;

$\phi_1, \ldots, \phi_p$ - параметри авторегресії;

t - час (зазвичай, місяці);

p - порядок моделі;

a_t - послідовність однаково розподілених величин (білий шум);

B - оператор зсуву назад.

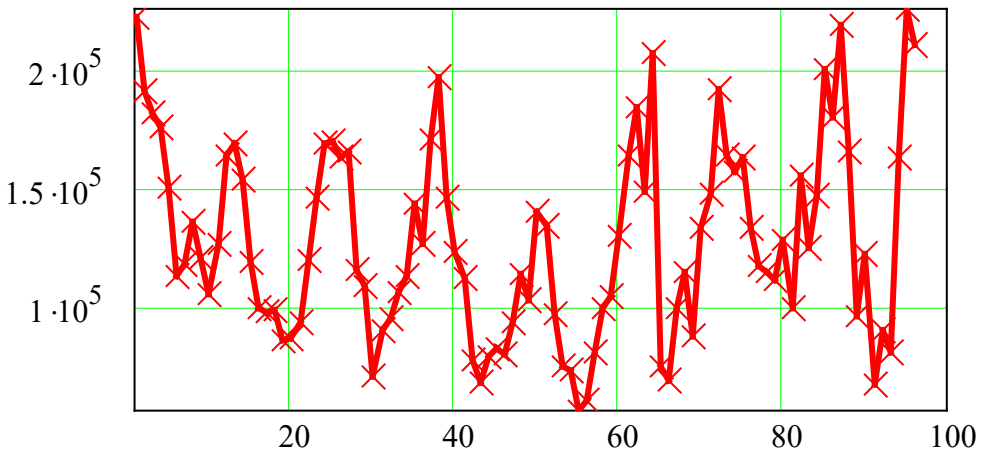


Рисунок 1.1 – Тимчасовий ряд одного з об'єктів

Важливими характеристиками авторегресійних моделей є вибіркові автоковараційні та автокореляційні функції [3,4]:

$$C_w(k) = \frac{1}{n}\sum_{1}^{n-k}(w_t - \overline{w})(w_{t+k} - \overline{w});$$

$$r_w(k) = \frac{C_w(k)}{s_w^2};\ k = 1,2,3,\ldots,n-1,$$

де $C_w(k)$ - вибіркова автоковарійна функція;

$r_w(k)$ - вибіркова автокореляційна функція;

$s_w^2 = \frac{1}{n}\sum_{1}^{n}(w_t - \overline{w})^2$ - вибіркова дисперсія часового ряду;

n - обсяг вибірки.

На рис. 1.2 представлено вибіркову автокореляційну функцію (АКФ) для тимчасового ряду одного з об'єктів техноценозу [35]. Вибіркова АКФ обчислюється з допомогою швидкого перетворення Фур'є, після чого потрібно ідентифікувати процес авторегресії, тобто. визначити порядок моделі, що розробляється. Це здійснюється за допомогою вибіркової автокореляційної функції (АКФ) моделі. Знання властивостей функції потрібно при вирішенні питання, чи є порядок предиктора k досить великим для адекватного опису структури часового ряду з можливістю використання коефіцієнтів $\phi_{k,j}$ для прогнозування Головне характеризує властивість авторегресійної моделі - це кінцівка її вибіркової АКФ [3,4], тобто:

$$\phi_{k,k} = 0, \text{при}\, k > p.$$

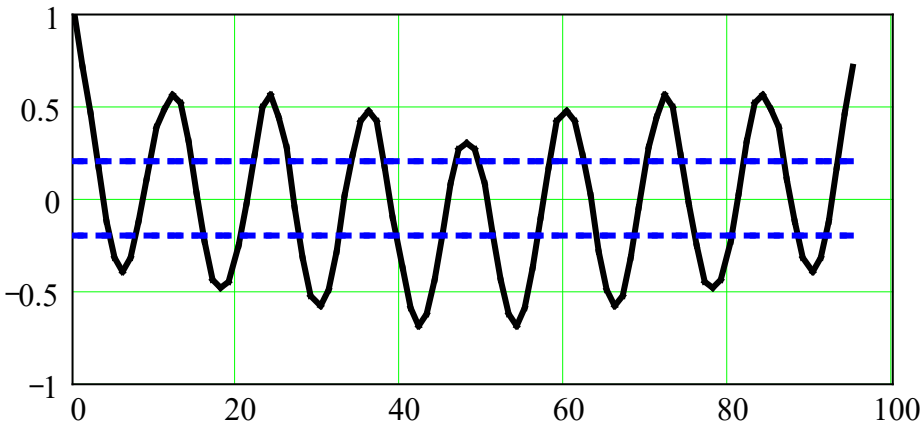


Рисунок 1.2 – Вибіркова АКФ для одного з об'єктів техноценозу:абсцис - відомі місяці передісторії; ордината – значення АКФ у відносних одиницях; штрихові лінії – межі довірчого інтервалу

Процедура обчислення коефіцієнтів кінцевого предиктора, що зупиняється при k = p, зводиться до розв'язання рівнянь Юла – Волкера для знаходження параметрів за даними вибіркових автокореляцій [3]:

$$r_j = \phi_{k1} \cdot r_{j-1} + \phi_{k2} \cdot r_{j-2} + \ldots + \phi_{k(k-1)} \cdot r_{j-k+1} + \phi_{kk} \cdot r_{j-k}$$

або в матричній формі $r_k = R_k \cdot \phi_k$, при j = 1, 2, …, k,

де ϕ_k - вектор-стовпець невідомих параметрів моделі;

r_k - вектор-стовпець вибіркових автокореляцій;

R_k - матриця вибіркових автокореляцій.

Вирішуючи систему послідовно для різних k отримаємо:

$$\phi_{11} = r_1;$$

$$\phi_{22} = \begin{pmatrix} 1 & r_1 \\ r_1 & r_2 \end{pmatrix} \Big/ \begin{pmatrix} 1 & r_1 \\ r_1 & 1 \end{pmatrix};\quad \phi_{33} = \begin{pmatrix} 1 & r_1 & r_1 \\ r_1 & 1 & r_2 \\ r_2 & r_1 & r_3 \end{pmatrix} \Big/ \begin{pmatrix} 1 & r_1 & r_2 \\ r_1 & 1 & r_1 \\ r_2 & r_1 & 1 \end{pmatrix};\ldots$$

На рис. 1.3 представлена вибіркова АКФ для часового ряду одного з об'єктів техноценозу. Аналіз малюнка показує, що вибіркова АКФ не дорівнює нулю на третій затримці, що свідчить про необхідність у разі вирішити систему рівнянь Юла – Уолкера для перевірки достатності моделі авторегресії. Входження вибіркової АКФ на четвертій затримці у межі довірчого інтервалу свідчить необхідність ідентифікувати процес з допомогою авторегресійної моделі третього порядку [2].

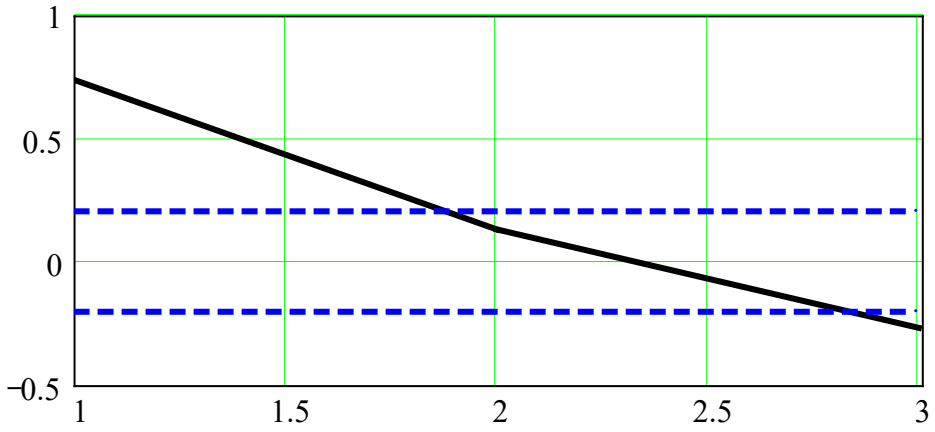


Рисунок 1.3 – Вибіркова АКФ для одного з об'єктів техноценозу:

абсцису – номери затримок АКФ; ордината – значення АКФ у відносних одиницях; штрихові лінії – межі довірчого інтервалу

Для остаточної оцінки параметрів моделі складається цільова функція, де білий шум представляється як функції значень ряду [36]:

$$F(\phi_1,\phi_2,\phi_3)=\sum_i(w_i-\phi_1\cdot w_{i-1}-\phi_2\cdot w_{i-2}-\phi_3\cdot w_{i-3})^2,$$

$$a_i=w_i-\phi_1\cdot w_{i-1}-\phi_2\cdot w_{i-2}-\phi_3\cdot w_{i-3}, i=1,\ldots,n.$$

Пошук мінімуму F здійснюється за допомогою ітераційних алгоритмів оптимізації з використанням методів квазин’ютонівського або сполучених градієнтів. Обидва методи засновані на ідеї заміни мінімізованої функції на

околиці чергової точки першими членами її розкладання в ряд Тейлора [3,4]. У градієнтному методі береться лінійна частина розкладання, методі Ньютона – квадратична.

Послідовність наближень $w^0, w^1, w^2, \ldots$ до точки мінімуму вибирається за правилом:

$$w^{k+1} = w^k + \alpha_k \cdot h^k,$$

де h^k - напрямок зменшення функції F у точці w^k;

α_k - параметр, що регулює довжину кроку вздовж h^k.

У градієнтному методі h^k дорівнює антиградієнту функції F у точці w^k [3,4]:

$$h^k = -F'(w^k).$$

Якщо довжина кроку в градієнтному методі вибирається з умови мінімізації функції вздовж напрямку антиградієнта, виходить метод якнайшвидшого спуску [44].

Метод сполучених градієнтів дає більш високу швидкість збіжності та реалізується у загальному вигляді наступним чином:

$$F(w^k + \alpha_k \cdot h^k) = \min_{\alpha \geq 0} F(w^k + \alpha_k \cdot h^k);\ w^{k+1} = w^k + \alpha_k \cdot h^k,$$

$$k = 0, 1, \ldots;$$

$$h^0 = -F'(w^0);\ h^k = -F'(w^k) + \beta_{k-1} \cdot h^{k-1},\ k \geq 1;$$

$$\beta_{k-1} = \begin{cases} \dfrac{\left\langle F'(w^k), F'(w^k) - F'(w^{k-1}) \right\rangle}{\left\| F'(w^{k-1}) \right\|^2}, & k \notin \{n, 2n, 3n, \ldots\}; \\ 0, & k \in \{n, 2n, 3n, \ldots\}. \end{cases}$$

де $\langle x, y \rangle = \sum_{k=1}^{n} x_k y_k$ - скалярний добуток елементів x і y;

$\|x\| = \sqrt{\langle x, x \rangle}$ - норма елемента x.

За першої ітерації α_k вибирається, як і методі якнайшвидшого спуску, а за наступних – вздовж напрямку h^k.

До методів оптимізації другого порядку належить квазин'ютонівський метод. Він дещо ефективніший за метод пов'язаних градієнтів, тому що в ньому результати обчислень сходяться швидше [3,4]. Квазин'ютонівський метод реалізується так [45]:

$$w^{k+1} = w^k + \alpha_k \cdot h^k, \; h^k = -H_k \cdot F'(w^k);$$

$$\Delta w^k = w^{k+1} - w^k, \; \Delta y^k = F'(w^{k+1}) - F'(w^k);$$

$$H_{k+1} = H_k + \frac{(\Delta w^k - H_k \cdot \Delta y^k) \cdot (\Delta w^k - H_k \cdot \Delta y^k)}{\left\langle \Delta w^k - H_k \cdot \Delta y^k, \Delta y^k \right\rangle}.$$

В якості H_0 часто застосовується поодинока матриця. Довжина кроку вибирається з умови мінімізації F вздовж заданого напрямку [29]:

$$F(w^k + \alpha_k \cdot h^k) = \min_{\alpha} F(w^k + \alpha_k \cdot h^k).$$

Недоліком методу є те, що його збіжність та результат обчислень залежить від початкового наближення. Результати, отримані за допомогою

двох перерахованих методів, розрізняються лише у восьмому знаку після коми, що збігається з результатами, отриманими [1-4].

Після ідентифікації моделі та знаходження остаточних оцінок її параметрів необхідно перевірити якість припасування моделі, тестуючи залишки на білий шум (рис. 1.4). Кумулятивний спектр залишків при цьому обчислюється за допомогою швидкого перетворення Фур'є.

У разі накопичений спектр залишків моделі лежить усередині довірчих кордонів, що підтверджує адекватність розробленої моделі і свідчить у тому, що залишки є білим шумом [5].

Оскільки розроблена модель адекватна, то з її допомогою можна здійснювати прогноз на рік наперед (рис. 1.4). Для нашого випадку (модель третього порядку) прогноз здійснюється за таким виразом:

$$wp_0 = w_0;\ wp_1 = w_1;\ wp_2 = w_2;$$

$$wp_q = \varphi_1 \cdot wp_{q-1} + \varphi_2 \cdot wp_{q-2} + \varphi_3 \cdot wp_{q-3} + a_q + \overline{w};$$

$$q = 3 \ldots (n + 12),$$

де q - індекси відомих значень часового ряду об'єкта плюс час запобігання (12 місяців).

Як показують розрахунки, відносна помилка для техноценозу загалом під час прогнозування з допомогою авторегресійних моделей може становити до 4%.

Модель декомпозиції часових рядів при аналізі процесу часових рядів у своїй основі містить такі складові: детерміновану (тренд), сезонну та залишкову випадкову. У загальному випадку, віднімання тренду призводить до моделювання стаціонарного процесу, що забезпечує підвищення стійкості та точності. Розглянуту модель можна уявити виразом виду [3,4]:

$$w_t = f(t) + s_t + e_t,$$

де $f(t)$ - детермінована складова процесу (тренд);

s_t - сезонна складова;

e_t - залишкова випадкова складова.

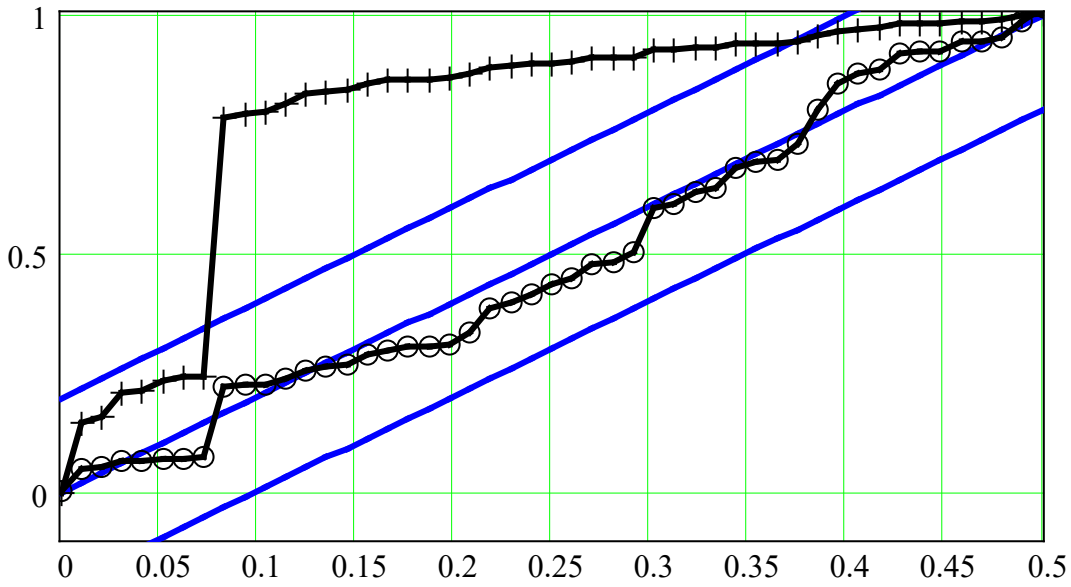


Рисунок 1.4 – Кумулятивний спектр залишків для одного з об'єктів:

1.3 Обмеження на вхідні і вихідні параметри, їх вплив на коректність рішення

Розглянемо загальну постановку задач адаптивного вибору варіантів, представлену на рис. 1.5.

Сенс підходу полягає в наступному–у кожен із послідовних моментів часу $t_n (n = \overline{1;N})$ необхідно вибирати варіант v_n з кінцевої множини можливих варіантів V [34].

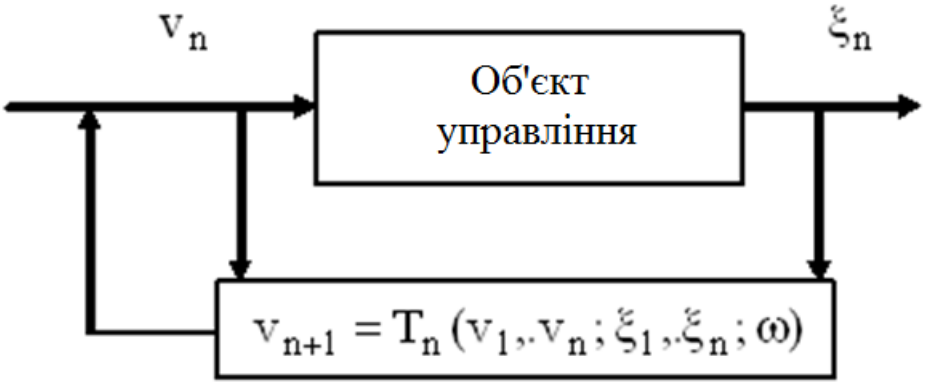


Рисунок 1.5 – Схема адаптивного вибору варіантів

Втрати системи ξ_n є функцією елементарного результату ω (має бінарні значення «штраф» та «відсутність штрафу») та залежать від обраного варіанта v_n, і навіть, можливо, стану системи. Реалізована при цьому послідовність варіантів $\{v_n\}$ має бути такою, щоб досягалася задана мета, що формулюється в термінах граничних значень поточних середніх втрат [33].

Більш проста реалізація детермінованих стратегій можлива за допомогою детермінованих кінцевих автоматів, які в основному орієнтовані на завдання з бінарними втратами, хоча можуть застосовуватись і в інших випадках. З іншого боку, їм характерно забезпечення прийнятного поведінки, близькість якого до оптимальному зростає зі збільшенням глибини пам'яті автомата. Однак це спричиняє зменшення швидкості досягнення мети та збільшує складність, а саме кількість станів, відповідного автомата. Це ж властиво та стохастичним автоматам із постійною структурою, які реалізують рандомізовані стратегії вибору. Складним рандомізованим стратегіям у теорії поведінки автоматів відповідають стохастичні автомати із змінною структурою.

1.4 Класифікація методів прогнозування

Якість моделі регресії пов'язують із відповідністю моделі спостережуваним (емпіричним) даним. Перевірка цієї відповідності проводиться через аналіз залишків (εі), які відображають відхилення фактичних значень залежної змінної від їх розрахункових (теоретичних)

значень. Під час створення рівняння регресії ми можемо розкласти значення в кожному спостереженні на дві компоненти. Залишок представляє собою відхилення фактичного значення залежної змінної від відповідного розрахункового значення, яке визначається розрахунковим методом. У випадку, коли εі=0, фактичні значення співпадають з розрахунковими для всіх спостережень. Графічно це означає, що теоретична лінія регресії (побудована за функцією y=a0+a1x) проходить через всі точки кореляційного поля, що можливо лише при наявності чіткого функціонального зв'язку. На практиці, зазвичай спостерігається певний розсіювання точок кореляційного поля відносно теоретичної лінії регресії, що свідчить про відхилення емпіричних даних від теоретичних. ($\varepsilon_i \neq 0$) [55].

Величина цих відхилень і лежить в основі розрахунку показників якості (адекватності) рівняння.

Для оцінки якості регресійних моделей використовують також к-т множинної кореляції:

$$R = \sqrt{\frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2}}$$

Цей коефіцієнт є універсальним, оскільки він відображає тісноту зв'язку і точність моделі, а також може використовуватися за будь-якої форми зв'язку змінних. Коефіцієнт множинної кореляції, зведений у квадрат, називається к-том детермінації [49]:

$$R^2 = 1 - \frac{\sum e(i)^2}{\sum (y_i - \bar{y}_i)^2} = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2}$$

Він показує частку варіації результативної ознаки, що перебуває під впливом чинників, які вивчаються, тобто визначає, яка частка варіації ознаки Y врахована в моделі й зумовлена впливом на неї чинників. Чим ближче він до 1, тим вища якість моделі [36].

Для перевірки значущості моделі регресії використовується F-критерій Фішера

$$F = \frac{r_{Y,X}^2}{1 - r_{Y,X}^2} \times (n-2)$$

Якщо розрахункове значення з t1 = k і t2 = (n-k-1) ступенями свободи, де k - кількість чинників, включених до моделі, більше від табличного за заданого рівня значущості, то модель вважається значущою.

Побудова довірчого інтервалу для точкового прогнозу за лінійною моделлю.

Регресійні моделі можуть бути використані для прогнозування можливих очікуваних значень залежної змінної. Прогнозоване значення змінної у отримують під час підстановки в рівняння регресії очікуваної величини фактора х. Цей прогноз називається точковим. Значення незалежної змінної $x_{прогн}$ не повинно значно відрізнятися від тих, що входять до досліджуваної вибірки, за якою обчислено рівняння регресії. Імовірність точкового прогнозу теоретично дорівнює 0. Тому розраховується середня помилка прогнозу або довірчий інтервал прогнозу з досить великою надійністю. Довірчі інтервали залежать від стандартної помилки, віддаленості $x_{прогнозу}$ від свого середнього значення, кількості спостережень і рівня значущості прогнозу. Визначимо довірчий інтервал прогнозу:

$$u = S_{\bar{y}} \times t_{\alpha} \times \sqrt{1 + \frac{1}{n} + \frac{(x_{ïðîãí} - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

Величину відхилення від лінії регресії обчислюють за формулою:

$$S_{\bar{y}} = \sqrt{\frac{\sum_{i=1}^{n} E_i^2}{n-k-1}}$$

Термін «метод прогнозування» дуже широко використовується від простих розрахунків до багатоетапних експертних опитувань, а також як метод теоретичних та практичних дій.

Надійність та точність прогнозу визначаються безпосередньо використанням методів та розрахункових моделей. В сучасній добі існує значна кількість різноманітних методів та моделей для прогнозування даних. Але лише 15-20 методів знаходять практичне застосування як основні засоби прогнозування. Для короткострокових та довгострокових прогнозів використовуються різні методи прогнозування, орієнтовані на різні завдання та використовують спеціалізовані бази даних [47].

Сучасні джерела літератури подають різні класифікації методів прогнозування. При складанні класифікації методів прогнозування важливо враховувати, що систематизація повинна бути обумовлена природою об'єкта прогнозування, його закономірностями та тенденціями розвитку.

Програмна сторона проблеми прогнозування включає в себе визначення часу реалізації кожного варіанту та ступеня їх точності, а також пошук різних способів отримання необхідних результатів. Це аспект визначається наявністю фінансових ресурсів та наукового потенціалу. На даному етапі обґрунтовується необхідність витрат економічних ресурсів на проведення прогнозної роботи, формулюється гіпотеза, та дається ймовірнісна оцінка розподілу можливостей та перспектив розвитку наукового потенціалу.

Загальні методи прогнозування можна класифікувати в чотири великі групи:

- методи нейронних мереж;
- методи екстраполяції;
- методи експертних оцінок;
- методи регресійного аналізу.

У табл. 1.1 наведено загальні показники методів прогнозування [4].

Таблиця 1.1 Загальні параметри методів прогнозування

Методи	Характеристика	Різновиди
Нейронні мережі	Метод заснований на сукупності нейронів та їх зв'язків. Мають можливість навчання з урахуванням інформації про довкілля.	

Методи	Характеристика	Різновиди
	Переважно використовується у короткостроковому прогнозі.	
Екстраполяція	Ґрунтується на статистичних даних певного параметра, тенденції його зміни. Застосовується за мінімальної кількості змін. До кожного об'єкта створюється окремий прогноз.	Метод експонентного згладжування. Метод ковзного середнього.
Експертні оцінки	Прогноз проводиться на основі суб'єктивних знань експерта та його інтуїції. Застосовується для короткострокового, середньострокового та довгострокового прогнозування	Колективні та індивідуальні експертні оцінки.
Регресійний аналіз	Досліджує кореляційну залежність параметрів. Переважно використовується у середньостроковому прогнозі	

При прогнозуванні деяких показників слід враховувати наявність стійкого взаємозв'язку між об'єктом, що вивчається, і його значеннями в минулому і майбутньому [41].

Щоб вибрати оптимальний спосіб побудови прогнозної моделі часових рядів, необхідно знати інтервал та точність прогнозування, мету прогнозу, адаптивність моделі та її швидкість [9].

1.5 Регресійні моделі

При математичній обробці масивів експериментальної інформації виникає необхідність у підборі емпіричних формул, що встановлюють зв'язок одного виміряного параметра з іншим.

Завдання визначення точного виду виявленої взаємозалежності параметрів вирішується за допомогою регресійного аналізу [25].

Регресійний аналіз полягає у визначенні аналітичного виразу зв'язку, в якому зміна одного параметра (y) зумовлена впливом іншого параметра (x).Кількісна оцінка цього взаємозв'язку здійснюється з допомогою побудови регресійної функції - рівняння регресії.

У загальному випадку рівняння регресії залежного параметра, від незалежного параметра x можна записати у вигляді аналітичного полінома ступеня n:

$$y = a + bx + cx2 + ... + fxn.$$

У найпростішому випадку між двома корельованими параметрами існує лінійний зв'язок, для якого вираз можна переписати в такому вигляді:

$$y = a + \beta x.$$

Вираз є лінійним рівнянням регресії, в якому величина називається вільним членом рівняння регресії, а величина β - коефіцієнтом рівняння регресії.

Припустимо, що в результаті вимірювань сформовані масиви експериментальної інформації за параметрами x та y, які пов'язані залежністю виду y = f(x), а графік цієї залежності представлений на рис. 1.6.

З курсу вищої математики відомо, що через будь-які точки завжди можна провести криву, що виражається аналітичним поліномом ступеня (n-1), так щоб вона в точності пройшла через кожну з точок (рис. 1.6, безперервна крива). Проте вигляд такої кривої вкрай складний її математичного описи. Виникає завдання згладжування експериментальної залежності. Експериментальні дані бажано обробити так, щоб по можливості досить точно відобразити загальну тенденцію залежності від x, але разом з тим «згладити» нехарактерні, випадкові відхилення, викликані, в тому числі, і неминучими похибками вимірювань, докладно представленими в попередньому розділі

навчального посібника. Одним із ефективних методів розрахункового згладжування є метод найменших квадратів [28].

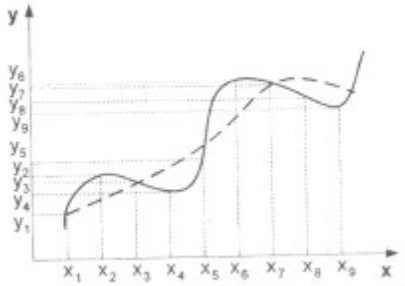


Рисунок 1.6 – Вид експериментальної залежності між параметрами

Нехай маємо дані експериментальних вимірювань за параметрами хі та уі. Розглянемо випадок лінійної регресії. Використовуючи метод найменших квадратів, можна записати:

$$\begin{cases} \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} (\alpha + \beta x_i) = 0; \\ \sum_{i=1}^{n} y_i x_i - \sum_{i=1}^{n} (\alpha + \beta x_i) x_i = 0; \end{cases}$$

Розкриємо дужки в цих рівняннях і, здійснивши підсумовування, отримаємо:

$$\begin{cases} n\alpha + \beta \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i ; \\ \alpha \sum_{i=1}^{n} x_i - \beta \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i x_i ; \end{cases}$$

Виразимо з другого рівняння системи коефіцієнт рівняння регресії:

$$\beta = \frac{n \sum_{i=1}^{n} y_i x_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2}$$

Вільний член рівняння регресії і можна знайти із системи рівнянь. Але простіше величину α виразити через β:

$$\alpha = \frac{\sum_{i=1}^{n} y_i - \beta \sum_{i=1}^{n} x_i}{n}$$

Вираз можна переписати у вигляді:

$$\alpha = \frac{1}{n}\sum_{i=1}^{n} y_i - \beta \sum_{i=1}^{n} x_i = M(y) - \beta M(x)$$

Величина α, обчислена за формулою, дозволяє відобразити найбільш характерну криву з урахуванням поведінки параметрів хі та уі та математичних очікувань M(x) та M(y). Розмір β характеризує кутовий коефіцієнт (кут нахилу до осі Ox) [52].

На жаль, такий підхід для розрахунку лінійного рівняння регресії дуже трудомісткий через складність розрахунків αіβ.

Розрахунки значно спрощуються, якщо використати коефіцієнт кореляції r. Знак при r показує характер тенденції кореляційного зв'язку та є одним із критеріїв правильності виконаних розрахунків:

- знак «+» означає, що зміна досліджуваних параметрів x та y має однакову тенденцію (кореляційний зв'язок позитивний);
- знак «-» означає, що зміна досліджуваних параметрів x і має різну тенденцію (кореляційний зв'язок негативний);
- значення r = 0 означає, що кореляційний зв'язок між параметрами x та y відсутня.

Для статистичного визначення коефіцієнта кореляції між двома випадковими величинами у них необхідно мати дані їх вимірювань [39]. Нехай спостерігалися наступні пари одночасних вимірів величин y та x: $\{y_1, x_1\}; \{y_2, x_2\}; \dots \{y_n, x_n\}$. Тоді для отримання залежності $y = f(x)$ потрібно знайти α, β, σx, σy,і r за такими формулами:

$$\begin{cases} \alpha = \frac{\sum_{i=1}^{n} x_i}{n}; \beta = \frac{\sum_{i=1}^{n} y_i}{n} \\ \sigma_x^2 = \sum_{i=1}^{n} (x_i - \alpha)^2/(n-1); \\ \sigma_y^2 = \sum_{i=1}^{n} (y_i - \beta)^2/(n-1); \\ r = \sum_{i=1}^{n} (x_i - \alpha)^2 (y_i - \beta)^2/[(n-1)\sigma_x \sigma_y] \end{cases} \quad (5.8)$$

Важливою перевіркою складання регресійної моделі є знак коефіцієнта р правої частини рівняння [11]:

- знак «+» означає, що зміна досліджуваних параметрів х і має однакову тенденцію;
- знак "-" - різну тенденцію.

1.5.1 Просте ковзне середнє

Термін ковзне середнє означає, що набір значень, усереднюють, безперервно рухається в часі. Ковзне середнє відображає тенденцію зміни цін і згладжує їх несуттєві коливання. Оскільки ковзне середнє є середнім значенням цін у минулому, графіки ковзних середніх "відстають" від поточних змін у часовому ряду. На ринках із чітко вираженим трендом ковзні середні показують добрий результат, проте на ринках, де немає чітко вираженої тенденції, метод дає великі похибки. Формально метод описується так [23]:

$$\hat{y_t} = \frac{1}{N} \sum_{i=1}^{N} y_{k-i},$$

де N число попередніх моментів часу було врахувати при побудові прогнозу,

yk-i реальні значення показника на момент часу tk-i.

1.5.2 Зважене ковзне середнє

Метод зваженого ковзного середнього розширює ідею простого ковзного середнього, оскільки кожен компонент часового ряду врівноважується відповідними ваговими коефіцієнтами, тобто:

$$\hat{y}_t = \frac{1}{N} \sum_{i=1}^{N} w_i y_{k-i},$$

де N число попередніх моментів часу було врахувати при побудові прогнозу,

yk-i - реальні значення показника в момент часу tk-i,

wi - ваговий коефіцієнт для і того компонента ряду.

Перевагою зважених коефіцієнтів є те, що в результаті отримано оцінку тренду набагато прасування. Замість того, щоб кожне вхідне спостереження різко змінювало значення середнього значення спостережень можуть бути зважені за допомогою коефіцієнтів [10].

1.5.3 Експоненційний ковзне середнє

Експоненційна ковзна середня (Exponential Moving Average EMA) походить з ідеї зваженого ковзного середнього, але в порівнянні з простим ковзним середнім, більший акцент робиться на останні точки даних і результуючі усереднені значення ближче до фактичних спостережень за набором даних. Вагові коефіцієнти зменшуються експоненційно, внаслідок чого акцент падає останні спостереження, хоча не відкидаються старі. Також передбачається, що в цьому процесі не існує систематичних трендів чи сезонних коливань, або що їх було визначено та видалено [20].

Процес експоненційно зваженого ковзного середнього заданого часового ряду визначається як:

$$\hat{y}_t = \alpha y_t + (1 - \alpha) y_{t-1},$$

де зменшення зважування, постійний коефіцієнт згладжування від 0 до 1,

yt – це значення у період часу t,

$\hat{y}_t$ - Значення EMA прогнозу в будь-який період часу t.

Використовуючи формулу експоненційно зваженого середнього до всього ряду, можна записати наступним чином:

$$\hat{y}_t = \alpha(y_t + (1 - \alpha) y_{t-1} + (1 - \alpha)^2 y_{t-2} + \ldots + (1 - \alpha)^k y_{t-k}) + (1 - \alpha)^{k+1} y_{t-(k+1)}$$

Для всіх:

$$k \in \{0, 1, 2, ..\}$$

Вага кожного спостереження yt і визначається як (1-α)і. З часом зваженим середнє оцінює дедалі більше спостережень, а відповідні ваги зменшуються у геометричній прогресії.

В експоненційному згладжуванні, однак, є один або більше параметрів згладжування, які потрібно визначити, і цей вибір впливає на вибір ваги для зважування спостережень.

Коефіцієнт приймає значення від 0 до 1. Якщо близько до 1, то згладжування мало, і yt приблизно дорівнює $\hat{y}_t$. Це прийнятно у разі, якщо очікуються дуже великі зміни у значенні середнього. У разі коли значення α близько до 0, прогноз дає дуже згладжені оцінки середнього значення, і не враховує останні спостереження.

1.5.4 Подвійне експоненційне ковзне середнє

Метод простого експоненційного згладжування не спрацьовує добре, коли в даних є чітко виражений тренд, наприклад ціна валюти, що різко зростає. У таких ситуаціях було розроблено кілька методів під назвою подвійне експоненційне згладжування або експоненційне згладжування другого порядку, є рекурсивним застосуванням експоненційного фільтра двічі. Основна ідея подвійного експоненційного згладжування полягає в тому, щоб ввести термін, що враховує можливість мати ряд тренд. Кожен набір даних часових рядів може бути розкладений на його складові, які є трендом trt, сезонною компонентою st та нерегулярними коливаннями rt.

Для того, щоб висловити це в математичній нотації, тепер потрібні три рівняння:

$$a_t = \alpha y_t + (1 - \alpha)(a_{t-1} + b_{t-1})$$

$$b_t = \beta(a_t - a_{t-1}) + (1 - \beta)b_{t-1}$$

$$\hat{y}_{t+h} = a_t + hb_t ,$$

де 0 <α <1 коефіцієнт згладжування даних,

0 <β <1 коефіцієнт згладжування тренду.

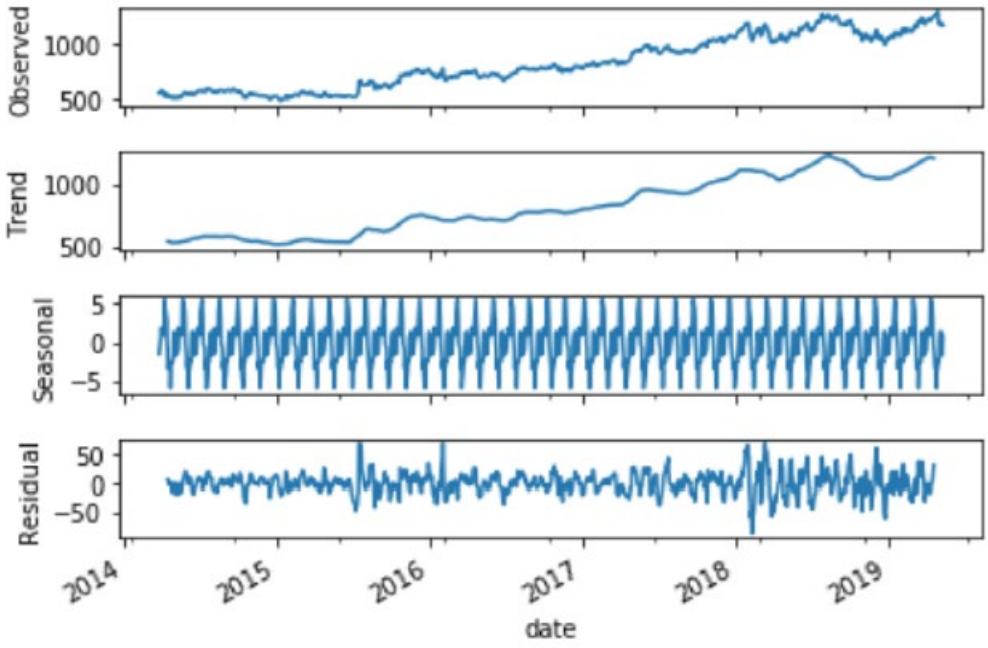


Рисунок 1.7 – Декомпозиція вихідного часового ряду

Потрійне експонентне згладжування, також відоме як метод Голта-Вінтерса, застосовує експоненційне згладжування три рази. Модель використовується для даних, у яких є як тренд, і сезонність.

Вона задається трьома параметрами, що згладжують модель, і має додаткове рівняння для коригування сезонної компоненти.

$$a_t = \alpha(y_t - s_{t-L}) + (1-\alpha)(a_{t-1} + b_{t-1})$$

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$

$$s_t = \gamma(y_t - a_t) + (1-\gamma)s_{t-L}$$

$$\widehat{y_{t+h}} = a_t + hb_t + s_{t+1+(h-1) \bmod L},$$

де 0 <α <1 - коефіцієнт згладжування даних,

0 <β <1 - коефіцієнт згладжування тренду,

0 <γ <1 - коефіцієнт згладжування сезонної компоненти,

L - довжина сезонного циклу,

h кількість кроків прогнозування.

Завдання прогнозування часових рядів – складний тип проблеми прогнозуючого моделювання. На відміну від регресійного передбачуваного моделювання, тимчасові ряди також додають складність залежності послідовності від вхідних змінних. Потужний тип нейронної мережі, призначений для обробки послідовностей, називається рекурентними нейронними мережами.

Довгою короткою пам'яттю або мережа LSTM - це тип рекурентної нейронної мережі, що використовується в глибокому навчанні, тому що можна успішно навчати дуже великі архітектури.

Завдання, яке ми розглянемо - проблема прогнозування пасажирських авіаперевезень. Завдання полягає в тому знаючи рік і місяць передбачити кількість пасажирів міжнародних авіакомпаній. Набір даних доступний безкоштовно можна завантажити з адреси https://datamarket.com/data/set/22u3/international-airline-passengers-monthly-totals-in-thousands-jan-49-dec-60#!ds=22u3&display=line з ім'ям файлу "international-airlines-passengers.csv". Дані варіюються від січня 1949 року до грудня 1960 року або 12 років із 144 спостереженнями. Ми можемо завантажити цей набір даних за допомогою бібліотеки Pandas. Нам не цікава дата з огляду на те, що кожне спостереження поділяється одним і тим же інтервалом в один місяць. Тому, коли завантажуємо набір даних, ми можемо виключити перший стовпець. Після завантаження ми можемо легко збудувати весь набір даних [59].

```
import pandas
import matplotlib.pyplot as plt
dataset = pandas.read_csv('international-airline-passengers.csv', usecols=[1], engine='python', skipfooter=3)
plt.plot(dataset)
```

plt.show() З часом можна побачити висхідний тренд у наборі даних та деяку періодичність для набору даних, який, ймовірно, відповідає періоду відпустки у північній півкулі.

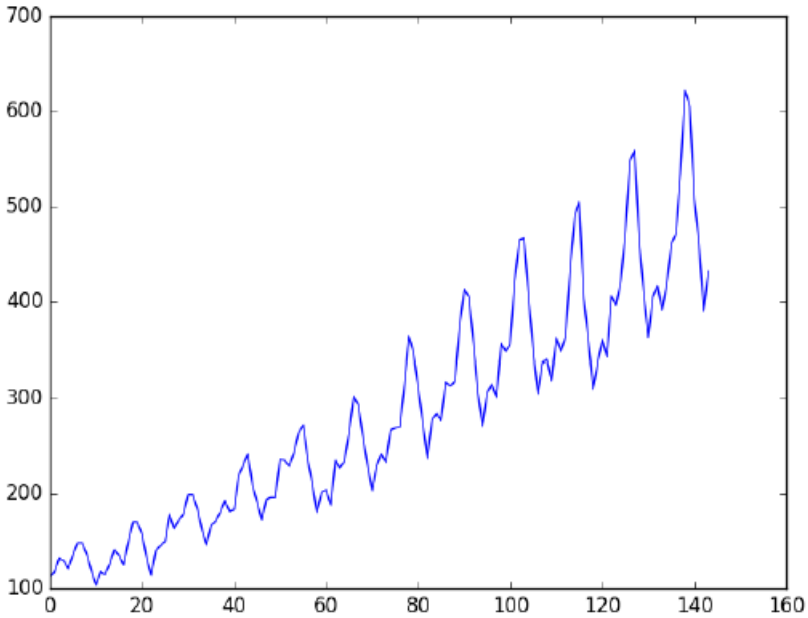


Рисунок 1.8 – Побудова діаграми даних

Ми можемо сформулювати це завдання як завдання регресії. Тобто з огляду на кількість пасажирів (у тисячах одиниць) цього місяця прогнозувати кількість пасажирів наступного місяця. Ми можемо написати просту функцію, щоб перетворити наш єдиний стовпець даних на двостовпцевий набір даних: перша колонка, що містить кількість пасажирів і другий стовпець, який міститиме кількість пасажирів наступного місяця.

LSTM чутливі до шкали вхідних даних, особливо коли використовуються сигмоїдні (за замовчуванням) або функції активації tanh. Тому необхідно провести масштабування даних до діапазону від 0 до 1, який також називається нормалізацією. Ми можемо легко нормалізувати набір даних, використовуючи клас попередньої обробки MinMaxScaler із бібліотеки scikit-learn [60].

scaler = MinMaxScaler(feature_range=(0, 1))

dataset = scaler.fit_transform(dataset) Після того, як ми моделюємо наші дані та оцінимо якість нашої моделі на навчальному наборі даних, нам потрібно дізнатися наскільки точний прогноз на даних, які мережа не бачила. Для

звичайної задачі класифікації чи регресії ми робитимемо це з використанням перехресної перевірки. При використанні часових рядів важлива послідовність значень. Простим методом, який ми можемо використовувати, є поділ впорядкованого набору даних на навчальний та тестовий набор даних. Нижче наведений код поділяє дані на навчальні набори даних з 67% спостережень, які ми можемо використовувати для навчання нашої моделі, залишаючи 33% для тестування моделі. train_size = int(len(dataset) * 0.67) test_size = len(dataset) - train_size train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:] Тепер визначимо функцію створення нового набору даних, як описано вище [65]. Функція приймає два аргументи: набір даних, який являє собою масив NumPy, який ми хочемо перетворити на набір даних, і look_back, який є числом попередніх кроків часу для використання в якості вхідних змінних для прогнозування наступного періоду часу - в цьому випадку за замовчуванням - 1. Це значення за промовчанням створить набір даних, де X - кількість пасажирів у заданий час (t), а Y - кількість пасажирів наступного часу (t + 1) [56].

```
def create_dataset(dataset, look_back=1):
dataX, dataY = [], []
for i in range(len(dataset)-look_back-1):
a = dataset[i:(i+look_back), 0]
dataX.append(a)
dataY.append(dataset[i + look_back, 0])
```

return numpy.array(dataX), numpy.array(dataY) Далі використовуємо цю функцію для підготовки наборів даних для навчання та для тестування нейронної мережі та перетворюємо дані у структуру, що відповідає входу нейронної мережі [63].

```
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
```

testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

Через те, як було підготовлено набір даних, ми повинні зрушити передбачення так, щоб вони вирівнялися по осі x з вихідним набором даних.

```
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, :] = trainPredict
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset)-1, :] = testPredict
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```

Вихідний набір даних відображається синім кольором, прогнози для набору навчальних даних зеленим кольором та прогнози тестового набору даних червоним кольором. Ми, що модель чудово впоралася з прогнозом як навчальному, і на тестовому наборі даних.

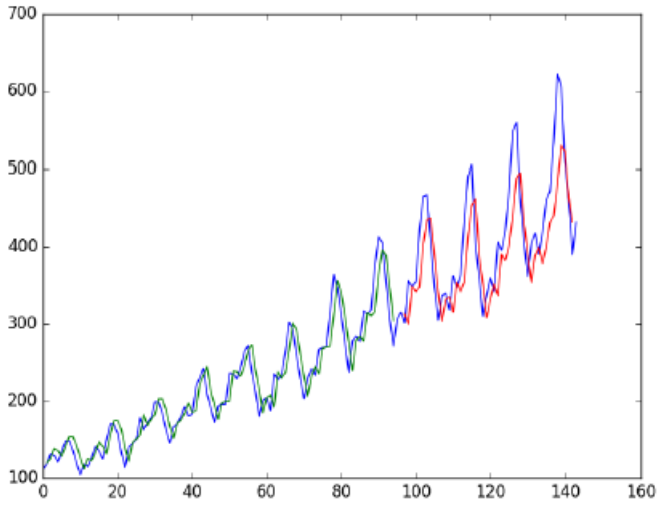


Рисунок 1.9 – Побудова діаграми даних та результату прогнозу

Висновки до розділу 1

В результаті виконання розділу було розглянуто класифікацію прогнозів при виконанні аналізу часових рядів, класифікацію методів прогнозування,

регресійні моделі. Завдання визначення точного виду виявленої взаємозалежності параметрів вирішується за допомогою регресійного аналізу.

Регресійний аналіз полягає у визначенні аналітичного виразу зв'язку, в якому зміна одного параметра (y) зумовлена впливом іншого параметра (x). Кількісна оцінка цього взаємозв'язку здійснюється з допомогою побудови регресійної функції - рівняння регресії [62].

Розділ 2 Алгоритмізація виконання аналізу часових рядів

2.1 Дослідження залежності коефіцієнтів початкової регресії від відносної ширини лінії вхідних реалізацій

При моделюванні реальних об'єктів дуже часто виявляється, що порушена передумова класичного регресійного аналізу, згідно з якою

$$\mathbf{V(\varepsilon)=\sigma^2 I.}$$

Таке порушення зазвичай проявляється у наступному. Може виявитися, що обурення**ε**має неоднорідні дисперсії в різних дослідах і його коваріаційна матриця (а значить, і матриця вектора у при невипадковій матриці плану) буде хоч і діагональної, але з різними елементами по діагоналі. І тут кажуть, що спостереження неоднорідні чи є неоднорідність.

Неоднорідність може виникнути природно, коли вона обумовлена природою явища, що вивчається. Характерним прикладом такого роду є об'єкти, про які з фізичних міркувань відомо, що дисперсія $\sigma^2(y)$ відгуку y залежить від його математичного очікування $E(y)$. Зрозуміло, що при змінах факторів у різних дослідах змінюватиметься і $\sigma^2(y)$. Такі властивості має розподіл Пуассона, а також біноміальний розподіл. Їхні дисперсії відповідно рівні

$$E(y) \qquad \text{і } E(y)\left(1-E(y)\right)/N,$$

де N – обсяг вибірки.

Часто природа досліджуваного явища підказує, що дисперсія відгуку зростає зі зростанням певного чинника. Подібні випадки трапляються в економіці, біології, хімічній кінетиці.

Є підстави вважати, що випадкове обурення іноді обумовлено головним чином помилкою виміру відгуку. Тоді цілком імовірно, що більшим значенням будуть відповідати й великі обурення, отже дисперсіїε- Неоднорідні.

Перерахуємо деякі випадки, коли неоднорідність, що виникла, введена «ззовні». Така неоднорідність виникає через лінеаризації деяких моделей. Так,

при дослідженні кінетики реакцій рівняння для швидкостей часто-густо лінеаризуються за допомогою звернення. При цьому неоднорідність, що вводиться, може виявитися дуже серйозною. Подібне явище спостерігається і коли переходять від поширених у хімічному експерименті нелінійних моделей з адитивною помилкою ε_0

$$y = \alpha e^{\beta x} + \varepsilon_0 = \alpha e^{\beta x}\left(1 + \varepsilon_0 / E\left(y\right)\right)$$

до лінійних моделей виду

$$\ln y = \ln \alpha + \beta x + \ln\left(1 + \varepsilon_0 / E\left(y\right)\right),$$

де

$$E\left(y\right) = \alpha e^{\beta x}.$$

Обурюючий вплив лінеаризованої моделі

$$\varepsilon = \ln\left(1 + \varepsilon_0 / E\left(y\right)\right)$$

і, отже, її дисперсія змінюється залежно від зміни $E\left(y\right)$ у планах.

Неоднорідність може виникнути також через неправильний вибір структури регресійної моделі. Якщо фактор x_i входить до неї лінійно, а в істинній моделі присутній і член x_i^2 У випадковому обуренні моделі з'явиться добавка, яка залежить від x_i^2. Тому дисперсія $\sigma^2\left(\varepsilon\right)$ буде змінюватися від досвіду до досвіду під впливом x_i^2 [50].

Неоднорідність виникає й за порушення передумови регресійного аналізу, що стосується невипадкового характеру матриці плану, як у ході експерименту задані рівні чинників встановлюються помилками. У цьому випадку для нелінійної за факторами моделі у всіх дослідах дисперсія у збільшується на деяку складову, що залежить не тільки від коефіцієнтів моделі та моментів помилок факторів, а й від заданих рівнів факторів. Тому й тут $\sigma^2\left(y\right)$ змінюється від досвіду до досвіду.

Експонентний відеоімпульс описується наступною математичною моделлю:

$$s(t) = \exp(-\alpha t) \cdot \sigma(t)$$

Даний імпульс у частотній області представлений своїм енергетичним спектром виду:

$$W_s(\omega) = \frac{1}{\left(\alpha^2 + \omega^2\right)}$$

та нормою

$$\|s\| = \sqrt{\frac{1}{\pi}\int_0^{\infty} \frac{d\omega}{\left(\alpha^2 + \omega^2\right)}} = \frac{0{,}7071}{\sqrt{\alpha}}$$

Ефективна тривалість експоненційного імпульсу дорівнює:

$$\tau_{\text{и}} = 2{,}303/\alpha\,.$$

Спектр сигналу, що розглядається, необмежений, тому слід попередньо піддати сигнал низькочастотної фільтрації, пропустивши його через фільтр нижніх частот (ФНЧ). Значення верхньої частоти ωу смуги пропускання фільтра слід вибирати в залежності від того, як часто беруться відліки сигналу на виході ФНЧ. Припустимо, що за час τ вимірюються k=10 відліків, тоді інтервал дискретизації

складе $t_0 = \tau_{\text{и}}/(k-1) = \tau_{\text{и}}/9 = 0{,}2558/\alpha =$

= 0,2558/2000 = 0,0001279 секунд (рис 2.1).

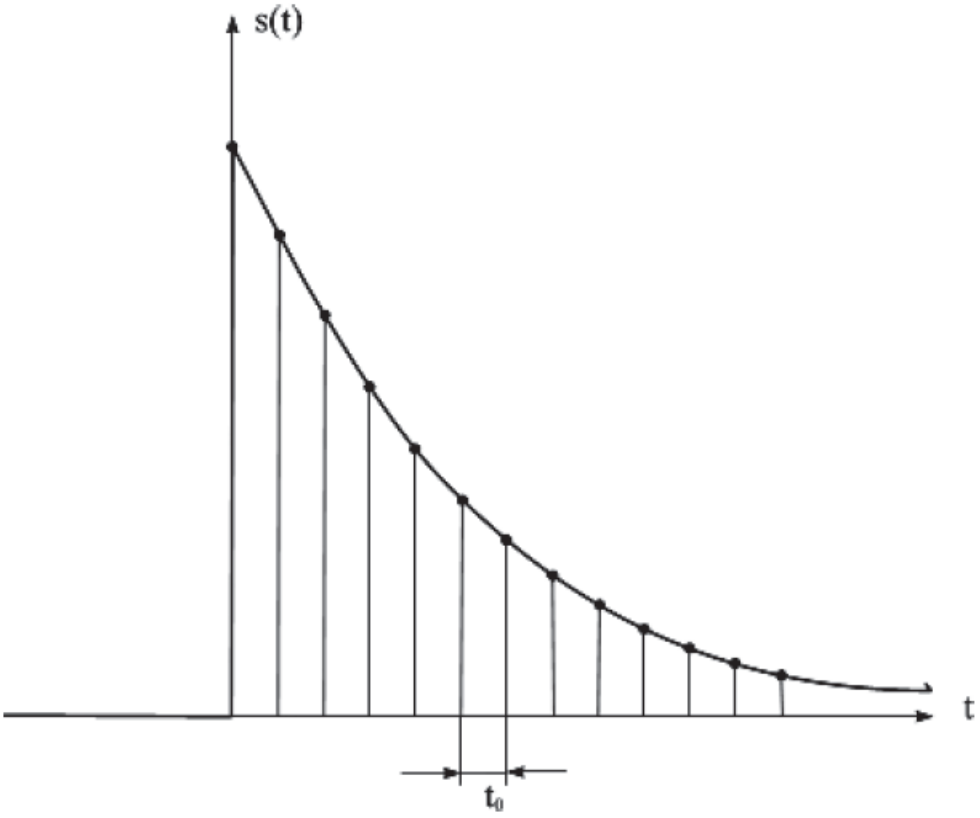


Рисунок 2.1 – Дискретизація експоненційного імпульсу

2.2 Моделі представлення знань

Модель авторегресії є ефективним інструментом для розуміння та прогнозування майбутніх значень тимчасового ряду, яка включає регресування змінної за значеннями ряду в минулому. Важливість моделей ARMA полягає в їхній гнучкості, а також у їхній здатності описувати майже всі особливості стаціонарних часових рядів. Авторегресивні частини цих моделей описують, як послідовні спостереження в часі впливають один на одного, тоді як частини ковзних середніх захоплюють деякі можливі потрясіння, що не спостерігаються, дозволяє моделювати різні явища, які можна спостерігати в різних галузях від біології до фінансів [40].

Проста модель авторегресії (AR Autoregression).

Модель авторегресії показує, що значення вихідного змінна залежить лінійно лише від попередніх значень і від випадкового шуму.

Позначення AR(p) позначає авторегресійну модель порядку p. AR(p) модель математично визначається як:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + e_n ,$$

де ai – параметри моделі, en – білий шум.

Найпростішим процесом AR є AR(0), який не має залежності між компонентами. На значення прогнозу впливає лише шум чи похибка. Для AR (1) з коефіцієнтом a1 тільки попереднє значення часового ряду впливає на значення прогнозу.

Якщо a1 близький до 0, процес буде виглядати як білий шум, але якщо a1 по модулю буде наближатися до 1, то вихідне значення отримуватиме більший внесок від попередніх значень ряду в порівнянні з шумом.

Модель авторегресії - ковзного середнього (ARMA Autoregressive moving average).

Модель ARMA характеризує стохастичний процес за допомогою двох компонентів автомобільної регресії (AR) і ковзного середнього (MA).

Частина AR передбачає регресування змінної на власні попередні значення. Частина MA включає моделювання похибки як лінійної комбінації помилок, які у минулому.

Позначення ARMA (p, q) характеризує модель з p авто регресійними компонентами і q компонентами для ковзного середнього:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + \sum_{i=1}^{q} b_i e_{n-i} + e_n ,$$ (

де ai, bi – параметри моделі, en – білий шум.

Модель ARMA доречна, коли система є функцією ряду потрясінь, що не спостерігаються, і власної поведінки. Наприклад, ціни на акції можуть бути суттєво змінитися через якусь фундаментальну інформацію, а також

продемонструвати ефект повернення до середнього через певні дії учасників ринку.

Модель авторегресії - інтегрованого ковзного середнього (ARIMA Autoregressive integrated moving average)

Модель ARIMA є ускладненням моделі ARMA. Вона складається з трьох компонентів:

– авторегресії (AR), що використовує залежний зв'язок між спостереженням та іншими спостереженнями із затримкою в часі;

– інтегрована компонента (I), що диференціює вхідні спостереження з метою зробити тимчасовий ряд стаціонарним;

– ковзного середнього (MA), що використовує залежність між спостереженням та відхиленням від ковзного середнього попередніх спостережень.

Кожна з цих компонентів визначає параметри моделі. Використовується стандартне позначення ARIMA (p, d, q), де параметри позначають:

– p - кількість лагових спостережень, включених у модель;

– d - кількість разів, коли до вхідного спостереження було застосовано диференціювання. Параметр також називається ступенем диференціювання;

– q - розмір вікна ковзного середнього, або порядок ковзного середнього.

Для застосування моделі ARIMA використовуються формули моделі ARMA, проте на вхід замість yt подається $\Delta yt = yt - yy-1$.

Для кожного параметра можна використовувати значення 0, що вказує на відсутність відповідного компонента моделі. Тобто модель ARIMA може бути налаштована для виконання функцій моделі ARMA або навіть простий AR, I або MA [17].

Слід також зауважити, що AR і MA є лінійними моделями, що працюють на стаціонарних часових рядах, в той час як модель I є процедурою

попередньої обробки даних, щоб звести ряд до стаціонарного, якщо це необхідно.

Розглянемо 5 методів згладжування, кожен із яких був реалізований для кількох параметрів. Кожен наступний метод є ускладненням попереднього, отже очевидно дає більш точні результати.

Метод ковзного середнього було реалізовано для різних значень параметра N кількості попередніх моментів часу, було врахувати при обудові прогнозу (рис. 2.2). Було перевірено, що при зменшенні довжини вікна N модель показує точніший результат на тестовій вибірці, вказує на властивість останніх даних впливати на прогнозну оцінку в майбутньому [31].

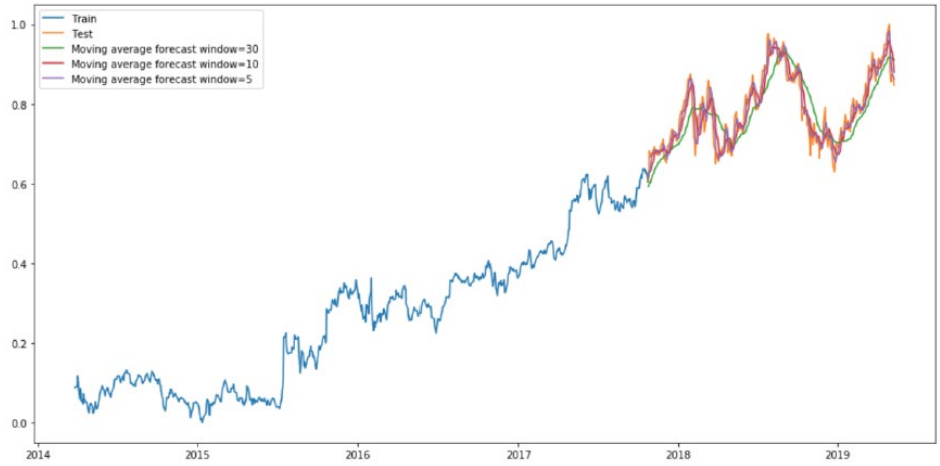


Рисунок 2.2. – Графік прогнозу простого ковзного середнього

Метод виваженого середнього було реалізовано за допомогою методу ewma бібліотеки Pandas (рис. 2.3). За документацією відповідні вагові коефіцієнти обчислювалися за такою формулою:

$$\alpha = 1 - e^{\frac{log(0.5)}{halflife}},$$

Тобто параметром налаштування моделі став halflife. Це дало наступні результати для різних значень halflife:

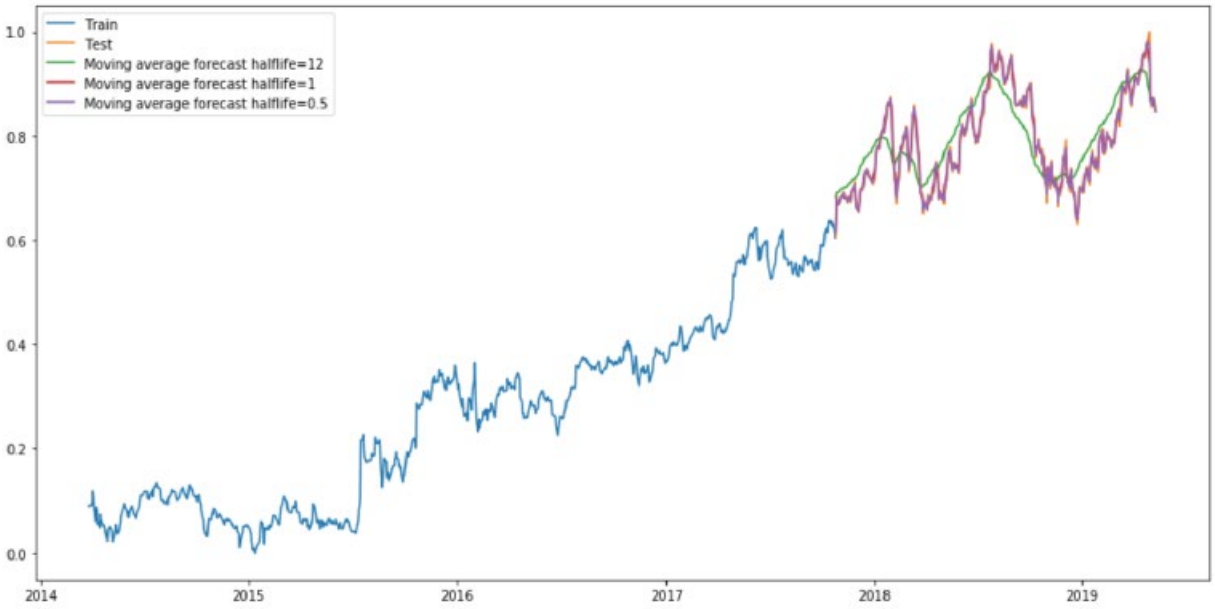


Рисунок 2.3 – Графік прогнозу методом виваженого ковзного середнього

Аналогічний підхід використовується для експоненційного ковзного середнього (рис. 2.4), де як параметр задається рівень згладжування коефіцієнт α, який є ступенем зменшення зважування від 0 до 1.

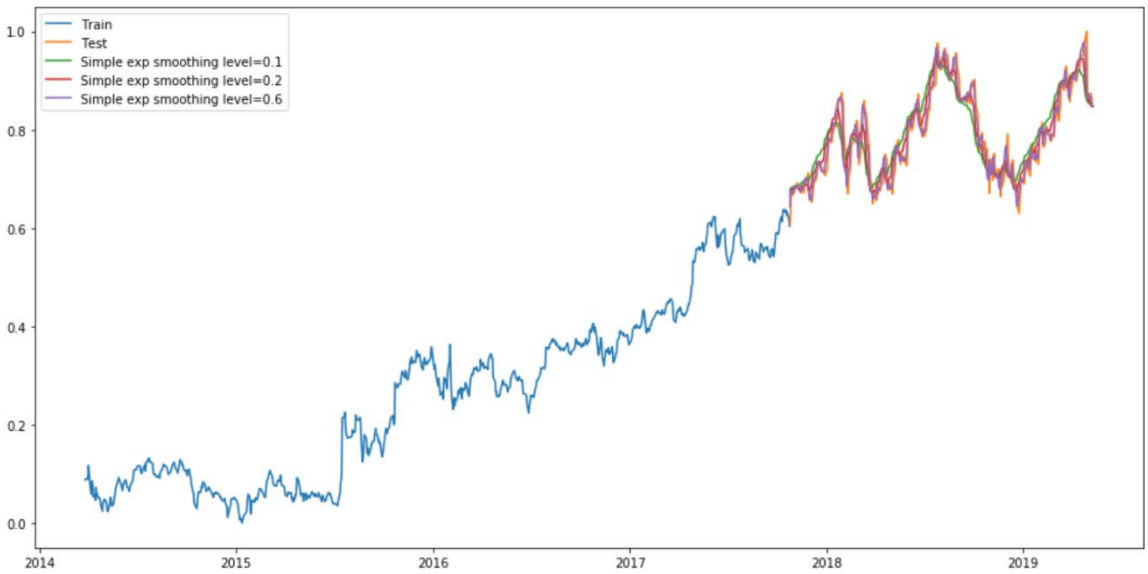


Рисунок 2.4 – Графік прогнозу методом експоненційного ковзного середнього

Чим менший рівень згладжування, тим точніше прогноз, оскільки кожне попереднє значення важить більше.

Метод подвійного експонентного згладжування передбачав завдання додаткового параметра β, який відповідає за згладжування тренду (рис. 2.5).

Комбінація пари α і β коригували точність та якість прогнозу.

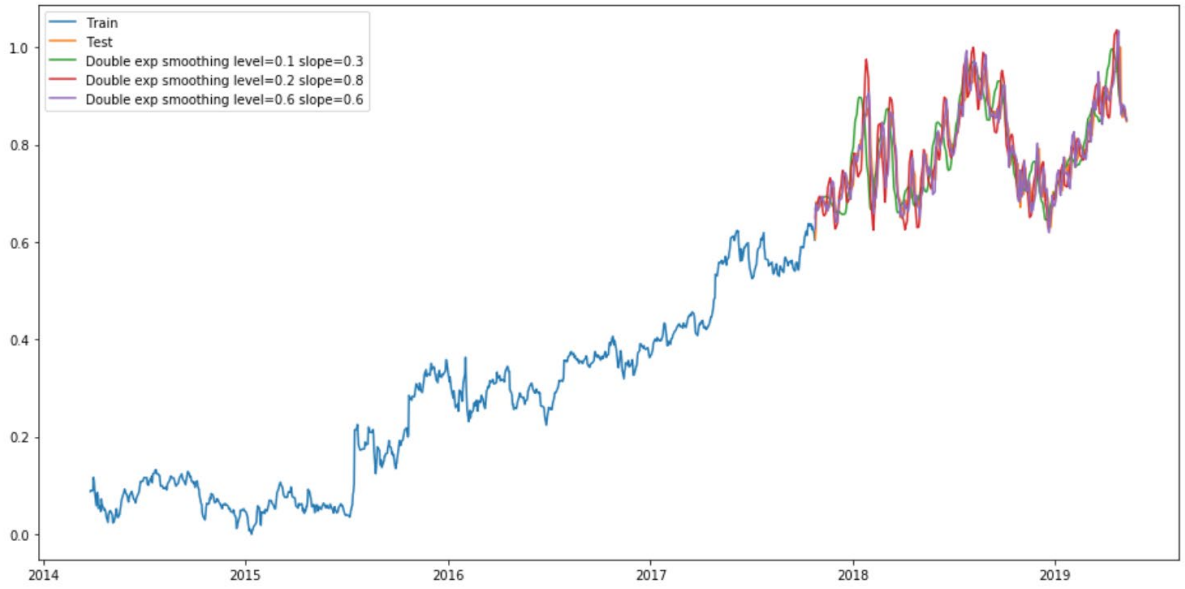


Рисунок 2.5 – Графік прогнозу методом подвійного експоненційного ковзного середнього

Потрійне експоненційне ковзне середнє або метод Голта-Вінтерса використовується для даних, в яких є як тренд, так і сезонність, і має додатковий коефіцієнт згладжування сезонної компоненти.

У роботі розглядається саме адитивна модель тренду та сезонної компоненти часового ряду. Функція ExponentialSmoothing у модулі tsa.holtwinters дозволяє задавати модель лише кількістю сезонних періодів, що є у даних.

Отриманий результат відбито у наступному малюнку (рис. 2.6).

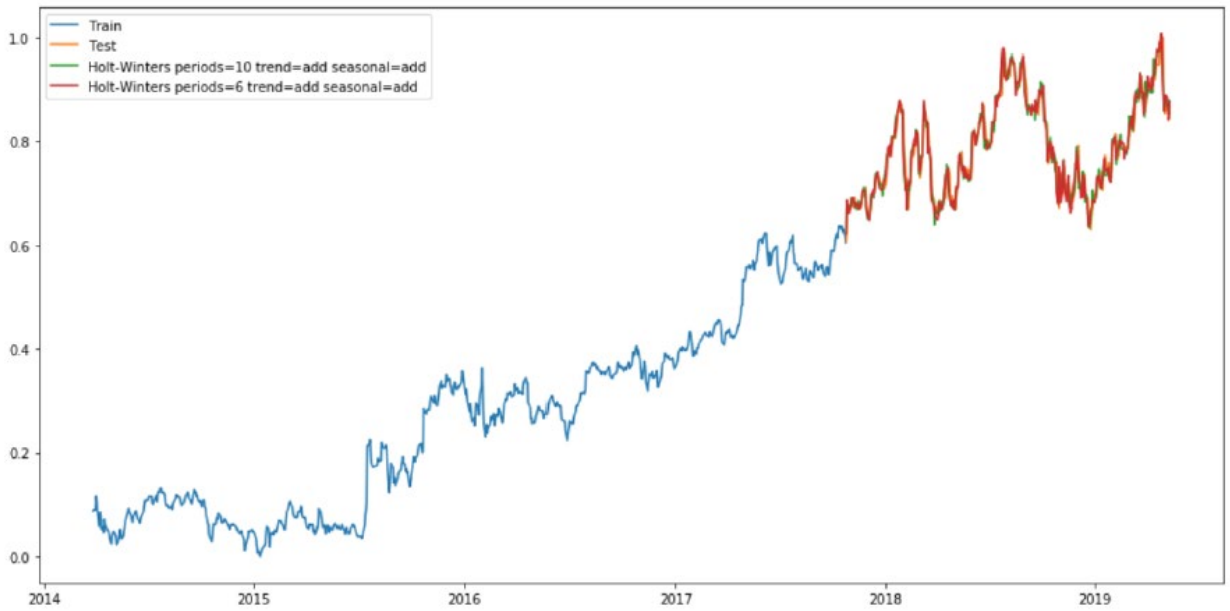


Рисунок 2.6 – Графік прогнозу методом Голта-Вінтерсу

Отже, у разі зменшення кількості періодів призводило до більш точного прогнозу.

2.3 Обробка знань та виведення рішень

Програмна частина моделі поведінки автономних сценаріїв у завдання управління інформаційними ресурсами має керувати безперервними процесами забезпечення необхідної якості обслуговування клієнта – споживача інформаційних ресурсів. Дані процеси включають управління QoS/SLA клієнта, управління QoS/SLA сервісу: моніторинг та оцінка QoS/SLA, відстеження порушень і загроз порушень SLA, формування та відстеження стану обробки заяви про проблеми QoS/SLA, формування звітності, формування оповіщень про зміни рівня якості [53].

Додаток управляє процесами активації, конфігурації та звільнення сервісів. Дані процеси включають: проектування рішення, визначення необхідних параметрів для сервісу, відстеження та управління наданням сервісу, впровадження, конфігурацію та активацію сервісу, наскрізне тестування сервісу, формування заяв для сервісу, інформування про надання сервісу, закриття сервісної заяви, вивільнення сервісу.

Дані системи призначені для підтримки всього каталогу послуг та сервісів

комп'ютерної мережі підприємства та мають бути легко розширюваними для швидкого впровадження нових послуг та сервісів.

Принципи створення.

При створенні системи слід використовувати такі принципи:

– модульність та відкритість;

– побудова на основі TMF NGOSS архітектури;

– використання сучасних гнучких програмних інструментів;

– поділ робочих процесів на рівні (послуги, технології, обладнання/ресурси);

– реалізація інтегрованих додатків та шаблонів для стандартних процесів.

Підсистема керування QoS/SLA клієнта.

Загальні характеристики функцій процесів управління якістю обслуговування клієнта та угодами про рівень обслуговування:

– призначені для порівняльної оцінки рівня обслуговування клієнта та рівня обслуговування, зазначеного в договорі з ним;

– оцінюють якість надання послуг з позиції споживача (наприклад, якість звуку під час телефонної розмови або швидкість передачі даних через ADSL);

– цей функціонал не перетинається з функціями рівнів управління продуктивністю сервісів та ресурсів;

– результати оцінки якості надання послуг необхідні дотримання умов контракту;

– якість обслуговування може визначатися як конкретного клієнта, так групи клієнтів, і навіть послуги у цілому;

– Функції керування SLA призначені для порівняння поточного QoS з рівнем обслуговування.

Вимоги захисту даних.

Для забезпечення захисту даних в інформаційній системі передбачено

розмежування прав доступу, яке регулюється шляхом встановлення паролів для користувачів, які надалі обслуговуватимуть її роботу. Для захисту даних про збої в мережі живлення ПК або аварійному завершенні роботи програми буде передбачено режим автоматичного збереження [22].

Вимоги до надійності.

Програма повинна нормально працювати при безперебійній роботі комп'ютера. При виникненні збою в роботі апаратури, відновлення нормальної роботи інформаційної системи повинно проводитися після: перезавантаження операційної системи або перезапуску файлу програми, що виконується, а також повторного виконання дій, втрачених до останнього збереження інформації.

Джерела збільшення кількості даних можна поділити на кілька категорій:

1. цифрові пристрої;
2. взаємодія людини із певними інтерфейсами;
3. обробка даних.

Обробка даних полягає в тому, щоб розглядати сирі або вже оброблені дані, щоб отримати певний результат або перейти на інший етап процесу розробки, який бере участь у конкретному проекті.

Великі дані можна визначити як обсяги даних, які доступні різною мірою складності, що створюються з різною швидкістю і різним ступенем невизначеності, які не можуть бути оброблені за допомогою традиційних технологій, методів обробки, алгоритмів або будь-яких комерційних рішень [18].

Стверджується, що є три атрибути, які виділяються як визначення характеристик великих даних:

"Великий обсяг даних: замість тисяч чи мільйонів рядків Big Data може зайняти мільярд рядків та мільйони стовпців".

Складність типів і структур даних: великі дані відображають різноманітність нових джерел даних, форматів та структур, у тому числі

цифрових слідів, що залишаються в Інтернеті та інших цифрових сховищах для подальшого аналізу.

Швидкість створення нових даних та зростання: великі дані можуть описувати дані з високою швидкістю, швидкою передачею даних та аналізом в режимі реального часу. На рис. 2.7 зображені основні атрибути Big Data.

Великі дані - це сховище для набору масивів даних, таких великих і складних, які важко обробити за допомогою традиційних інструментів управління базою даних або програм обробки даних [61].

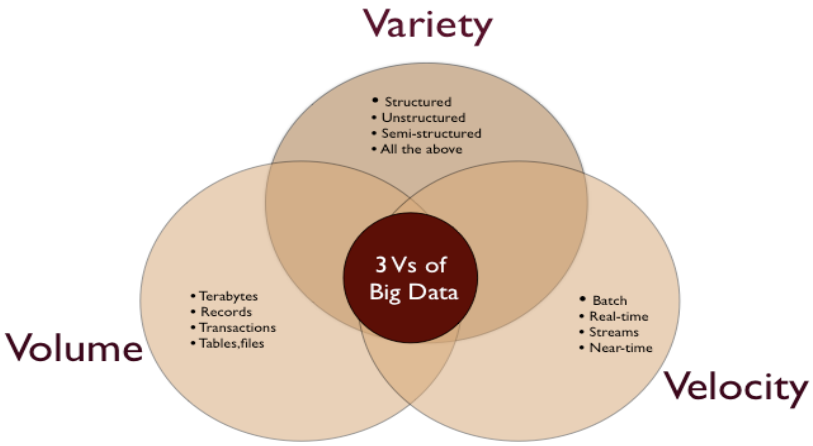


Рисунок 2.7 – Атрибути Big Data

Зазвичай, для опису різних аспектів великих даних використовуються три атрибути:

– Об’єм;
– швидкість;
– структурованість.

Ці три атрибути дозволяють визначити природу даних, доступних для аналізу.

Багато можна отримати з текстових даних, даних геолокації або файлів журналів. Наприклад, комунікативні шаблони електронною поштою, споживчі переваги та тенденції, дослідження безпеки [14]. Технології BD пропонують рішення для отримання корисної інформації із цих величезних

обсягів даних.

Дані надходять користувачам із великою швидкістю. Інтернет та мобільні технології дозволили компаніям налагодити зворотний зв'язок зі своїми клієнтами.

Технології BD з іншого боку сприяють повному збереженню всіх даних для подальшого аналізу. Головний принцип технологій BD полягає в тому, що в кожному біті даних може бути прихована важлива та цінна інформація.

Раніше для зберігання великих обсягів даних використовувалися реляційні СУБД. Технології BD перевищують реляційні СУБД за низкою критеріїв, включаючи потребу у складніших резервних копіях, відновленні та більш швидких алгоритмів пошуку.

Переваги використання технологій BD можуть призвести до втрати конфіденційності даних.

2.4 Обґрунтування вибору середовища розробки

Для реалізації такої технології знадобилося створення цілої інфраструктури. Загальномовна інфраструктура (CLI) – це набір специфікацій, що визначають певні аспекти технології .NET [54]:

2.4.1 Загальна система типів (CTS)

CTS – це певний стандарт, який визначає способи взаємодії програм та його частин, написаних різними мовами програмування. Кожен компілятор, що виробляє інструкції CIL, повинен використовувати тільки типи даних CTS.

2.4.2 Загальна проміжна мова (CIL)

Це проміжна мова інструкцій абстрактного процесора. На відміну від байт-коду, CIL також описуються спеціальні метадані, необхідні для самостійної роботи програми.

2.4.3 Розширювані метадані

У процесі компілювання програми CIL-інструкції поміщаються у спеціальну одиницю розповсюдження – складання, та супроводжуються метаданими, які роблять складання повністю самодостатнім об'єктом. До

метаданих відносяться ім'я, версія складання, дані про типи, включені в складання, список зовнішніх файлів, від яких залежить складання і т. д.

2.4.4 Бібліотека класів (.NET Framework)

Бібліотека класів – ключовий компонент технології.

Об'єктна орієнтація є особливістю технології. У .NET класи організовані у вигляді деревоподібної структури, що гілкується, які також називаються просторами імен.

Усі стандартні простори імен породжені від простору System, де описаний базовий клас Object. У просторі System визначені значні та посилання типи даних, події, інтерфейси та атрибути.

Автоматичне управління пам'яттю є одним із сервісів, які CLR надає під час керованого виконання. Під час збору сміття відбувається звільнення пам'яті. Це позбавляє розробника необхідності писати відповідний код. Завдяки автоматичному керуванню пам'яттю вирішуються типові проблеми, такі як витік пам'яті чи спроба звільнити вже знищений об'єкт [16].

Коли ініціюється новий процес, йому резервується безперервний адресний простір, зване керованої купою. Керована купа підтримує покажчик на наступний об'єкт, що розподіляється в пам'яті. Спочатку він вказує на базову адресу керованої купи. Усі типи покажчиків розподіляються в купі. Коли створюється перший покажчик, пам'ять асоційованого з ним типу починається з базової адреси купи. При створенні наступного покажчика пам'ять виділяється безпосередньо за першим. Поки адресний простір доступний, процес продовжується описаним чином.

Висновки до розділу 2

В результаті виконання розділу було досліджено залежності коефіцієнтів початкової регресії від відносної ширини лінії вхідних реалізацій, розглянуто моделі представлення знань. Виконано обробку знань та виведення рішень.

Розділ 3 Методи аналізу часових рядів з використанням нейронних мереж

3.1 Рекурентні нейронні мережі для аналізу числових рядів

Рекурентні нейронні мережі не так сильно відрізняються від звичайних нейронних мереж. Їх можна уявити, як безліч копій однієї і тієї ж мережі, причому кожна копія передає повідомлення наступної копії [27].

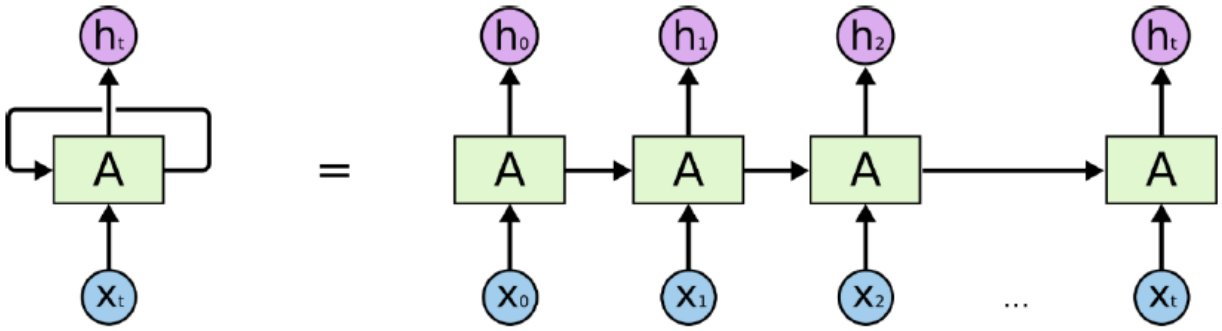


Рисунок 3.1 – Передача копій

Така “ланцюгова” сутність показує, що рекурентні нейронні мережі за своєю природою тісно пов'язані з послідовностями і списками.

Це базова архітектура, розроблена в 1980-х. Мережа будується з вузлів, кожен з яких з'єднаний з усіма іншими вузлами. У кожного нейрона поріг активації змінюється з часом і є речовим числом. Кожне з'єднання має змінну дійсну вагу. Вузли поділяються на вхідні, вихідні та приховані.

Для навчання з учителем з дискретним часом, кожен (дискретний) крок часу на вхідні вузли подають дані, а інші вузли завершують свою активацію, і вихідні сигнали готують для передання нейроном наступного рівня. Якщо, наприклад, мережа відповідає за розпізнавання мови, у результаті на вихідні вузли надходять уже мітки (розпізнані слова).

У навчанні з підкріпленням (reinforcement learning) немає вчителя, що забезпечує цільові сигнали для мережі, натомість інколи використовується функція придатності[en] або функція оцінки (reward function), за якою проводять оцінювання якості роботи мережі, при цьому значення на виході впливає на поведінку мережі на вході. Зокрема, якщо мережа реалізує гру, на

виході вимірюється кількість пунктів виграшу або оцінки позиції. Кожен ланцюжок обчислює помилку як сумарну девіацію за вихідними сигналами мережі. Якщо є набір зразків навчання, помилка обчислюється з урахуванням помилок кожного окремого зразка.

Одна з ідей, яка робить РНЗ настільки привабливими, полягає в тому, що вони могли б використовувати отриману в минулому інформацію для поточних завдань. Наприклад, вони могли б використати попередні кадри відео для розуміння наступних.

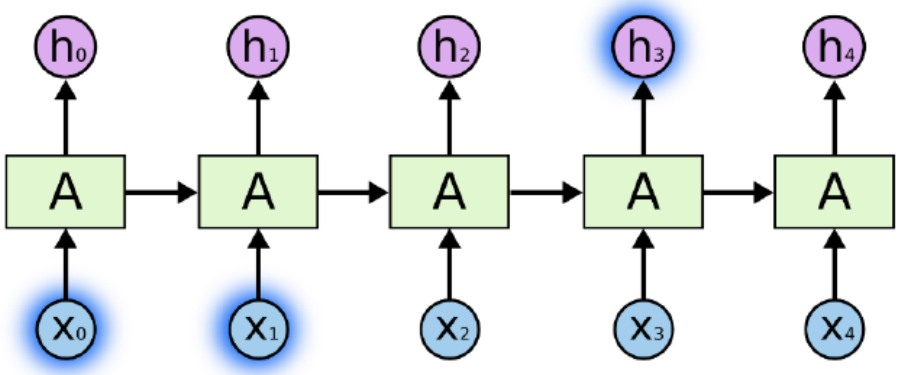


Рисунок 3.2 – Довгострокові залежності

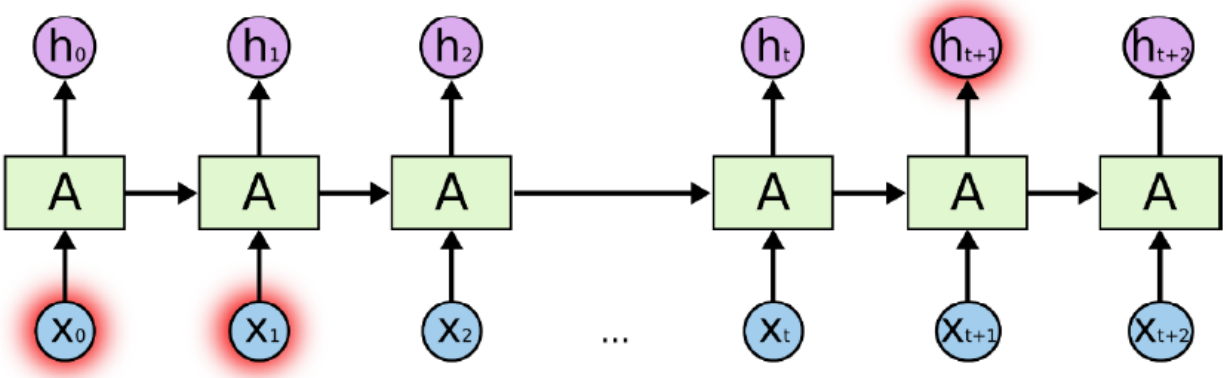


Рисунок 3.3 – Проблема поєднання

Теоретично РНС здатні обробляти такі довгострокові залежності. Людина може ретельно підібрати їх параметри, щоб вирішувати іграшкові проблеми такої форми. Проте, практично, РНС не здатні вивчити таке.

Мережі довготривалої пам'яті (Long Short Term Memory) - зазвичай просто називають "LSTM" - особливий вид РНС, здатних до навчання довгострокових залежностей. Вони працюють неймовірно добре на великому розмаїтті проблем і зараз широко застосовуються. LSTM спеціально спроектовано таким чином, щоб уникнути проблеми довгострокових залежностей. Запам'ятовувати інформацію на тривалий період часу - це практично їхня поведінка за замовчуванням, а не щось таке, що вони тільки намагаються зробити.

У мережах LSTM вдалося обійти проблему зникнення або зашкалювання градієнтів у процесі навчання методом зворотного поширення помилки. Мережа LSTM зазвичай управляється за допомогою рекурентних вентилів, які називаються вентилі (gates) «забування». Помилки поширюються назад у часі через потенційно необмежену кількість віртуальних шарів. Таким чином відбувається навчання в LSTM, при цьому зберігаючи пам'ять про тисячі і навіть мільйони часових інтервалів у минулому. Топології мереж типу LSTM можуть розроблятися відповідно до специфіки завдання. У мережі LSTM навіть великі затримки між значними подіями можуть враховуватися, і тим самим високочастотні та низькочастотні компоненти можуть поєднуватися.

Усі рекурентні нейронні мережі мають форму ланцюга повторювальних модулів (repeating module) нейронної мережі. У стандартній РНС ці модулі, що повторюють, матимуть дуже просту структуру, наприклад, всього один шар гіперболічного тангенсу (tanhtanh).

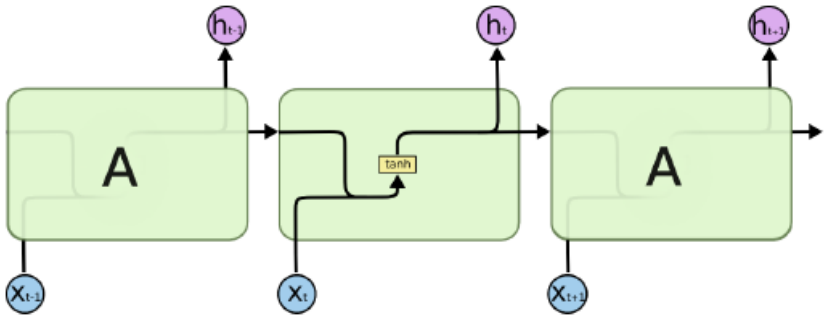


Рисунок 3.5 – Ланцюгова структура

LSTM теж мають таку ланцюгову структуру, але модуль, що повторює, має іншу будову. Замість одного нейронного шару їх чотири, причому вони взаємодіють особливим чином.

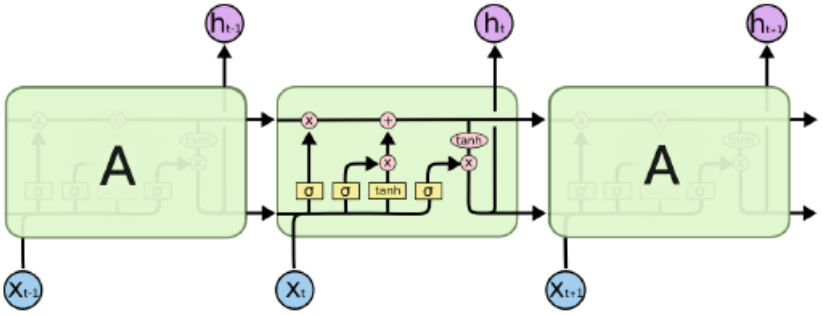


Рисунок 3.6 – LSTM-модель

У діаграмі вище, кожна лінія передає цілий вектор від виходу одного вузла до входів інших. Рожеві кола представляють поточкові оператори, такі як складання векторів, тоді як жовті У діаграмі вище кожна лінія передає цілий вектор від виходу одного вузла до входів інших. Рожеві кола представляють поточкові оператори, такі як складання векторів, тоді як жовті прямокутники - це навчені шари нейронної мережі. Лінії, що зливаються, позначають конкатенацію, в той час як лінії, що гілкуються, позначають, що їх вміст копіюється, і копії відправляються в різні місця [67].

3.2 Основи роботи з часовими рядами в TensorFlow

TensorFlow - це бібліотека програмного забезпечення з відкритим кодом, створена Google, яка використовується для впровадження систем машинного навчання та глибокого навчання [14]. Ці два імені містять низку потужних алгоритмів, які поділяють загальне завдання - дозволити комп'ютеру дізнатися, як автоматично визначати складні шаблони та/або приймати найкращі можливі рішення. TensorFlow, в основі своєї, є бібліотекою для

програмування потоку даних. Він використовує різні методи оптимізації, щоб зробити обчислення математичних виразів простіше та ефективніше.

Деякі з ключових особливостей TensorFlow:

– Ефективно працює з математичними виразами, що включають багатовимірні масиви.

– Оптимальна підтримка глибоких нейронних мереж та концепцій машинного навчання

– Використання GPU/CPU, де один і той же код може бути виконаний на обох архітектурах

– Висока масштабованість обчислень на машинах та величезні масиви даних

Встановити TensorFlow можна за допомогою команди:

pip install tensorflow

Завантаження та складання TensorFlow може тривати кілька хвилин.

У TensorFlow обчислення описується з використанням графів потоків даних. Кожен вузол графа являє собою екземпляр математичної операції (наприклад, додавання, поділ або множення), і кожне ребро є багатовимірним набором даних (тензор), на якому виконуються операції.

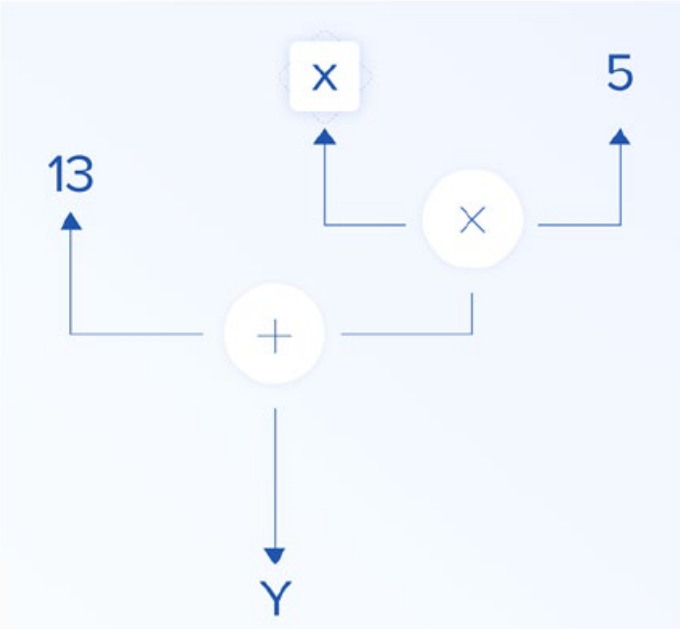


Рисунок 3.7 – Графи потоків даних

У TensorFlow константи створюються за допомогою функції:

constant(value, dtype=None, shape=None, name='Const', verify_shape=False),

де value постійне значення, яке буде використовуватися при подальших обчисленнях, dtype є параметром, що вказує тип даних (наприклад, float32/64, int8/16), shape є необов'язковим параметром, що вказує розмір масиву даних, name є необов'язковим ім'ям для тензора. Якщо вам потрібні константи з певними значеннями всередині вашої навчальної моделі, тоді об'єкт типу constant може використовуватися як у наступному прикладі [8]:

z = tf.constant(5.2, name="x", dtype=tf.float32)

Змінні TensorFlow є буферами в пам'яті, що містять тензори, які повинні бути явно ініціалізовані. Просто викликаючи конструктор, ми додаємо змінну обчислювальний граф. Змінні особливо корисні, як тільки ви починаєте з моделей навчання, і вони використовуються для зберігання та оновлення параметрів. Початкове значення, передане як аргумент конструктора, являє собою тензор або об'єкт, який може бути перетворений або повернутий як тензор. Це означає, що якщо ми хочемо заповнити зміну деякими зумовленими чи випадковими значеннями, які будуть використовуватись згодом у процесі навчання та оновлені при ітераціях, ми можемо визначити її наступним чином:

k = tf.Variable(tf.zeros([1]), name="k")

Інший спосіб використання змінних у TensorFlow - це обчислення, коли ця змінна не є навчальною і може бути визначена таким чином:

k = tf.Variable(tf.add(a, b), trainable=False)

Сеанс TensorFlow інкапсулює керування та стан середовища виконання TensorFlow. Сеанс без параметрів використовуватиме стандартний граф, створений у поточному сеансі, інакше клас сеансу приймає параметр графа, який використовується в цьому сеансі для виконання. Нижче наведено

короткий фрагмент коду, де показано, як терміни, визначені вище, можуть використовуватися в TensorFlow для обчислення простої лінійної функції y=a*x+b:

```
    import tensorflow as tf
    x = tf.constant(-2.0, name="x", dtype=tf.float32)
    a = tf.constant(5.0, name="a", dtype=tf.float32)
    b = tf.constant(13.0, name="b", dtype=tf.float32)
    y = tf.Variable(tf.add(tf.multiply(a, x), b))
    init = tf.global_variables_initializer()
    with tf.Session() as session:
    session.run(init)
print(session.run(y))
```

Перевага під час роботи з графами потоків даних у тому, що модель відділена її виконання, тобто. неважливо, на якому обчислювальному пристрої виконується програмний код, на процесорі, графічному процесорі або деякій комбінації. Програма серед TensorFlow може виконуватися на процесорі чи графічному процесорі, а вся складність, пов'язана з кодом виконання прихована від користувача. Граф обчислення - це вбудований процес, який використовує бібліотеку без прямого виклику об'єкта графа. Граф обчислень може бути побудований у процесі використання бібліотеки TensorFlow без необхідності створювати об'єкти Graph [48].

Об'єкт Graph TensorFlow може бути створений в результаті простого рядка коду c = tf.add(a, b). Це код створить операційний вузол, який приймає два тензори a і b які обчислюють їхню суму c як результат.

Плейсхолдер - це спосіб, що дозволяє розробникам вводити дані до графіка обчислень через заповнювачі, які пов'язані всередині деяких виразів. Сигнатура плейсхолдера:

placeholder(dtype, shape=None, name=None)

де dtype – тип елементів у тензорах.

Перевага плейсхолдерів полягає в тому, що вони дозволяють розробникам створювати операції у обчислювальному графі взагалі, без необхідності надавати заздалегідь дані, і дані можуть бути додані під час виконання із зовнішніх джерел.

Розглянемо просте завдання множення двох цілих чисел x і y у TensorFlow та використанням плейсхолдера [57]:

```
import tensorflow as tf
x = tf.placeholder(tf.float32, name="x")
y = tf.placeholder(tf.float32, name="y")
z = tf.multiply(x, y, name="z")
with tf.Session() as session:
print(session.run(z, feed_dict={x: 2.1, y: 3.0}))
```

3.3 Візуалізація обчислювального графа за допомогою TensorBoard

TensorBoard - інструмент візуалізації для аналізу графіків потоку даних. Він може бути корисним для кращого розуміння моделей машинного навчання. З TensorBoard ви можете отримати уявлення про різні типи статистики про параметри та подробиці про частини обчислювального графа. Глибока нейронна мережа має велику кількість вузлів. TensorBoard дозволяє розробникам отримати уявлення про кожен вузл і про те, як обчислення виконується під час виконання TensorFlow [58].

Тепер давайте повернемося до нашого прикладу, де ми визначили лінійну функцію y = a*x + b.

Щоб реєструвати події з сеансу, які пізніше можна використовувати в TensorBoard, TensorFlow надає клас FileWriter. Його можна використовувати для створення файлу подій для зберігання зведень та подій. Конструктор FileWriter приймає шість параметрів і має такий вигляд:

__init__(logdir, graph=None, max_queue=10, flush_secs=120, graph_def=None, filename_suffix=None) - де потрібно вказати обов'язковий

параметр logdir, інші - значення за промовчанням. Параметр графа буде передано з об'єкта сеансу, створеного у програмі. Повний код прикладу виглядає так [64]:

```
import tensorflow as tf
x = tf.constant(-2.0, name="x", dtype=tf.float32)
a = tf.constant(5.0, name="a", dtype=tf.float32)
b = tf.constant(13.0, name="b", dtype=tf.float32)
y = tf.Variable(tf.add(tf.multiply(a, x), b))
init = tf.global_variables_initializer()
with tf.Session() as session:
merged = tf.summary.merge_all()
writer = tf.summary.FileWriter("logs", session.graph)
session.run(init)
print(session.run(y))
```

Ми додали два нові рядки, у яких створюється і використовується об'єкт FileWriter для виведення подій у файл, як описано вище.

Після запуску програми у нас з'являється файл у журналах каталогів, і щоб подивитися вміст цього файлу необхідно запустити tensorboard:

tensorboard --logdir logs/

Після відкриття http://localhost:6006 та натискання на пункт меню «Графіки» (розташований у верхній частині сторінки) ви зможете побачити графік, як на малюнку нижче:

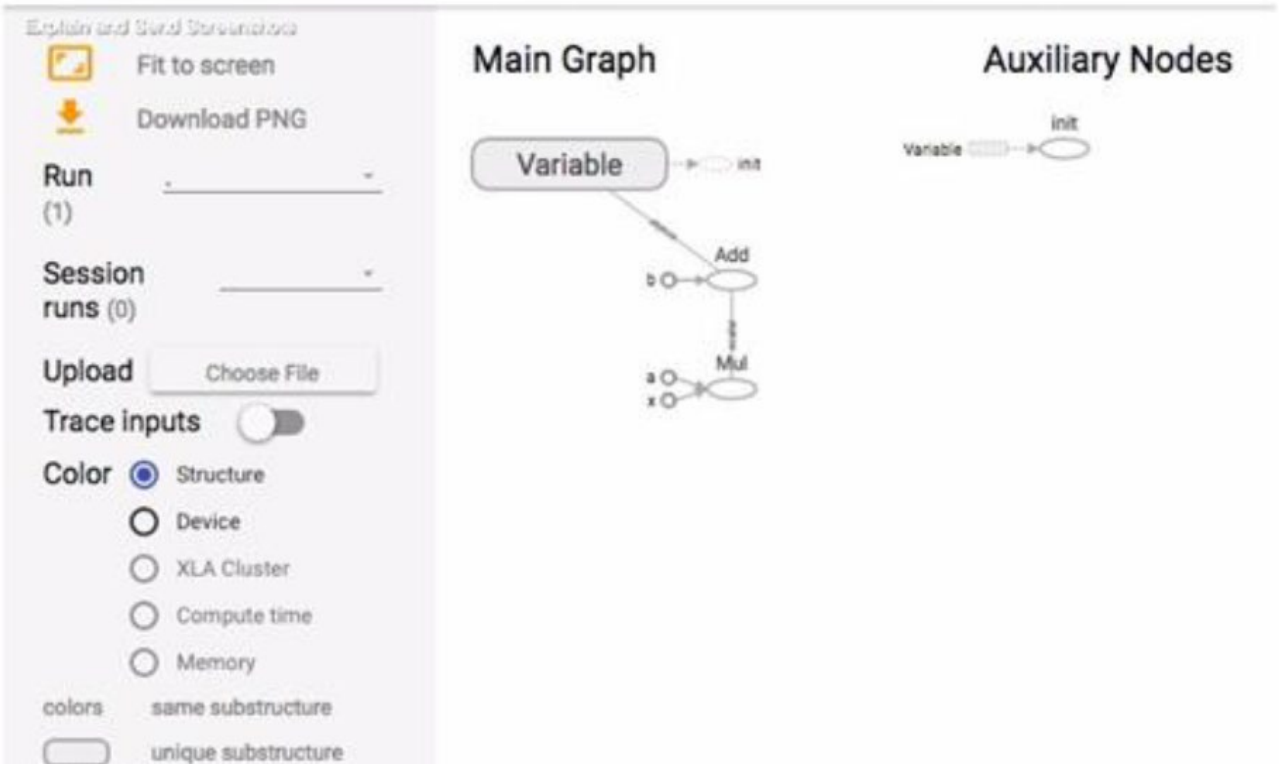


Рисунок 3.8 – Налаштування моделі

TensorBoard маркує константи та зведені вузли конкретними символами, що показані нижче.

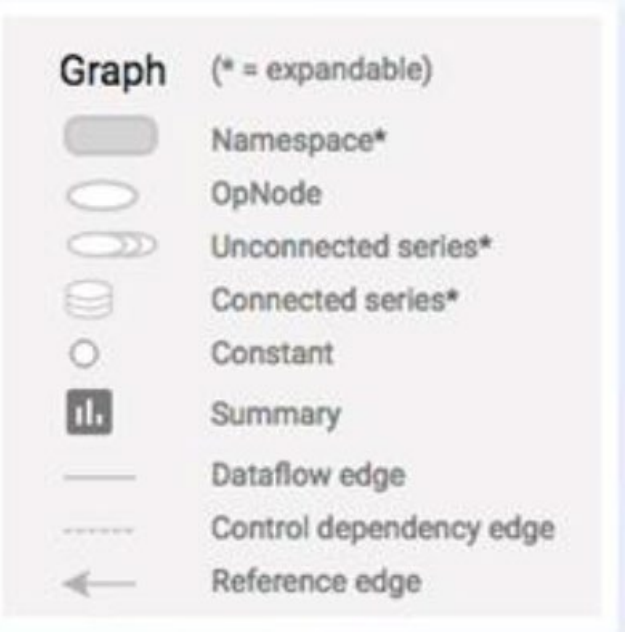


Рисунок 3.9 – Компоненти моделі потоків даних

Тензори - це основні структури даних в TensorFlow, і вони є сполучними грані в графі потоку даних.

Тензор просто ідентифікує багатовимірний масив чи список. Тензорну структуру можна ідентифікувати трьома параметрами: рангом, розміром та типом.

– Ранг: Визначає кількість вимірів тензора. Ранг відомий як порядок або n-розмірності тензора, де, наприклад, тензор рангу 1 є тензор вектора або рангу 2, є матрицею.

– Розмір: Розмір тензора – це кількість рядків та стовпців.

– Тип: Тип даних елементів тензора.

Щоб побудувати тензор TensorFlow, ми можемо побудувати n-мірний масив. Це можна зробити легко, використовуючи бібліотеку NumPy, або шляхом перетворення n-вимірного масиву Python в тензор TensorFlow.

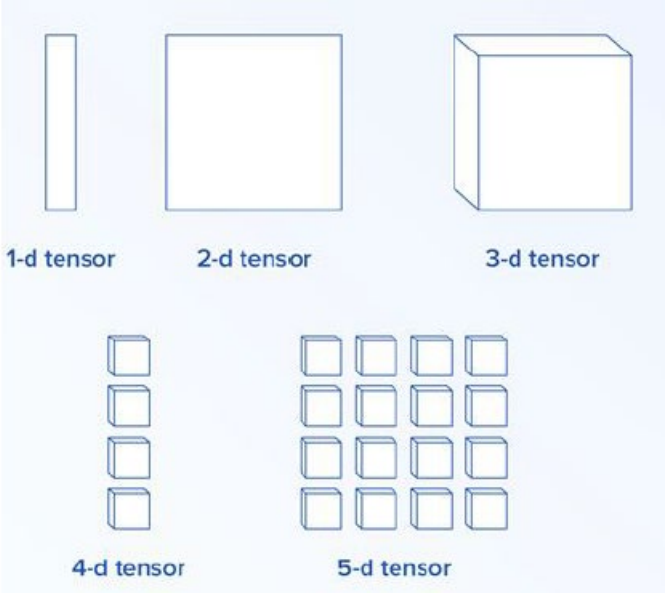


Рисунок 3.10 – Побудова тензора

Щоб побудувати 1-мірний тензор, ми будемо використовувати масив NumPy:

import numpy as np

tensor_1d = np.array([1.45, -1, 0.2, 102.1])

Робота з подібним масивом аналогічна роботі із вбудованим списком Python. Основна відмінність полягає в тому, що масив NumPy також з масивом NumPy можна легко перетворити в тензор TensorFlow використовуючи допоміжну функцію convert_to_tensor, що допомагає розробникам перетворювати об'єкти Python на об'єкти тензора. Ця функція приймає тензорні об'єкти, масиви NumPy та списки Python [51].

tensor = tf.convert_to_tensor(tensor_1d, dtype=tf.float64)

Тепер, якщо прив'яжемо наш тензор до сеансу TensorFlow, ми зможемо побачити результати нашого перетворення.

import numpy as np

import tensorflow as tf

tensor_1d = np.array([1.45, -1, 0.2, 102.1])

tensor = tf.convert_to_tensor(tensor_1d, dtype=tf.float64)

with tf.Session() as session:

print(session.run(tensor))

print(session.run(tensor[0]))

print(session.run(tensor[1]))

Висновок:

[1.45 -1. 0.2 102.1]

1.45

-1.0

Рекурентні нейронні мережі (RNN) є багатим класом динамічних моделей, які використовуються для генерації послідовностей у таких різних областях, як музика, текст і дані захоплення руху. RNN можуть бути навчені для генерації послідовностей шляхом обробки послідовностей реальних даних по одному кроку за раз і прогнозування того, що буде далі. Припускаючи, що прогнози є ймовірнісними, нові послідовності можуть бути згенеровані з навченої мережі шляхом ітеративної вибірки вихідного розподілу мережі, а

потім подачі вибірки як вхідні дані на наступному кроці. Хоча сама мережа є детермінованою, стохастичність, запроваджена під час відбору вибірок, викликає розподіл за послідовностями. Цей розподіл є умовним, оскільки внутрішній стан мережі

Рекурентні нейронні мережі «нечіткі» у тому плані, що вони не використовують точні шаблони з даних для навчання, щоб передбачити, а скоріше використовують багатовимірну інтерполяцію між прикладами навчання. Рекурентні нейронні мережі, на відміну від шаблонних, складним шляхом синтезують та відновлюють навчальні дані та рідко генерують одні й ті самі речі двічі. Більше того, нечіткі передбачення не страждають від «прокляття розмірності», і тому вони набагато краще підходять для моделювання реальних чи багатовимірних даних, ніж точні збіги (див. рис. 3.11).

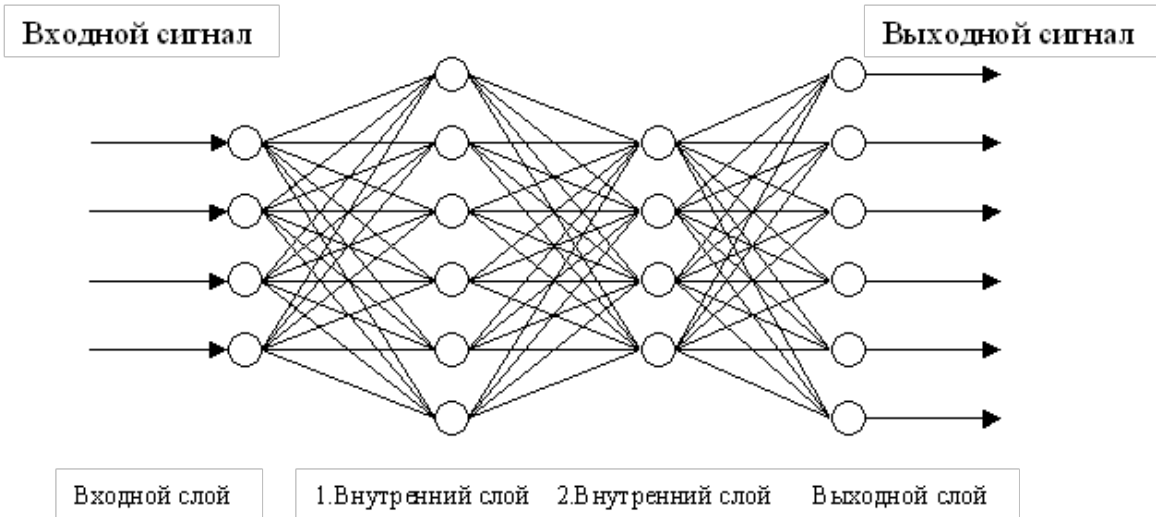


Рисунок 3.11 – Архітектура рекурентної нейронної мережі

Те, що рекурентні нейронні мережі дають хороші результати під час роботи з різними послідовностями зумовлено особливостями їхньої архітектури. Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) – це мережі, що

містять зворотні зв'язки та дозволяють зберігати інформацію. Рекурентну мережу можна розглядати як кілька копій однієї і тієї ж мережі, кожна з яких передає інформацію наступної копії. Якщо її розгорнути, ми побачимо, що RNN нагадують ланцюжок (див. рис. 3.12). Це і говорить про те, що вони тісно пов'язані із послідовностями та списками. Таким чином, RNN – найприродніша архітектура нейронних мереж для роботи з даними такого типу.

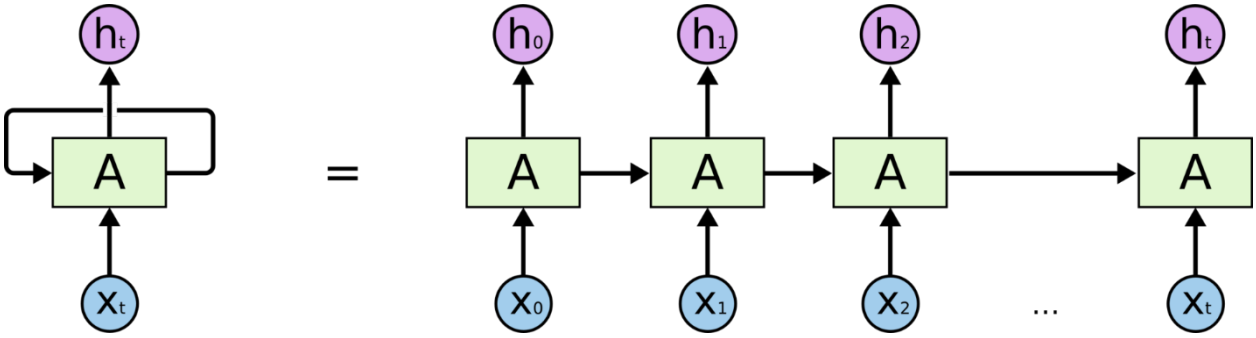


Рисунок 3.12 – Подання рекурентної нейронної мережі у вигляді послідовності

Довгострокова короткочасна пам'ять (LSTM – Long short-term memory) – це архітектура RNN, призначена для покращеного зберігання та доступу до інформації, ніж стандартні RNN, т.к. вона здатна до навчання довгострокових залежностей.

LSTM-модуль – це рекурентний модуль мережі, здатний запам'ятовувати значення як у короткі, і довгі проміжки часу. Ключем до цієї можливості є те, що LSTM-модуль (див. рис. 3.13) не використовує функції активації всередині своїх рекурентних компонентів. Таким чином, значення, що зберігається, не розмивається в часі, і градієнт або штраф не зникає при використанні методу зворотного поширення помилки в часі при тренуванні мережі.

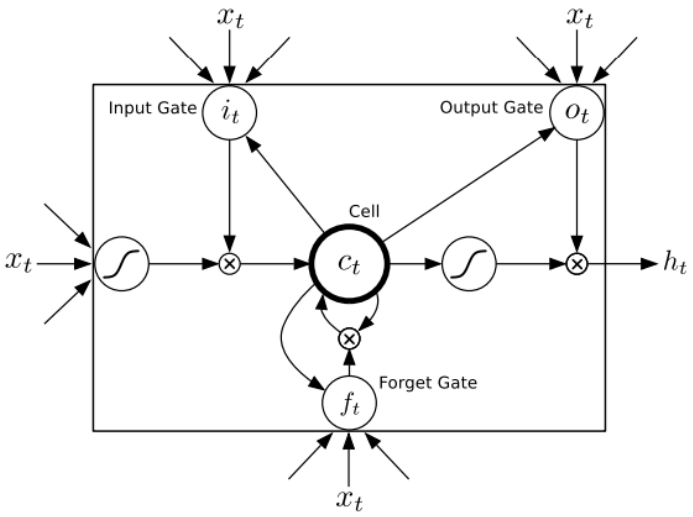


Рисунок 3.13 – Клітка довгої короткострокової пам'яті

Стан осередку схожий на конвеєрну стрічку (див. рис. 3.14), яка проходить безпосередньо через весь ланцюжок і бере участь лише в деяких лінійних перетвореннях. По ній інформація може текти без перешкод і не зазнавати жодних змін. Однак LSTM може видаляти інформацію зі стану комірки. Причому операція видалення регулюється спеціальними структурами, які називаються фільтрами (gates) (див. рис. 3.15).

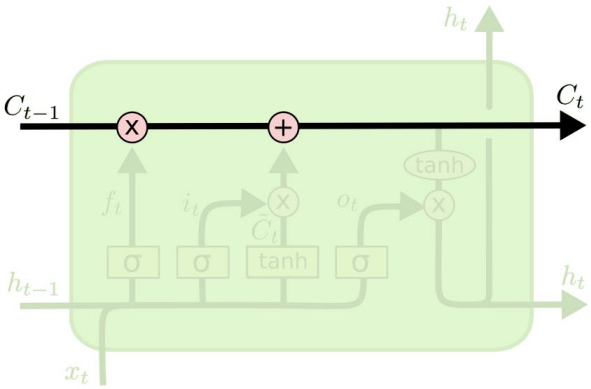


Рисунок 3.14 – Стан осередку – конвеєрна стрічка

Основою фільтрації є деякі умови, що дозволяють пропускати інформацію. Вони складаються із двох об'єктів. Ними є шар сигмоїдальної нейронної мережі та операція крапкового множення. Сигмоїдальний шар повертає числа

від нуля до одиниці, які позначають, яку частку кожного блоку інформації слід пропустити далі через мережу. Нуль у разі означає “не пропускати нічого”, одиниця – “пропустити все”. У LSTM включає три фільтри, які дозволяють контролювати і захищати стан комірки.

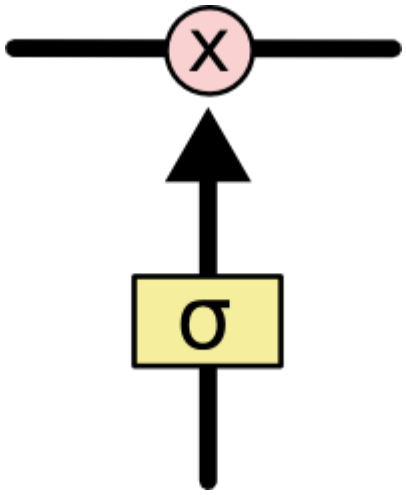


Рисунок 3.15 – Фільтр у LSTM

LSTM зараз дає передові результати в різних завданнях обробки послідовності, включаючи розпізнавання мови і почерку.

На рис. 3.16 зображено рекурентну нейронну мережу прогнозування. Вхідна векторна послідовність$x(1)$передається через зважені з'єднання в стек з N рекурентно пов'язаних прихованих шарів, щоб спочатку обчислити приховані векторні послідовності$h^n(2)$, а потім вихідну векторну послідовність (3).y

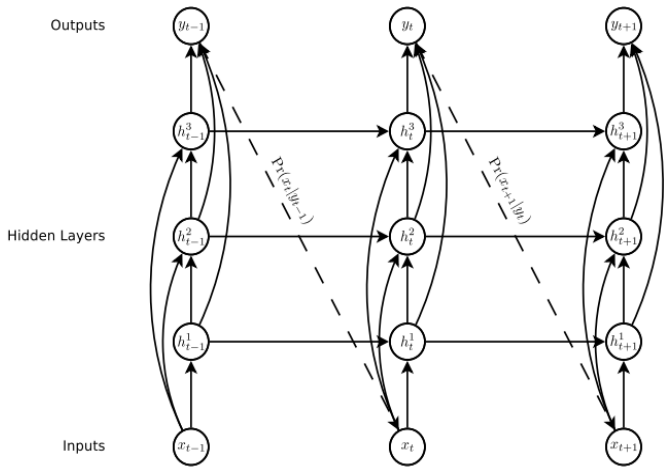


Рисунок 3.16 – Архітектура рекурентної нейронної мережі, де кола є шарами

мережі, суцільні лінії – зважені зв'язки, а пунктирні – прогнози

$$x = (x_1, \dots, x_T)(1)$$

$$h^n = (h_1^n, \dots, h_T^n)(2)$$

$$y = (y_1, \dots, y_T)(3)$$

Кожен вихідний векторy_tвикористовується для того, щоб параметризувати прогнозуючий розподіл по всіх можливих входах. Перший елемент кожної вхідної послідовності завжди є нульовим вектором, всі записи якого дорівнюють нулю; отже, мережа генерує прогноз першого реального входу без попередньої інформації. Мережа є «глибокої» як у просторі, і у часі, тому, що у кожен фрагмент інформації, що проходить або вертикально, чи горизонтально через обчислювальний граф, впливатимуть численні послідовні вагові матриці і нелінійності.$\Pr(x_{t+1}|y_t)x_{t+1}x_1x_2$

Необхідно звертати увагу на «пропуск з'єднань» від входів до всіх прихованих шарів та від усіх прихованих шарів до виходів. Це значно полегшує процес навчання для глибоких мереж, зменшуючи кількість етапів обробки між дном мережі та верхом та, таким чином, пом'якшуючи проблему зникаючого градієнта. В особливому випадку, коли архітектура зводиться до звичайного однорівневого RNN (recurrent neural network$N = 1$- Рекурентної нейронної мережі) з прогнозуванням наступного кроку. Активації прихованого шару обчислюються шляхом ітерації наступних рівнянь від до і до :$t = 1Tn = 2N$

$$h_1^t = \mathcal{H}(W_{ih^1}x_t + W_{h^1h^1}h_{t-1}^1 + b_n^1)$$

$$h_t^n = \mathcal{H}(W_{ih^n}x_t + W_{h^{n-1}h^n}h_t^{n-1} + W_{h^nh^n}h_{t-1}^n + b_n^1)$$

деW- вагові матриці (наприклад,W_{ih^n} - вагова матриця, що з'єднує входи зn-м прихованим шаром,$W_{h^1h^1}$- рекурентне з'єднання на першому прихованому шарі і т.д.),b – вектору зміщення (bias vector),$\mathcal{H}$ - Функція прихованого шару.

З урахуванням прихованих послідовностей, вихідні послідовності розраховуються так:

$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n$$

$$y_t = \mathcal{Y}(\hat{y}_t)$$

де$\mathcal{Y}$- Функція вихідного шару. Таким чином, вся мережа визначає функцію, параметризовану ваговими матрицями, від вхідних історій до вихідних векторів$x1{:}\ ty_t$.

Вихідні вектори використовуються для параметризації прогнозуючого розподілу для наступного входу. Форма має бути підібрана свідомо, спираючись на вхідні дані. Зокрема, знаходження відповідного прогнозуючого розподілу для багатовимірних реальних даних (зазвичай згадується як «моделювання щільності») може бути досить скрутним.$y_t \Pr(x_{t+1}|y_t) \Pr(x_{t+1}|y_t)$

Імовірність, задана мережею для вхідної послідовності:x

$$\Pr(x) = \prod_{t=1}^{T} \Pr(x_{t+1}|y_t)$$

а втрати послідовності, що використовується для того, щоб навчити мережу, є негативним логарифмом від :$\mathcal{L}(x)\ \Pr(x)$

$$\mathcal{L}(x) = -\sum_{t-1}^{T} \log \Pr(x_{t+1}|y_t) \quad (4)$$

Приватні похідні втрат за вагами мережі можуть бути ефективно розраховані зі зворотним розповсюдженням помилки у часі, застосованим до графа обчислень і мережа потім може бути навчена за допомогою алгоритму градієнтного спуску.

У більшості RNN функція прихованого шару є поелементним застосуванням сигмоїдальної функції. Для версії LSTM реалізована за допомогою наступної складової функції:$\mathcal{HH}$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o)$$
$$h_t = o_t \tanh(c_t)$$

Деσ-функція логістичної сигмоїди, а йi, f, o c-відповідно, вхідні вектори активації для вхідного фільтра, що пропускає фільтра, тобто. фільтра, який здатний видаляти інформацію зі стану комірки, вихідного фільтра, а також комірки та входу в комірку (input gate, forget gate, output gate, cell and cell input), кожен з яких має той же розмір, що і прихований вектор .h

Висновки до розділу 3

В результаті виконання розділу було наведено опис принципів роботи з часовими рядами за допомогою нейронних мереж. В оригінальному алгоритмі LSTM використовується спеціально розроблений розрахунок приблизного градієнта, який дає змогу оновлювати ваги після кожного тимчасового кроку. Однак натомість повний градієнт може бути розрахований зі зворотним поширенням помилки у часі, метод, який буде використовуватися далі.

Одна з труднощів при навчанні LSTM з повним градієнтом полягає в тому, що похідні іноді стають надмірно великими, що призводить до чисельних проблем. Щоб запобігти цьому, можна обрізати похідну втрат по відношенню до виходів мережі в шари LSTM (до застосування функцій сигмоїду та гіперболічного тангенсу), щоб вони знаходилися в заздалегідь визначеному діапазоні.

Розділ 4 Програмна реалізація алгоритму аналізу часових рядів

4.1 Формат вхідних та вихідних даних

Документування архітектури ПЗ спрощує процес комунікації між заінтересованими особами (англ. stakeholders), дозволяє зафіксувати прийняті на ранніх етапах проектування рішення про високорівневий дизайн системи та дозволяє використовувати компоненти цього дизайну та шаблони повторно в інших проектах [26].

Основною ідеєю програмної архітектури є ідея зниження складності системи шляхом абстракції та розмежування повноважень.

Типове рішення модель-подання-контролер має на увазі виділення трьох окремих ролей. Модель – це об'єкт, який дає деяку інформацію про домен. У моделі немає візуального інтерфейсу, вона містить у собі всі дані та поведінку, не пов'язані з призначеним для користувача інтерфейсом. В об'єктно-орієнтованому контексті найчистішою формою моделі є об'єкт моделі предметної області. Як модель можна розглядати і сценарій транзакції, якщо він не містить ніякої логіки, пов'язаної з призначеним для користувача інтерфейсом. Подібне визначення не надто розширює поняття моделі, проте повністю відповідає розподілу ролей у розглянутому типовому рішенні.

Подання відображає вміст моделі засобами графічного інтерфейсу. Таким чином, якщо модель – це об'єкт користувача, відповідне уявлення може бути кадром з великою кількістю елементів керування або HTML-сторінкою, заповненою інформацією про користувача. Функції подання полягають лише у відображенні інформації на екрані. Усі зміни інформації обробляються третім «учасником» системи – контролером. Контролер отримує вхідні дані від користувача, виконує операції над моделлю і вказує на необхідність відповідного оновлення. У цьому плані графічний інтерфейс можна як сукупність уявлення і контролера (рис. 4.1).

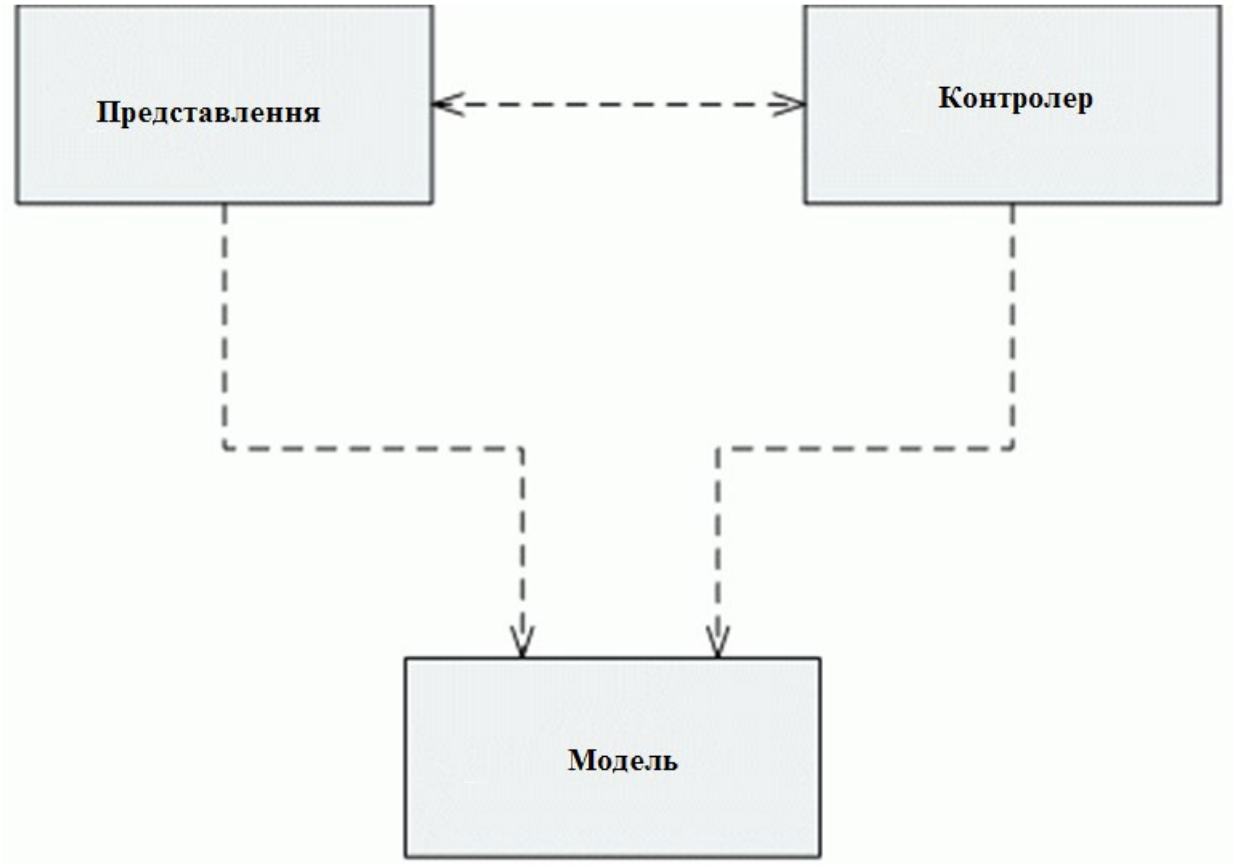


Рисунок 4.1 – Загальний вид рішення «Модель-Подання-Контролер»

Говорячи про типове рішення модель-представлення-контролер, не можна не підкреслити два основних типи поділу: відділення подання від моделі та відділення контролера від подання [21].

Відділення уявлення від моделі – це один із фундаментальних принципів проектування програмного забезпечення. Наявність такого поділу дуже важлива з низки причин:

– уявлення та модель відносяться до абсолютно різних сфер програмування;

– користувачі хочуть, щоб, залежно від ситуації, та сама інформація могла бути відображена різними способами;

– об'єкти, які не мають візуального інтерфейсу, набагато легше тестувати, ніж об'єкти з інтерфейсом [3].

Ключовим моментом у відділенні уявлення від моделі є усунення залежностей: уявлення залежить від моделі, але модель залежить від уявлення. Це означає, що зміна уявлення вимагає зміни моделі [19].

Стандартну схему архітектури "Модель-вид-контролер" зображено на рис. 4.2:

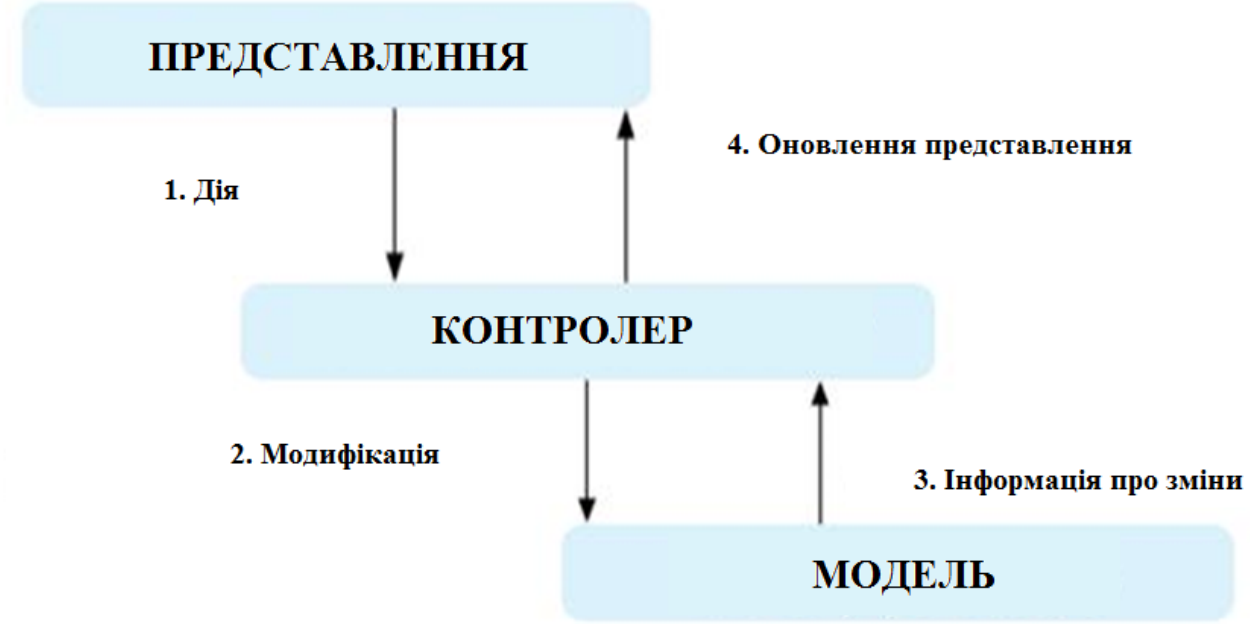


Рисунок 4.2 – Схема архітектури MVC

4.2 Модульна структура програмного засобу

Структура програмної системи прогнозування за допомогою нейронної мережі складається з таких основних програмних модулів:

1. Форма Main.cs – модуль, що містить засоби виконання класифікації та емуляції перцептрону, із встановленням числа ітерацій навчання, глибини процесорності навчання, класифікатора властивостей.

2. Форма MnemoticForm.cs – програмна форма для візуалізації розрахунків прогнозного алгоритму з введенням початкових даних, вибору параметрів навчання моделі нейронної мережі, побудови розрахункового графіка.

На формі представлені основні елементи керування програмою, вікно відображення вмісту файла-процесу, вікна статистичних відомостей, кнопки активації дій.

Вміст класів модуля структуризації оперативної пам'яті наведено на рис. 4.3.

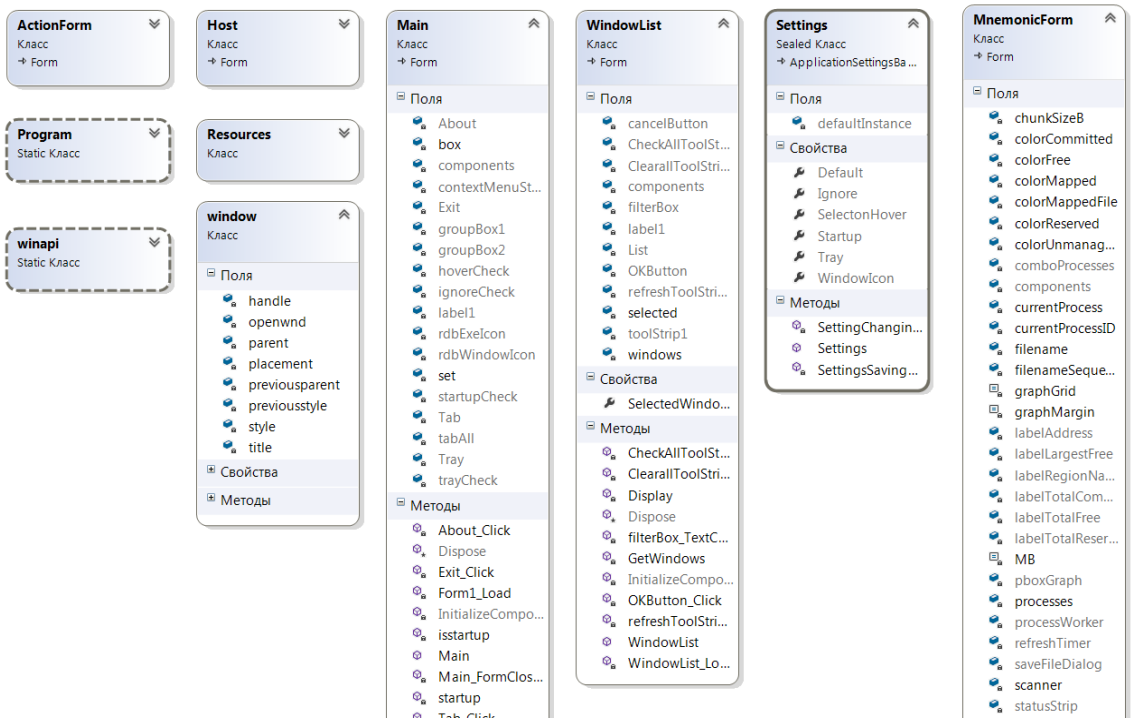


Рисунок 4.3 – Діаграма класів модуля моделювання

4.3 Діаграма потоків даних

Концептуальна схема являє собою узагальнені функціональні та інформаційні компоненти проєктованого програмного продукту, принципи їхньої взаємодії між собою, з користувачем і зовнішнім середовищем. За результатами вищеописаної роботи було спроектовано концептуальну схему системи постановки процесів на виконання та структуризації оперативної пам'яті, яку показано на рис. 4.4, де показано дві концептуальні частини - компонент оброблення даних, що надійшли в систему, і компонент аналізу ідентифікованих характеристик.

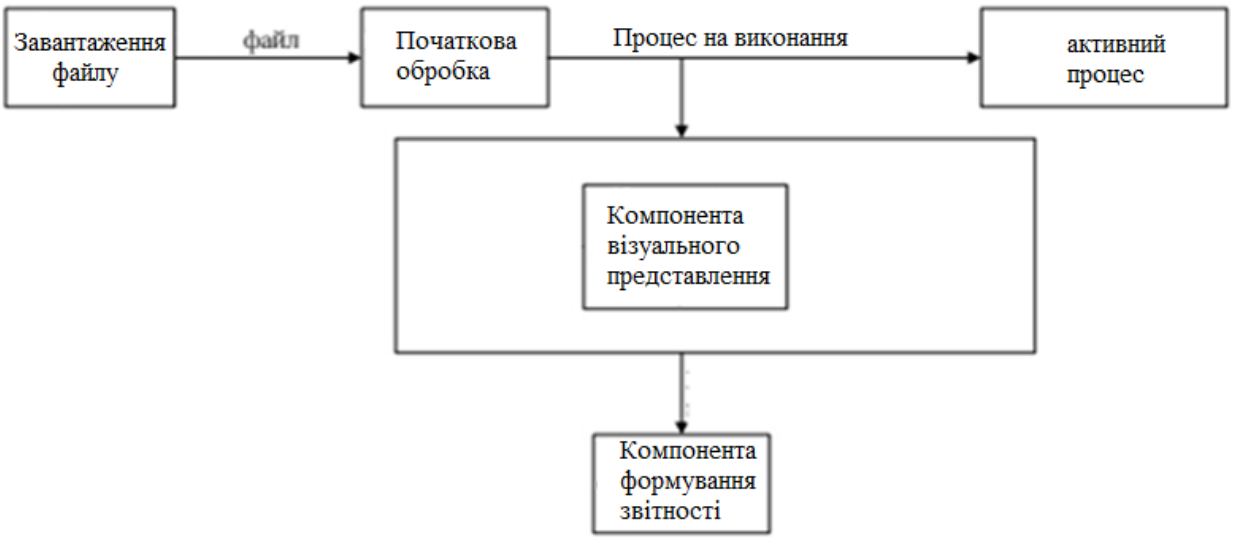


Рисунок 4.4 – Концептуальна схема розроблюваної програмної системи

4.4 Опис інтерфейсу програмного продукту

У підсистемі оцінки набору даних кількісних показників статистичних даних представлений механізм статистичної оцінки вхідних параметрів, що імпортуються за допомогою формату Excel:

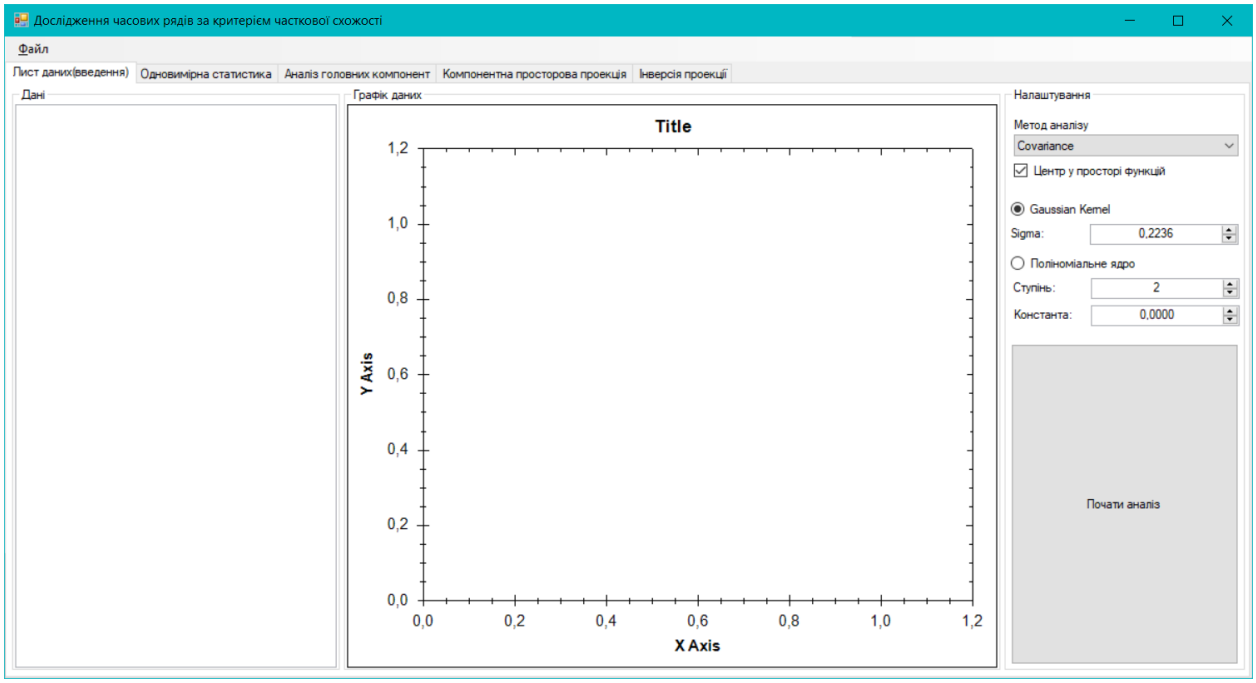


Рисунок 4.5 – Відкриття файлу даних

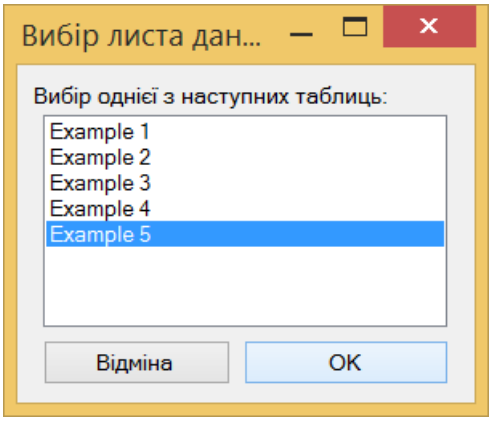


Рисунок 4.5 – Відкриття файлу даних (2)

Після відкриття достатньо натиснути кнопку «Запуск аналізу». Результати визначення статистичних показників формуються автоматично та розміщуються у вкладках головного вікна.

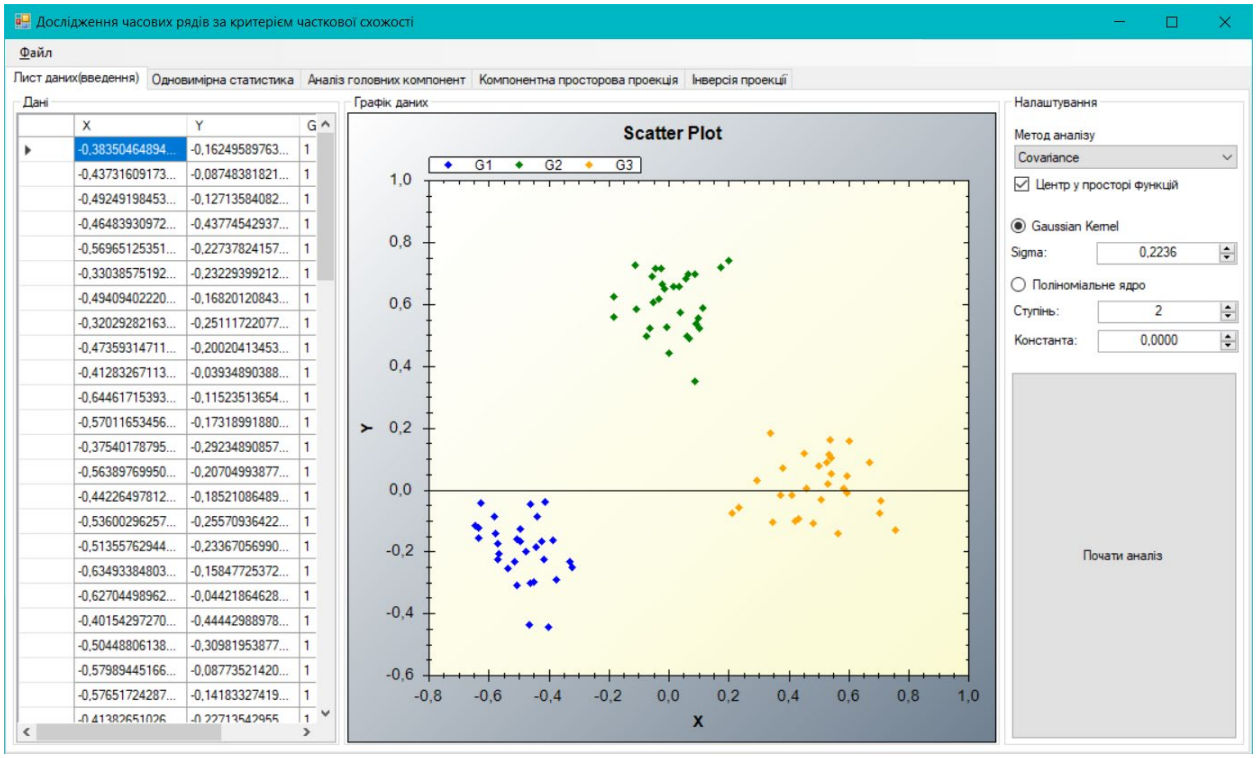


Рисунок 4.6 – Дані, готові до аналізу

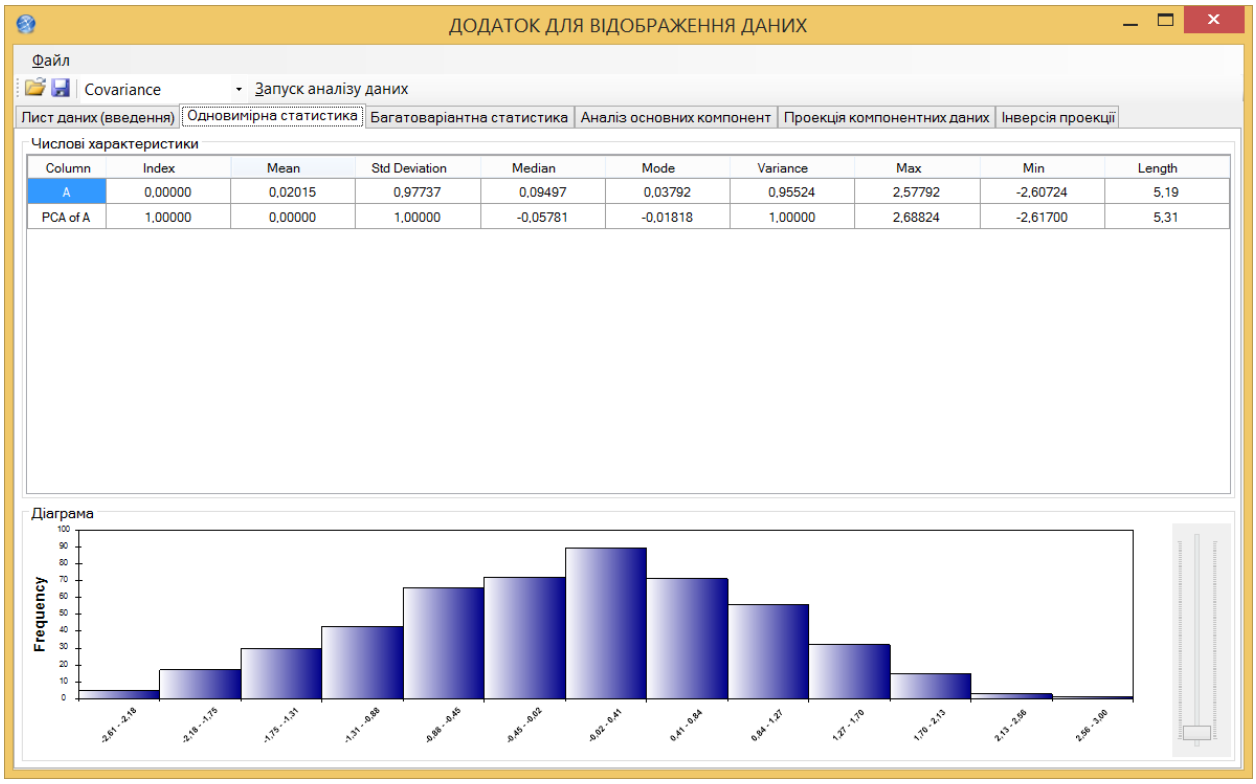


Рисунок 4.7 – Побудова гістограми розподілу

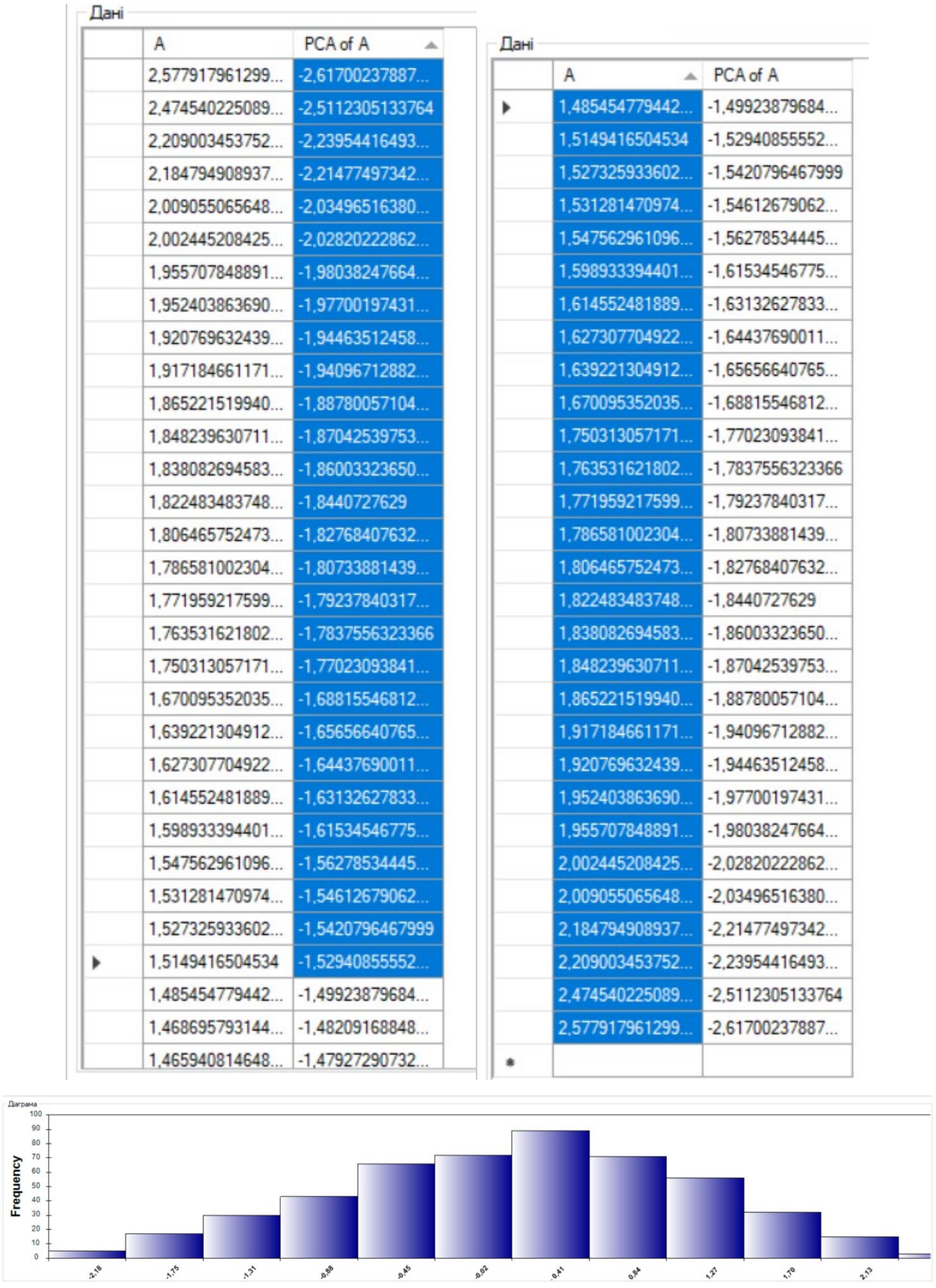


Рисунок 4.8 – Впорядкування часових рядів за критерієм часткової схожості

Реверсія даних

	A	PCA of A
	0,037918361997...	-0,01817882342...
	-0,01195975374...	0,032854421663...
	-1,11366444704...	1,160073539682...
	0,468596422362...	-0,45883097486...
	1,219638673854...	-1,22726664503...
	0,312560714467...	-0,29918162953...
	1,485454779442...	-1,49923879684...
	-0,41279082123...	0,442968352480...
	-1,98337323778...	2,049923952401...
	-0,35622987138...	0,385097505248...
	0,057654607591...	-0,03837214155...
	0,378629211462...	-0,36678020959...
	-0,82206459784...	0,861720517473...
	-1,42496984252...	1,478588469735...
	1,005899114216...	-1,00857708206...
	-0,65659339922...	0,692417164501...
	-0,55409892471...	0,587549016198...
	1,386667919075...	-1,39816412747...
	-1,33331016582...	1,384806042779...
▶	0,660621508140...	-0,65530317757...

Рисунок 4.9 – Результат реверсії даних

Дані

	A	B
▶	1,1	1,32
	1,2	1,44
	1,61	1,932
	2,12	2,544
	2,2	2,64
	2,5	3
	3,26	3,911999999999...
	4,2	5,04
	5	6
*		

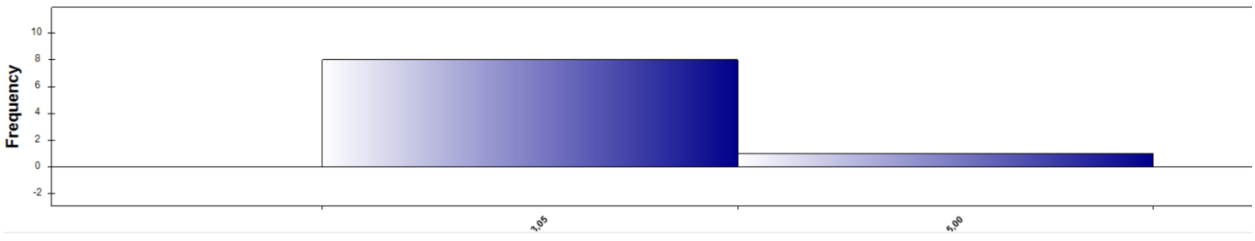


Рисунок 4.10 – Впорядкування часових рядів за критерієм часткової схожості

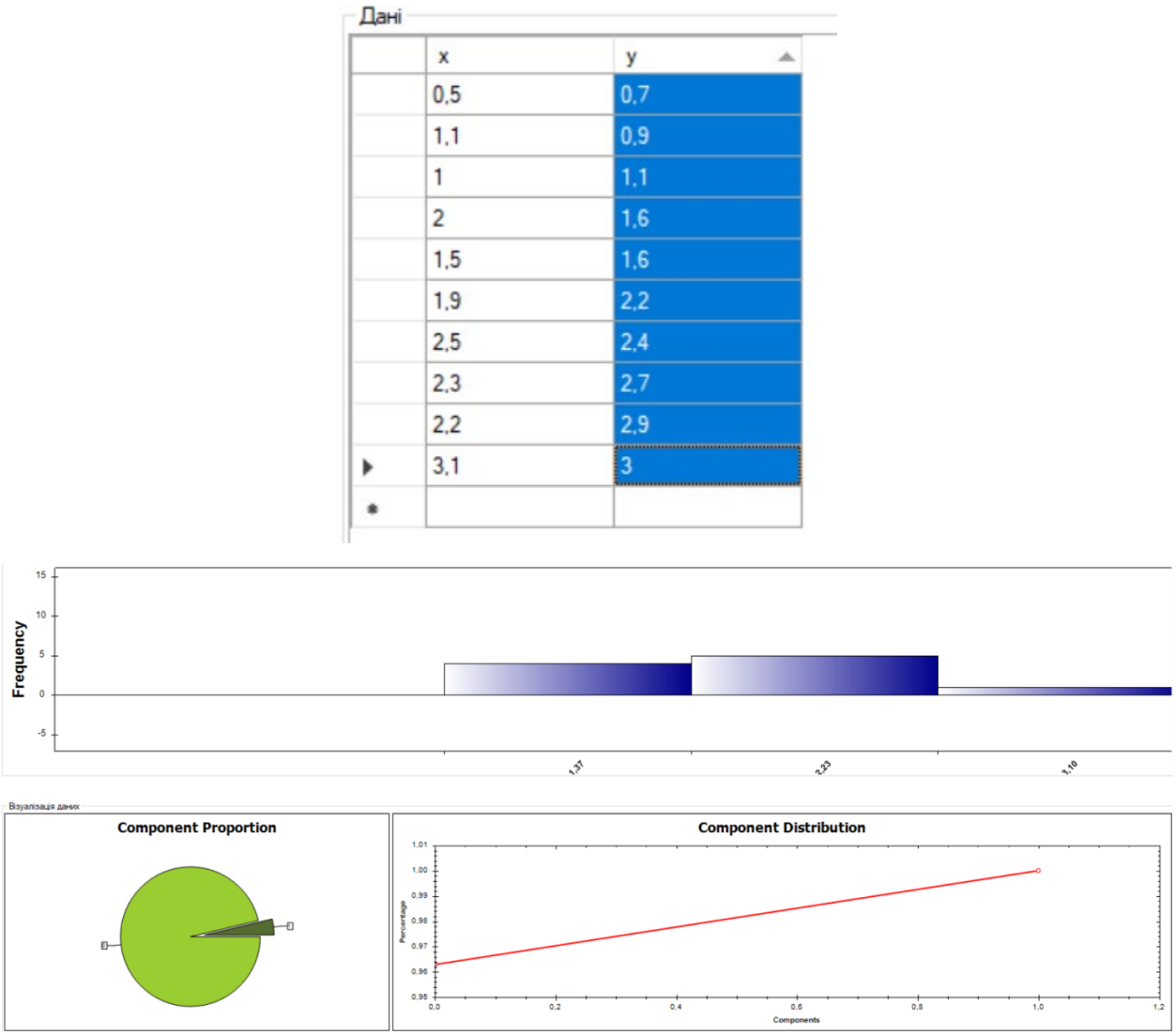


Рисунок 4.11 – Кластеризація часових рядів за критерієм часткової схожості

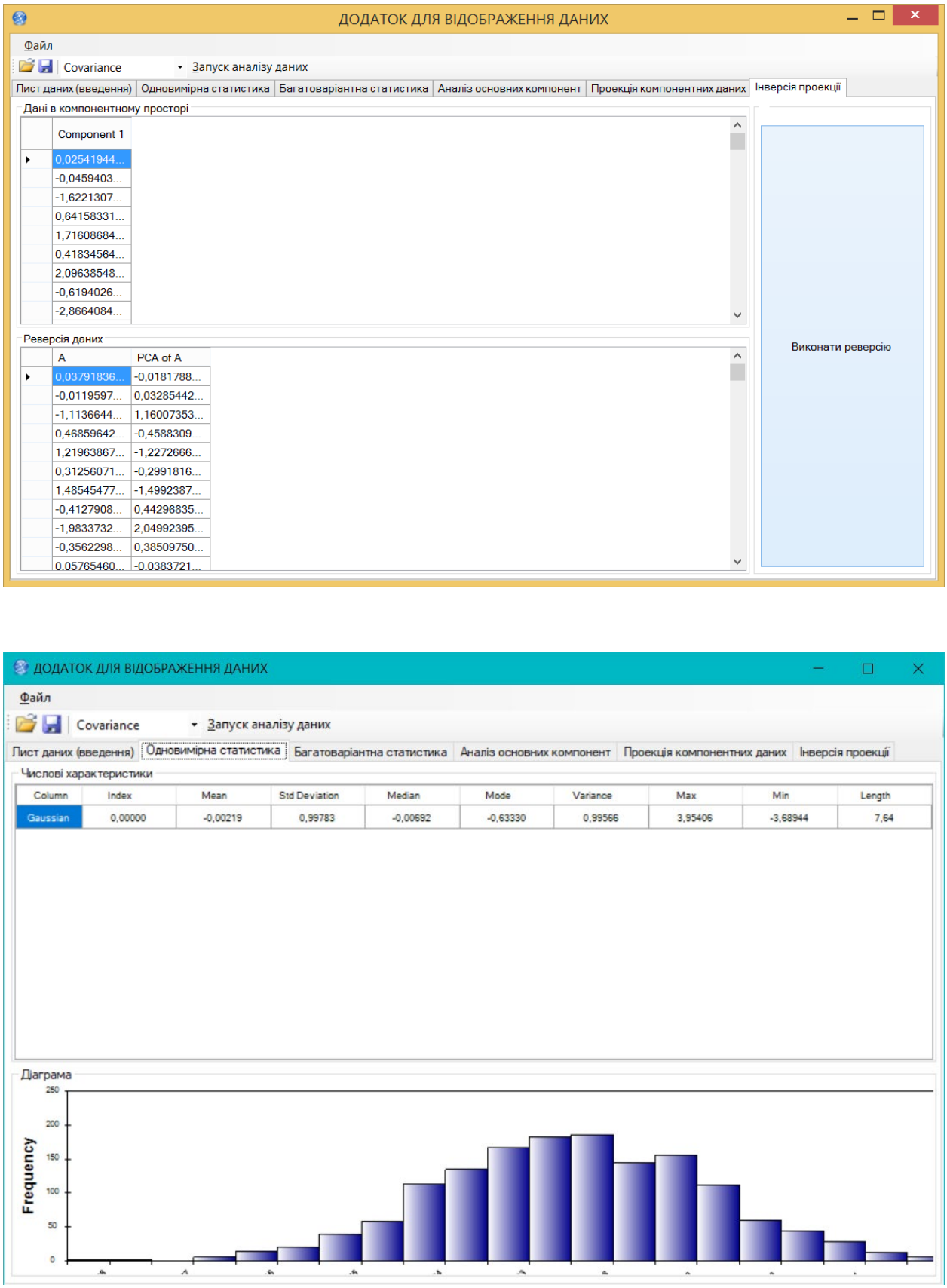
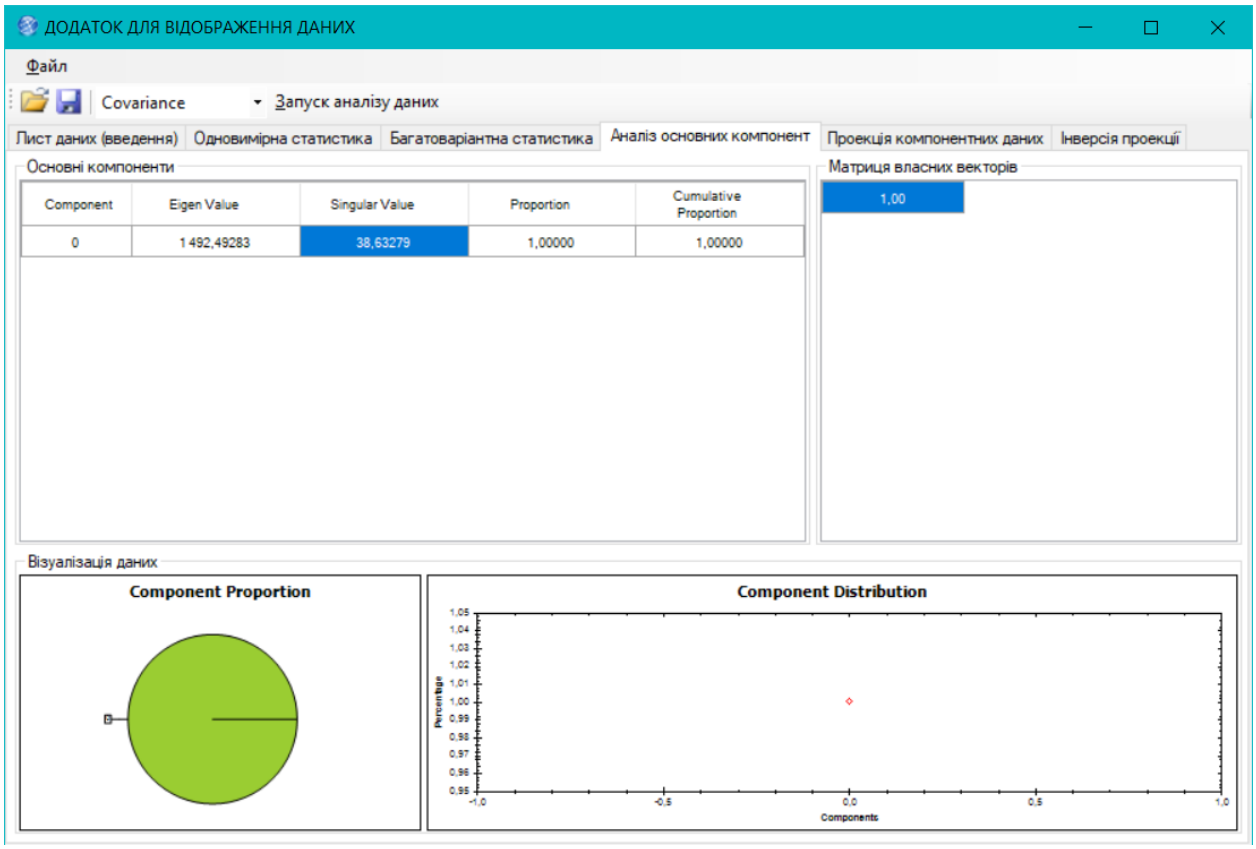


Рисунок 4.12 – Виконання проекції компонентних даних



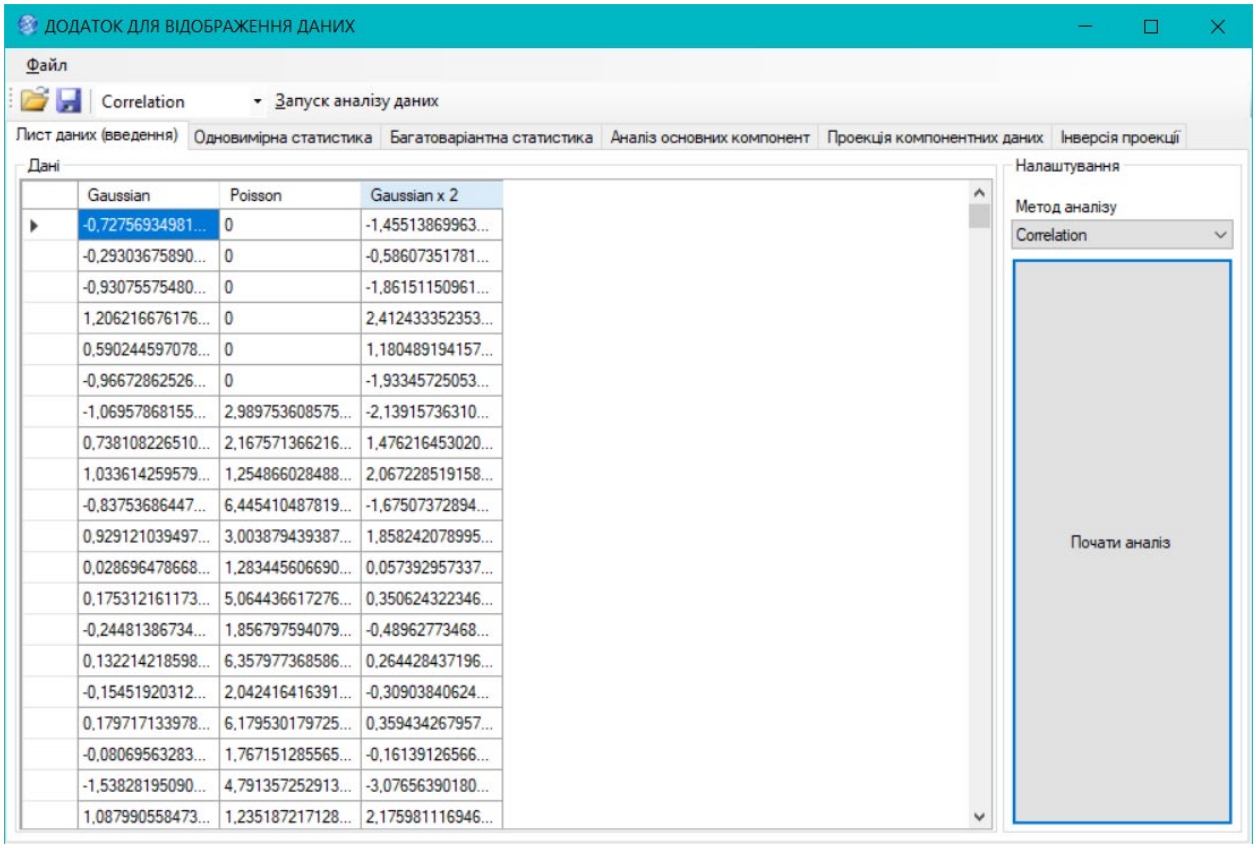
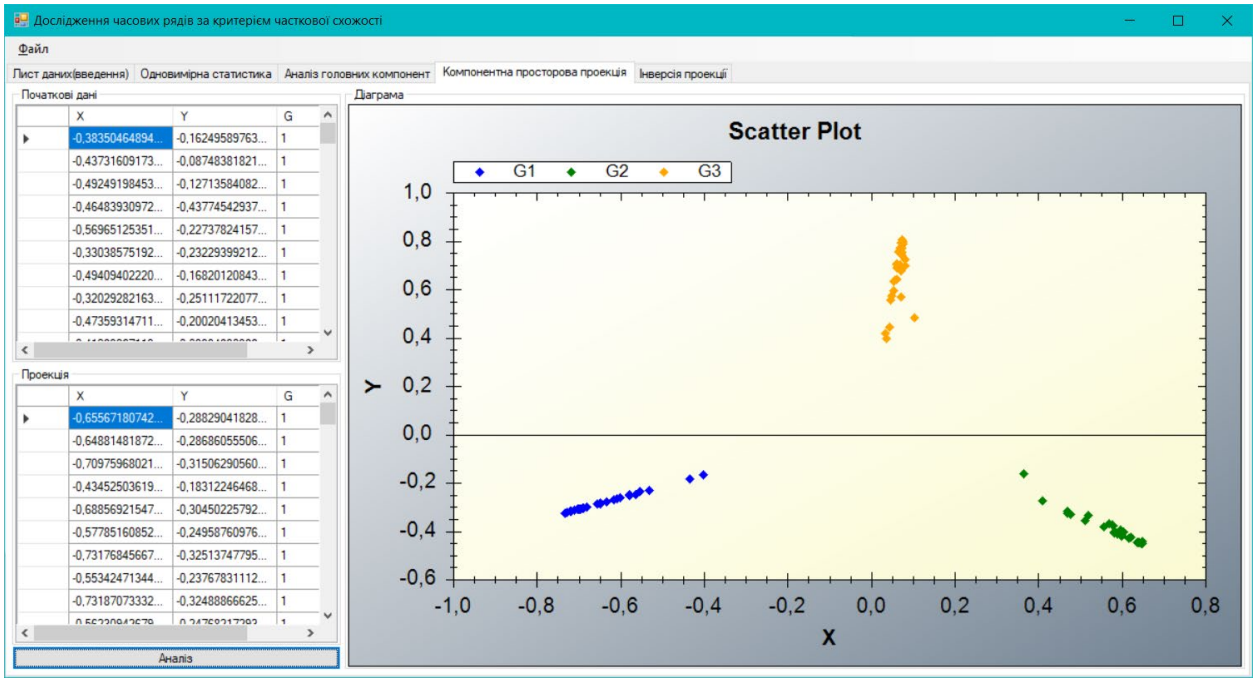


Рисунок 4.13 – Дослідження діаграм розрахунків на тестових даних



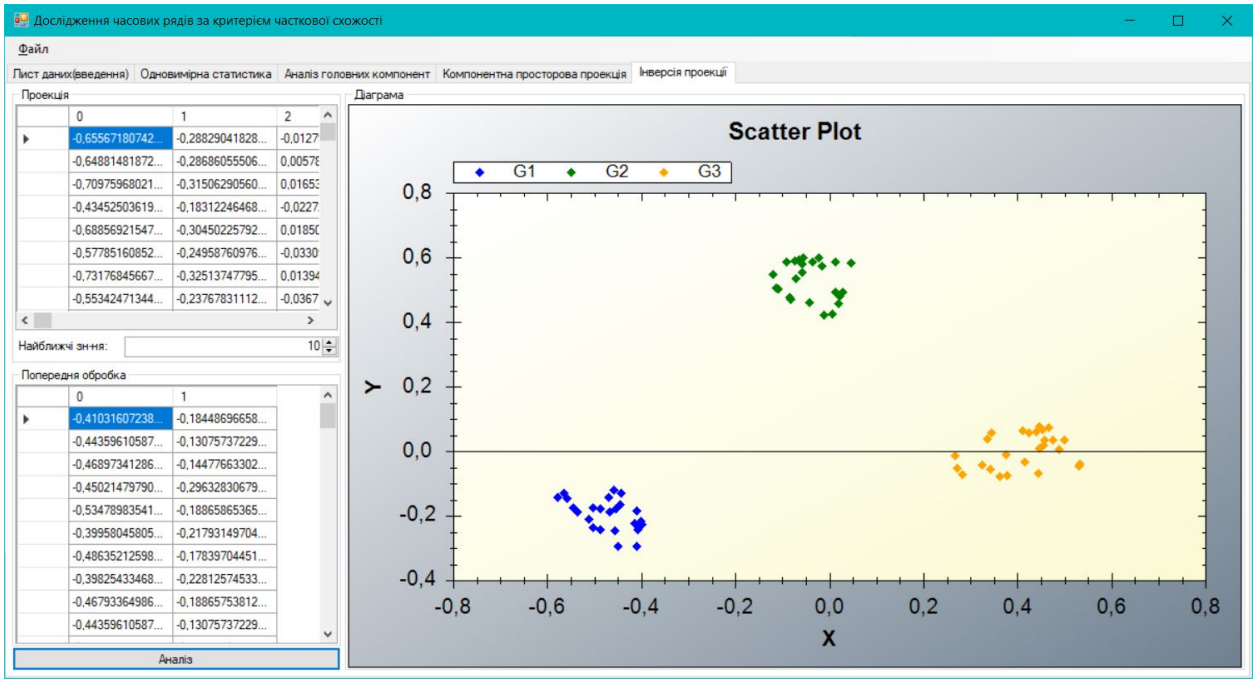


Рисунок 4.14 – Дослідження діаграм розрахунків на тестових даних

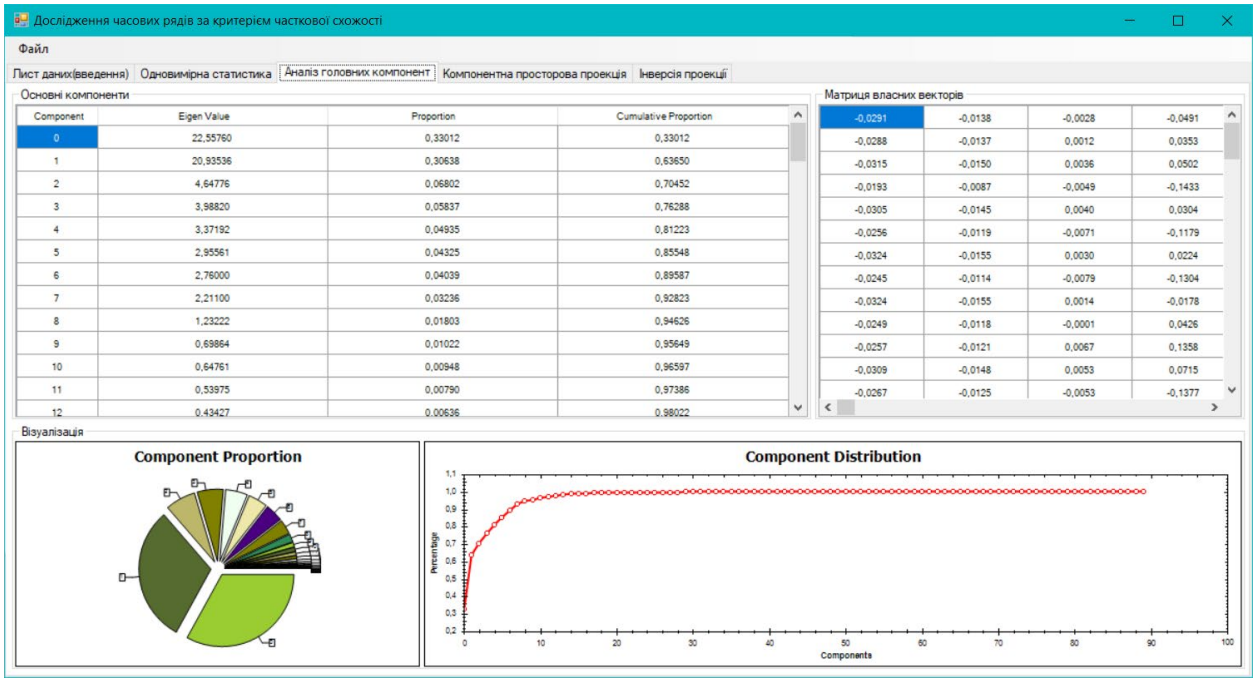


Рисунок 4.15 – Дослідження діаграм розрахунків на тестових даних

4.5 Обробка часових рядів та виведення рішень

Регресійним аналізом називається розділ математичної статистики, що поєднує методи дослідження регресійної залежності між величинами за статистичними даними. В результаті роботи регресійного аналізу можна визначити загальний вид рівняння регресії, виконати розрахунок оцінок невідомих параметрів, що входять до рівняння регресії, та виконати перевірку статистичних гіпотез про отриману регресію [6].

Завдання регресійного аналізу вирішується після виконання кількох кроків.

Аналіз тренду. Тренд - це зміна, що визначає загальний напрямок розвитку, основну тенденцію низки. Часто виділяють лінійний, логарифмічний, поліноміальний та експоненційний тренди.

Не існує "автоматичного" способу виявлення тренду в часовому ряді. Однак якщо тренд є монотонним (стійко зростає або стійко зменшується), аналізувати такий ряд зазвичай неважко. Якщо тимчасові ряди містять значну помилку, то першим кроком виділення тренда є згладжування [7].

Згладжування. Згладжування завжди включає певний спосіб локального усереднення даних, у якому несистематичні компоненти взаємно погашають

одне одного. Найзагальніший метод згладжування - ковзне середнє, в якому кожен член ряду замінюється простим або виваженим середнім n сусідніх членів, де n - ширина "вікна". Замість середнього можна використовувати медіану значень, що потрапили у вікно. Основна перевага медіанного згладжування, у порівнянні зі згладжуванням ковзним середнім, полягає в тому, що результати стають більш стійкими до викидів (всередині вікна). Таким чином, якщо даних є викиди (пов'язані, наприклад, з помилками вимірювань), то згладжування медіаною зазвичай призводить до більш гладким або, принаймні, більш "надійним" кривим, порівняно зі ковзним середнім з тим же самим вікном [30].

Зовнішній вигляд автокореляційної корелограми показано на рис. 4.16.

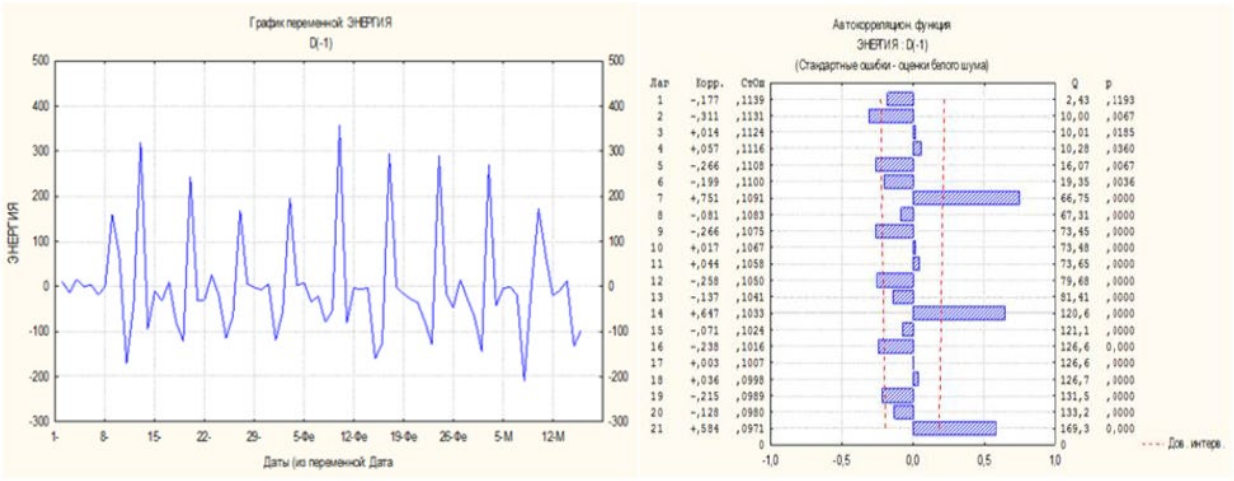


Рисунок 4.16 – Автокореляційна корелограма перетвореного стаціонарного часового ряду

Зовнішній вигляд автокореляційної корелограми показаний на рис. 4.17

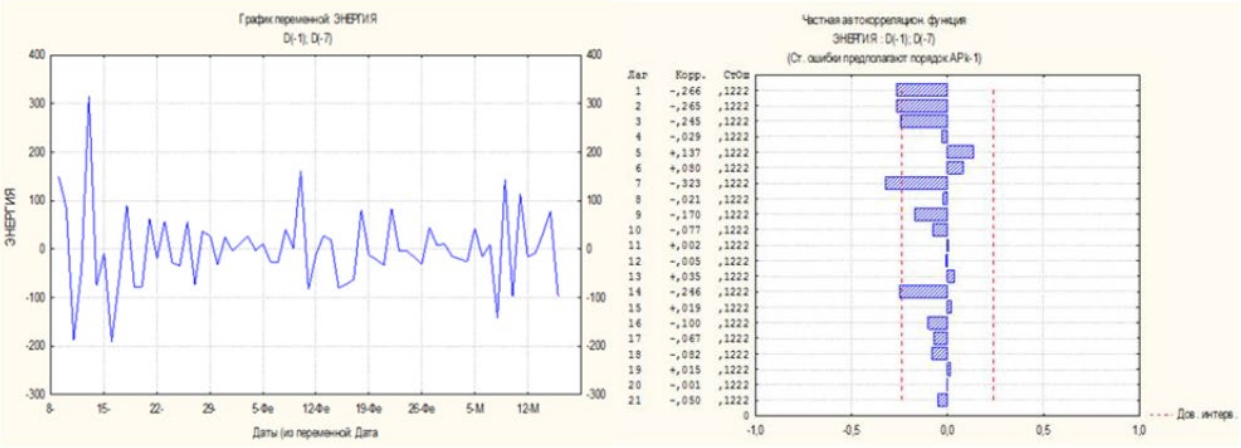


Рисунок 4.17 – Автокореляційна корелограма перетвореного стаціонарного часового ряду

Періодична та сезонна залежність (сезонність) є іншим загальним типом компонент тимчасового ряду. Періодична залежність може бути формально визначена як кореляційна залежність порядку k між кожним і елементом ряду і (ik) елементом. Її можна виміряти за допомогою автокореляції (тобто кореляції між самими членами ряду); k зазвичай називають лагом (іноді використовують еквівалентні терміни: зсув, запізнення). Якщо помилка виміру не надто велика, то сезонність можна визначити візуально, розглядаючи поведінку членів низки через кожні k часових одиниць [47].

Автокореляційна корелограма. Сезонні складові часового ряду можуть бути знайдені за допомогою корелограми. Корелограма (автокорелограма) показує чисельно і графічно автокореляційну функцію (АКФ), тобто коефіцієнти автокореляції (та його стандартні помилки) для послідовності лагів з певного діапазону.

Коефіцієнти приватної кореляції більш високих порядків можна визначити через коефіцієнти приватної кореляції нижчих порядків за наступною формулою рекурентної:

$$r_{yx_i(x_1x_2\ldots x_k)} = \frac{r_{yx_i(x_1x_2\ldots x_{k-1})} - r_{yx_k(x_1x_2\ldots x_{k-1})} r_{x_ix_k(x_1x_2\ldots x_{k-1})}}{\sqrt{\left(1-r^2_{yx_k(x_1x_2\ldots x_{k-1})}\right)\cdot\left(1-r^2_{x_ix_k(x_1x_2\ldots x_{k-1})}\right)}}$$

Коефіцієнти приватної кореляції широко використовуються на стадії формування моделі при відборі факторів.

При побудові багатофакторної моделі застосовується метод виключення змінних, під час якого будується рівняння регресії з повним набором змінних, потім розраховується матриця окремих коефіцієнтів кореляції. Далі перевіряється статистична значущість кожного з коефіцієнтів згідно з t-критерієм Ст'юдента. Незалежна змінна, що має найменшу та несуттєву кореляцію із залежною змінною, виключається. Потім будується нове рівняння регресії, і процедура триває до того часу, доки виявиться, що це приватні коефіцієнти кореляції статистично значимі, тобто істотно від нуля [15].

Перевірка статистичної значущості приватного коефіцієнта кореляції - перевірка гіпотези про те, що він дорівнює нулю

$$\text{H0}: r_{yx_i(x_1x_2\ldots x_k)} = 0\,.$$

Розраховується статистика:

$$t = \frac{r_{yx_i(x_1x_2\ldots x_k)}}{\sqrt{1-(r_{yx_i(x_1x_2\ldots x_k)})^2}} \cdot \sqrt{n-(k+1)}$$

Висновок про значимість приватного коефіцієнта кореляції робиться при$|t|>t_\varepsilon$де t_εвідповідне табличне значення t-розподілу з (n-(k+1)) ступенями свободи.

Розрахуємо парні лінійні коефіцієнти кореляції, застосовуючи формулу і водночас перевіряючи їхню статистичну значимість [14].

$$r_{YX_1} = \frac{n\sum_{i=1}^{n} X_1 Y - \sum_{i=1}^{n} X_1 \sum_{i=1}^{n} Y}{\sqrt{\left[n\sum_{i=1}^{n} X_1^2 - \left(\sum_{i=1}^{n} X_1\right)^2\right]\left[n\sum_{i=1}^{n} Y^2 - \left(\sum_{i=1}^{n} Y\right)^2\right]}} =$$

$$= \frac{20 \cdot 8912{,}57 - 277{,}2 \cdot 454{,}5}{\sqrt{(20 \cdot 5860{,}9 - 76839{,}84) \cdot (20 \cdot 18206{,}89 - 206570{,}3)}} = 0{,}6553,$$

$$t = 0{,}6553 \cdot \sqrt{20-2} \,/\, \sqrt{1-(0{,}6553)^2} = 3{,}68,$$

$$r_{YX_2} = \frac{n\sum_{i=1}^{n} X_2 Y - \sum_{i=1}^{n} X_2 \sum_{i=1}^{n} Y}{\sqrt{\left[n\sum_{i=1}^{n} X_2^2 - \left(\sum_{i=1}^{n} X_2\right)^2\right]\left[n\sum_{i=1}^{n} Y^2 - \left(\sum_{i=1}^{n} Y\right)^2\right]}} =$$

$$= \frac{20 \cdot 908{,}56 - 31{,}8 \cdot 454{,}5}{\sqrt{(20 \cdot 61{,}5 - 1011{,}24) \cdot (20 \cdot 18206{,}89 - 206570{,}3)}} = 0{,}6346,$$

$$t = 0{,}6346 \cdot \sqrt{20-2} \,/\, \sqrt{1-(0{,}6346)^2} = 3{,}60,$$

$$r_{X_1X_2} = \frac{n\sum_{i=1}^{n} X_1 X_2 - \sum_{i=1}^{n} X_1 \sum_{i=1}^{n} X_2}{\sqrt{\left[n\sum_{i=1}^{n} X_1^2 - \left(\sum_{i=1}^{n} X_1\right)^2\right]\left[n\sum_{i=1}^{n} X_2^2 - \left(\sum_{i=1}^{n} X_2\right)^2\right]}} =$$

$$= \frac{20 \cdot 8912{,}57 - 277{,}2 \cdot 31{,}8}{\sqrt{(20 \cdot 5860{,}9 - 76839{,}84) \cdot (20 \cdot 61{,}5 - 1011{,}24)}} = 0{,}1247,$$

$$t = 0{,}1247 \cdot \sqrt{20-2} \,/\, \sqrt{1-(0{,}1247)^2} = 2{,}80.$$

Складемо матрицю парних лінійних коефіцієнтів кореляції (у дужках значення t-статистик):

	y	x_1	x_2
y	1,0	0,6553 (3,68)	0,6346 (3,60)
x_1	0,6553 (3,68)	1,0	0,1247(2,80)
x_2	0,6346(3,60)	0,1247(2,80)	1,0

Розрахуємо коефіцієнти приватної кореляції та перевіримо їх значущість:

$$r_{yx_1(x_2)} = \frac{0{,}6553 - 0{,}6346 \cdot 0{,}1247}{\sqrt{(1-(0{,}6346)^2) \cdot (1-(0{,}1247)^2)}} = 0{,}7513;$$

$$t = \frac{0{,}7513}{\sqrt{1-(0{,}7513)^2}} \cdot \sqrt{20-(2+1)} = 4{,}69,$$

$$r_{yx_2(x_1)} = \frac{0{,}6346 - 0{,}6553 \cdot 0{,}1247}{\sqrt{(1-(0{,}6553)^2) \cdot (1-(0{,}1247)^2)}} = 0{,}7377;$$

$$t = \frac{0{,}7377}{\sqrt{1-(0{,}7377)^2}} \cdot \sqrt{20-(2+1)} = 4{,}51,$$

$$r_{x_1x_2(y)} = \frac{0{,}1247 - 0{,}6553 \cdot 0{,}6346}{\sqrt{(1-(0{,}6553)^2)\cdot(1-(0{,}6346)^2)}} = -0{,}4987;$$

$$t = \frac{-0{,}4987}{\sqrt{1-(-0{,}4987)^2}} \cdot \sqrt{20-(2+1)} = -2{,}37.$$

Складемо матрицю окремих коефіцієнтів кореляції (у дужках значення t-статистик):

	y	x_1	x_2
y	1,0	0,7513 (4,69)	0,7377 (4,51)
x_1	0,7513 (4,69)	1,0	-0,4987 (-2,37)
x_2	0,7377(4,51)	-0,4987 (-2,37)	1,0

Висновки до розділу 4

В результаті виконання розділу була наведена програмна реалізація алгоритму аналізу часових рядів, представлена модульна структура програмного засобу, наведена діаграма потоків даних. Виконано опис інтерфейсу програмного продукту, обробку часових рядів та виведення рішень.

Коефіцієнти кореляції показують " чисту " кореляцію пари змінних, що виключає вплив інших змінних, включених у рівняння. Таким чином, найбільш сильним є взаємозв'язок між вартістю перевезення та вагою вантажу. Однак зауважимо, що приватні коефіцієнти кореляції між у та х1, у та х2 свідчать про сильніші взаємозв'язки незалежних змінних із залежною, ніж це показують значення парних коефіцієнтів кореляції. Це сталося тому, що парний коефіцієнт кореляції завищив тісноту зв'язку між х1 та х2, занизивши при цьому тісноту зв'язку між у та х1, у та х2. Зазначимо також, що це приватні коефіцієнти кореляції статистично значущі.∇.

Остаточні висновки

Дослідження процедури агрегування критеріїв при дослідженні часових рядів за критерієм часткової схожості показало, що результати агрегування сильно залежать від знань експертів та переваг методів дослідження статистичних показників. Навіть якщо завдання вибору вирішується однією людиною, то й у цьому випадку на різних етапах процедури агрегування критеріїв результати можуть більшою чи меншою мірою відрізнятись один від одного залежно від суджень людини, як обумовлених поглядом на проблему з різних сторін, так і продиктованих життєвим досвідом.

У процесі виконання роботи з'ясувалося, що ступінь агрегування вихідних показників впливає на трудомісткість та пояснення результатів у завданні вибору. Чим більше буде підсумкових критеріїв, тим складніше зробити правильний вибір та пояснити отримані результати. З іншого боку, занадто мала кількість підсумкових критеріїв у сукупності з неправильним агрегуванням показників на попередніх рівнях ієрархії може призвести до суперечностей, таким, що всі запропоновані варіанти при своїй очевидній відмінності можуть мати однакові оцінки за результатами порівняння.

Традиційні математичні моделі не можуть забезпечити необхідну точність прогнозування фінансових часових рядів, які характеризуються великою ступенем випадковості. З цієї причини особливу вагу набуває розробка моделей та методів, призначених для передбачення знаків приростів у таких часових рядах. Дослідження у цьому напрямку спрямоване на створення методу, який здатен здійснювати прогноз знаку приросту часового ряду з максимальною точністю. При цьому використовуються засоби фрактального аналізу інформації та підходи інтелектуального аналізу часових рядів, такі як індексація за методами найближчого сусіда і K-найближчих сусідів та інші. Це дослідження не тільки має значення для розвитку науки, але і має практичне застосування в області фінансового прогнозування, забезпечуючи ефективні і точні результати для управління фінансовими ризиками та прийняття рішень.

Бібліографічний список

1. Boser, B., Guyon I., Vapnik, V. A training algorithm for optimal margin classifiers. In D. Haussler, editor, proceedings of the 5th Annual ACM Workshop on COLT, pp. 144-152, Pittsburgh, 1992.

2. Breiman, L. Bagging predictors. Machine Learning, Vol. 24 (2), pp. 123- 140, 1996.

3. Shoghian, Sh., Kouzehgar, M. A Comparison among Wolf Pack Search and Four other Optimization Algorithms. Wvrld Academy of Science, Engineering & Technology, Vol. 6, Issue 72, pp. 418-423, 2012.

4. Zahadat, P., Schmickl, T. Wolfpack-inspired evolutionary algorithm and a reaction-diffusion-based controller are used for pattern formation. In proceedings of the 2014 Conference on Genetic and Evolutionary Computation (GECCO‘2014), pp. 241-248, 2014.

5. Lia, C.-M., Duc, Y.-C., Wua, J.-X., Lind, C.-H., Hoe, Y.-R., Lina, Y., Chen, T. Synchronizing chaotification with support vector machine and wolf pack search algorithm for estimation of peripheral vascular occlusion in diabetes mellitus. Biomedical Signal Processing and Control, Vol. 9, pp. 45-55, 2014.

6. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. / Анісімов А.В., Кулябко П.П. − Київ. – 2017. – 110 с.

7. Антоненко В. М. Сучасні інформаційні системи і технології: управління знаннями : навч. Посібник / В. М. Антоненко, С. Д. Мамченко, Ю. В. Рогушина. – Ірпінь : Нац. університет ДПС України, 2016. – 212 с.

8. Бенгфорт Б. Прикладний аналіз текстових даних на Python. Машинне навчання та створення додатків обробки природної мови / Б. Бенгфорт // СПб .: Питер, 2019. - 368 с.

9. Буйницька О. Інформаційні технології та технічні засоби навчання. Київ : Центр навчальної літератури, 2019. 240 с.

10. Бахрушин В.Є. Методи аналізу даних: навчальний посібник для студентів. – Запоріжжя: КПУ, 2011. – 268 с.

11. Данильченко О.М., Данильченко А.О. Інтелектуальний аналіз даних: Навч. посібник. – Житомир: ЖДТУ, 2009. – 405 с.

12. Березовська Л., Кириченко А. Розвиток електронної комерції в Україні та ЄС.

13. Економіка та суспільство. 2022. № 42. URL: тhttps://economyandsociety.in.ua/index.php/journal/article/view/1614/1551.

14. Богуш В. М., Богуш В. В., Бровко В. Д., Настрадін В. П. Основи кіберпростору, кібербезпеки та кіберзахисту : навч. посіб. / [під ред. В. М. Богуша]. Київ : Ліра-К, 2020. 552 с.

15. Борейко Н. М. Специфіка електронної комерції у вітчизняному сегменті мережі Інтернет. Бізнес-навігатор. 2020. № 2. С. 87–93. URL: http://www.businessnavigator.ks.ua/journals/2020/58_2020/18.pdf.

16. Буров Є. В. Комп'ютерні мережі : підручник. Львів : Магнолія 2006, 2020.262 с.

17. Варіс І. О., Кравчук О. І., Завгородня С. А. Цифрова трансформація бізнесу:вибір, впровадження та вдосконалення CRM-систем. Маркетинг і цифровітехнології. 2021. Т. 5. № 2. С. 48–66. URL: http://mdtopu.com.ua/index.php/mdt/article/view/139/124.

Використання Веб 2.0 технологій на уроках природничих дисциплін : метод. посібник / О. Кожемякіна та ін. Краматорськ : б.в., 2020. 102 с.

18. Войтович Н. В., Найдьонова А. В. Використання хмарних технологій Google та сервісів Web 2.0 в освітньому процесі : метод. рекоменд. Дніпро : ДПТНЗ «Дніпровський центр ПТОТС», 2017. 113 с.

19. Гущіна Н. І. Путівник світом цифрових технологій : посібник для вчителів. Київ : Видавничий центр «Освіта», 2018. 32 с.

20. Пшенична О. С. Інформатика та програмування: засоби і технології обробки інформації : методичні рекомендації до лабораторних занять для здобувачів ступеня вищої освіти бакалавра спеціальності «Середня освіта»,

освітньо-професійних програм «Середня освіта (Фізика)», «Середня освіта (Математика)», «Середня освіта (Інформатика)». Запоріжжя : ЗНУ, 2019. 137 с.

21. Литвинова С. Г. Спірін О. М., Анікіна Л. П. Хмарні сервіси Office 365 : навчальний посібник. Київ : Компринт, 2015. 170 с.

22. Носенко Ю. Г., Попель М. В., Шишкіна М. П. Хмарні сервіси і технології у науковій і педагогічній діяльності : методичні рекомендації / за ред. М. П. Шишкіної. Київ : ІІТЗН НАПН України, 2016. 73 с.

23. Пшенична О. С. Інформаційні технології у вищій школі : методичні рекомендації до лабораторних занять для здобувачів ступеня вищої освіти магістра спеціальності «Комп'ютерні науки». Запоріжжя : ЗНУ, 2020. 99 с.

24. Redecker C. European Framework for the Digital Competence of EducatorsDigital Competence Framework for Educators (DigCompEdu) : report. Luxembourg: Publications Office of the European Union, 2017. 95 p.

25. Воронін А. М. Інформаційні системи прийняття рішень: навчальний посібник. / Воронін А. М., Зіатдінов Ю. К., Климова А. С. − К. : НАУ-друк, 2009. − 136с.

26. Галузинський Г. П. Інформаційні системи у бізнесі. Практикум для індивідуальної роботи: навчметод. посіб. для самост. вивч. Дисципліни. / Галузинський Г. П., Денісова О. О., Писаревська Т. А. − К. : КНЕУ, 2008. − 524с.

27. Годун В.М. Інформаційні системи і технології в статистиці: навч. посіб. / В.М. Годун, Н.С. Орленко, М. А. Сендзюк; за ред. В.Ф. Ситника. – К.: КНЕУ, 2003. – 267 с.

28. Грицунов О. В. Інформаційні системи та технології: навч. посіб. для студентів за напрямом підготовки «Транспортнітехнології» / О. В. Грицунов; Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ, 2010. – 222 с.

29. Інформаційні системи в економіці : навч. посібник / Пономаренко В. С., Золотарьова І. О., Бутова Р. К. та ін. – Х. : Вид. ХНЕУ, 2011. – 176 с.

30. Інформаційні системи в промисловості : навчальний посібник / Л. О. Добровольська, О. О. Черевко. – Маріуполь : ПДТУ, 2014. – 238 с.

31. Інформаційні системи в сучасному бізнесі : навчальний посібник / В. С. Пономаренко, І. О. Золотарьова, Р. К. Бутова та ін. – Х. : Вид. ХНЕУ, 2011. – 484 с.

32. Ситник В.Ф., Орленко Н.С. Імітаційне моделювання: Навч. посібник. – К.: КНЕУ, 1998. – 232 с.

33. Кравець І.О. Імітаційне моделювання: Навч. посібник. – ЧДУ ім. Петра Могили, 2010.- 107 с.

34. Томашевський В.М. Моделювання систем. – К.:Видавнича група BHV,2005. - 351с.

35. Коробова М.В. Основи математичного моделювання економічних, екологічних та соціальних процесів/ М.В. Коробо- ва, І.М. Ляшенко, А.М. Столяр. – Тернопіль: “Навчальна книга – Богдан”, 2006. – 304 с.

36. Обушна О.М. Навчально-методичний комплекс з дисципліни «Імітаційне моделювання» для студентів спеціальності «Економічна кібернетика».- К.: РВВ ІМФ, 2003. – 102 с.

37. Ситник В.Ф. Імітаційне моделювання: Навч.-метод. посібник для самост. вивч. дисц/ В.Ф. Ситник, Н.С. Орленко – К.: КНЕУ, 2019. – 208 с.

38. Лужковський А. Г. Теорія масового обслуговування в телекомунікаціях / А.Г. Лужковський. – Одеса : ОНАЗ ім. О.С. Попова, 2010. – 112 с.: іл.

39. Нейман В., Параскун А. Використання сучасних програм у бізнес-плануванні. Економіст. 2021. № 12. С. 60–61.

40. Нікітін Ю. О., Кульчицький О. І. Цифрова парадигма як основа визначень: цифровий бізнес, цифрове підприємство, цифрова трансформація. Маркетинг і цифрові технології. 2019. Т. 3. № 4. С. 77–87. URL: http://mdtopu.com.ua/index.php/mdt/article/view/86/83.

41. Новаківський І. І., Грибик І. І., Смолінська Н. В. Інформаційні системи в менеджменті: адаптивний підхід : підручник. Київ : Кондор, 2019. 440 с.

42. Приймак В.І. Математичні методи економічного аналізу: навч. посіб. – К.: Центр учбової літератури, 2009. – 296 с.

43. Сявавка М.С., Рибицька О.М. Математичне моделювання за умов невизначеності. – Львів: НВФ «Українські технології», 2000 – 320 с.

44. Вергунова І.М. Системне моделювання в економіці. – 2016. Ел. ресурс. Режим доступу: http://cyb.univ.kiev.ua/library.school-guides.html.

45. Вергунова І.М. Системне моделювання в економіці. – 2013. Ел. ресурс. Режим доступу: http://mi.unicyb.kiev.ua/?page_id=56&lang=ua.

46. Інформаційні системи і технології в банківській сфері: навч. посіб. для студ. спец. 6.050105 "Банківські справи". / Аніловська Г. Я., Чуй І. Р., Вус М. Л., Стоколоса Т. М. − Л. : ЛКА, 2008. – 332 с.

47. Ніколенко С. Глибоке навчання / С. Ніколенко, А. Кадурін, Е. Архангельська // СПб .: Питер, 2018. - 480 с.

48. Плас Д. Python для складних завдань. Наука про дані і машинне навчання. Керівництво / Плас Джейк Вандер // К .: BMC, 2018. - 759 с.

49. Редько В.Г. Еволюція, нейронні мережі, інтелект: Моделі і концепції еволюційної кібернетики / В.Г. Редько // М .: Ленанд, 2019. - 224 с.

50. Хайкін С. Нейронні мережі: повний курс / С. Хайкін // М .: Діалектика, 2019. - 1104 с.

51. Програмування числових методів мовою PYTHON / А. Ю. Дорошенко [та ін.]; за ред. А. В. Анісімова. – ВПЦ "Київський університет", 2013. – 464 с.

52. Шаховська Н. Б. Алгоритми та структури даних / Н. Б. Шаховська, Р.О. Голощук. – Львів : Магнолія-2006. – 2009. – 216 с.

53. Шеховцов В.А. Операційні системи / В.А. Шеховцов. – К. : Видавнича група BHV, 2005. – 576 с.

54. Мнушка О.В., Савченко В.М. Конспект лекцій з дисципліни «Об’єктно-орієнтоване програмування для студентів напрямів підготовки 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки». – Харків, ХНАДУ, 2020.

55. Data model. – URL: https://docs.python.org/3.8/reference/datamodel.html

56. PEP 0 – Index of Python Enhancement Proposals (PEPs).– URL: https://www.python.org/dev/peps/

57. Links to Python related information in Ukranian. – URL: https://wiki.python.org/moin/UkranianLanguage

58. Stackoverflow. – URL:https://stackoverflow.com/

59. Python Software Foundation. The Python Tutorial [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/tutorial/index.html

60. Python Software Foundation. Python 3.7.12 documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3.7/

61. Lutz M. Learning Python, 5th Edition. – O'Reilly Media Inc., 2013. – 1648 p.

62. Chun Wesley J. Core Python Application Programming. Third Edition. – Pearson Education, Inc., 2012.

63. Lutz M. Programming Python, Forth Edition. – O'Reilly Media Inc., 2011.

64. Prometheus: CS50. Вебпрограмування з Python та JavaScript CS50. – Prometheus. – 2021. [Електронний ресурс] – Режим доступу:

65. https://courses.prometheus.org.ua/courses/coursev1:Prometheus+CS50+2021_T1/about

66. Сороко Н.В. Інтеграція сучасних інформаційно-комунікаційних технологій у навчальний процес: зарубіжний та вітчизняний досвід / Наукові записки. – Випуск 77. - Серія: Педагогічні науки. - Кіровоград: РВВ КДПУ ім . В. Винниченка. - 2008. Частина 1. - 354 с., с. 113 - 118.

67. Хмельов О. Г., Хмельова А. В., Інформаційні технології і засоби навчання. — 2009. — №5 (13). — [Електронний ресурс]: сайт Інституту інформаційних технологій і засобів навчання НАПН України. — Режим доступу: http://www.ime.eduua.net/em.html.

ДОДАТКИ

ДОДАТОК А

Технічне завдання

ЗАТВЕРДЖУЮ

Проректор
Українського державного університету науки і технології
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЧАСОВИХ РЯДІВ ЗА КРИТЕРІЄМ ЧАСТКОВОЇ СХОЖОСТІ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1329-01-ЛЗ

Завідувач кафедри КІТ

_____Вадим ГОРЯЧКІН

Керівник розробки

_____Віктор ШИНКАРЕНКО

Виконавець

_____Тімур КДИРОВ

Нормоконтролер

_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

44165850.1329-01

ДОСЛІДЖЕННЯ ЧАСОВИХ РЯДІВ ЗА КРИТЕРІЄМ ЧАСТКОВОЇ СХОЖОСТІ

Технічне завдання

44165850.1329-01-ЛЗ

Листів 13

2024

ЗМІСТ

ВСТУП

Завданням програми є створення інтуїтивного інтерфейсу для завантаження, обробки та відображення часових рядів у вигляді графіків. Основний функціонал включає в себе аналіз схожості між різними часовими рядами, враховуючи часткові аспекти, такі як форма, амплітуда та інші важливі характеристики.

Програма повинна надавати можливості вибору параметрів аналізу, таких як величина вікна часового інтервалу, метод вимірювання схожості, а також інші налаштування. Додатково, вона може включати в себе функціонал для виявлення аномалій у часових рядах та надавати зручні засоби для користувачів для взаємодії з візуалізованими даними, такі як збільшення, зменшення, експорт графіків тощо.

Мета програмного забезпечення - спростити процес вивчення та аналізу часових рядів, дозволяючи користувачам легко виявляти та розуміти часткову схожість між різними даними. Також, програма може забезпечити можливість збереження результатів аналізу для подальшого використання та обміну даними.

Даний продукт призначений для використання науковцями для побудови різних моделей прогнозування та аналізу даних.

1 ПІДСТАВИ ДО РОЗРОБКИ

Підставою для розробки є наказ від 21 серпня 2023 р. ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем магістерських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

При математичній обробці масивів експериментальної інформації виникає необхідність у підборі емпіричних формул, що встановлюють зв'язок одного виміряного параметра з іншим.

Завдання визначення точного виду виявленої взаємозалежності параметрів вирішується за допомогою регресійного аналізу.

Традиційні математичні моделі не є достатньо ефективними для точного прогнозування фінансових часових рядів, які мають випадковий характер коливань. Тому виникає необхідність розробки моделей та методів, спрямованих на прогнозування знаків приростів у таких рядах. Дослідження в цьому напрямку спрямоване на створення методу, який забезпечив би максимальну точність прогнозу знаку приросту часового ряду. Для цього можуть використовуватися методи фрактального аналізу інформації, інтелектуального аналізу часових рядів, такі як індексація за методами найближчого сусіда та K-найближчих сусідів тощо.

Мета проєкту:

Розробка програмного забезпечення для аналізу та дослідження часових рядів з метою виявлення трендів та прогнозування майбутніх значень.

Експлуатаційне призначення програми:

створення більш точного засобу аналізу статистичних даних;

збільшення продуктивності при оцінці регресійних характеристик набору даних, візуалізація числових характеристик, вибір експериментальних даних для аналізу.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Програма повинна надавати можливість:

- програмні методи завантаження даних з фізичного файлу;
- фіксація змін даних після редагування користувачем окремих значень полів;
- інтерфейсна частина, з наданням користувачу елементів управління основними режимами роботи з програмою;
- отримати статистичні дані процесу моделювання;
- підтримка різних форматів даних, таких як CSV, Excel, JSON;
- фільтрація та очищення вхідних даних від аномалій та викидів;
- нормалізація даних для забезпечення однорідності та правильної інтерпретації;
- побудова графіків та діаграм для візуалізації часових рядів;
- розрахунок основних статистичних параметрів (середнє, медіана, стандартне відхилення);
- використання алгоритмів прогнозування для побудови моделей на основі історичних даних.

3.2 Вимоги до надійності

Програма має забезпечити стійку роботу, коректне виконання своїх основних функцій та цілісність і збереженість даних. Повинні виконуватися наступні вимоги:

- користувачу повинно повідомлятися про некоректний вибір набору даних;
- час відновлення після відмови, що була викликана несправністю технічних засобів, не повинно перевищувати часу, який необхідний для усунення недоліків технічних засобів і переустановлення програмних засобів;
- кількість помилок не повинна перевищувати однієї на 1000 операцій;

– архівна копія тексту програми повинна зберігатися на зовнішньому носії.

3.3 Умови експлуатації

Для забезпечення сталого функціонування програми користувачеві і програмісту необхідно дотримуватися таких умов:

– програма повинна використовуватись у приміщеннях, які відповідають умовам роботи ЕОМ [1];
– кваліфікація робочого персоналу з даною програмою повинна бути на рівні досвідченого користувача ЕОМ, операційною системою Windows. Робітник повинен бути ознайомлений з керівництвом користувача;
– програмний комплекс повинен використовуватись в приміщеннях, з наступними кліматичними умовами: температура 21-25 °С, відносна вологість повітря 40-60%

3.4 Вимоги до складу і параметрів технічних засобів

Програма, що розробляється, розрахований на використання на IBM-сумісних персональних комп’ютерах, що мають наступні характеристики:

– процесор з тактовою частотою не нижче 1,6 ГГц;
– не менше 512 МБ (для 32-розрядної системи) або 1024 МБ (для 64-розрядної системи) оперативної пам’яті;
– відеоадаптер з підтримкою OpenGL 3.0 та відеопам’яттю не менше 256 МБ;
– не менше 512 МБ (для 32-розрядної системи) або 1 ГБ (для 64-розрядної системи) простору на жорсткому диску;
– монітор з роздільною здатністю екрану 1024x768 та більше, з підтримкою не менш ніж 16,7 млн. кольорів;
– клавіатура, маніпулятор миша;
– наявність CD/DVD приводу, USB роз’єму або LAN-адаптеру для встановлення необхідного ПЗ.

3.5 Вимоги до інформаційної і програмної сумісності

Для функціонування програмного продуту необхідні:

- програма розрахована для роботи у будь-якій операційній системі Windows, де встановлений .NET Framework 4.8;
- встановлений Microsoft .Net Framework 4.8 і вище.

3.6 Вимоги до маркування і упаковки

Програма може зберігатися на змінних носіях (CD/DVD-диски). Упаковка програми повинна забезпечувати захист від механічних пошкоджень.

Упаковка повинна мати маркування: назва програмного продукту, розробник, контакти, рік. На рис. 3.1 приведений приклад маркування.

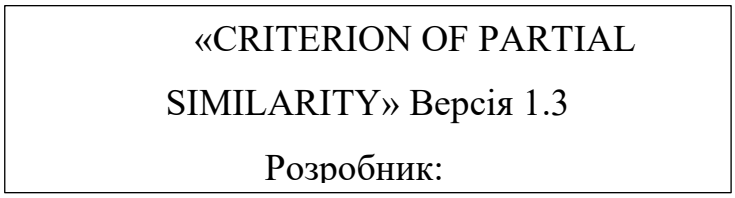


Рисунок 3.1 – Маркування

3.7 Вимоги до транспортування і зберігання

Умови транспортування та зберігання повинні забезпечувати захист носія від фізичних пошкоджень. Транспортування та зберігання передбачається на змінних носіях (CD/DVD-диски) або портативних пристроях (флешки, тощо).

Строк зберігання програми залежить від носія інформації. Потрібно кожний місяць перевіряти стан носія і при необхідності – робити резервну копію.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації повинні входити:

– керівництво користувача;

– текст програми.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів (ГОСТ 19.101-77) [1].

5 ЕКОНОМІЧНІ ПОКАЗНИКИ

Показники та їх розрахунок не були описані у документах бо розробка додатку не є комерційною.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки програми приведені в табл. 6.1.

Таблиця 6.1 Стадії та етапи розробки

Стадії розробки	Етапи розробки	Терміни виконання
Підготовчий етап	Збір вимог та уточнення специфікації	04.09.2023 — 06.09.2023
Розробка	Реалізація основних функцій програми	06.09.2023 — 10.09.2023
Тестування та виправлення помилок	Відлагодження програми	10.09.2023 — 18.09.2023
Документація	Підготовка технічної документації та користувацької інструкції	18.09.2023 — 22.09.2023
Завершальний етап	Завершення всіх заходів з розробки та виправлення останніх помилок	22.09.2023 — 28.09.2023

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙОМУ

Контроль за виконанням роботи здійснює керівник розробки доц. Шинкаренко В. І.

Прийом здійснюється комісією у складі, визначеному університетом.

8 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

ДОДАТОК Б

Керівництво користувача

ЗАТВЕРДЖУЮ
Проректор
Українського державного
університету науки і технології
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЧАСОВИХ РЯДІВ ЗА КРИТЕРІЄМ ЧАСТКОВОЇ СХОЖОСТІ

Керівництво користувача

ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1329-01 ІЗ 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Віктор ШИНКАРЕНКО
Виконавець
_____Тімур КДИРОВ
Нормоконтролер
_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

44165850.1329-01 ІЗ 01

ДОСЛІДЖЕННЯ ЧАСОВИХ РЯДІВ ЗА КРИТЕРІЄМ ЧАСТКОВОЇ СХОЖОСТІ

Керівництво користувача
44165850.1329-01 ІЗ 01

Листів 14

2024

АНОТАЦІЯ

Документ 44165850.95107-01 І3 01 «Дослідження часових рядів за критерієм часткової схожості. Опис програми» входить до складу програмної документації на програму, що реалізовує дослідження часових рядів за критерієм часткової схожості, які описуються відповідними математичними моделями та алгоритмами аналізу часових рядів

У даному документі представлено керівництво користувача. Програми написані на мові C#. Об'єм пам'яті, що займають програми комплексу, складає 500 МБ (з врахуванням бібліотеки .NET). Конфігурація комп'ютера стандартна. Комплекс функціонує в середовищі MS WINDOWS 10.

ЗМІСТ

ВСТУП

Завданням програми є створення інтуїтивного інтерфейсу для завантаження, обробки та відображення часових рядів у вигляді графіків. Основний функціонал включає в себе аналіз схожості між різними часовими рядами, враховуючи часткові аспекти, такі як форма, амплітуда та інші важливі характеристики.

Програма повинна надавати можливості вибору параметрів аналізу, таких як величина вікна часового інтервалу, метод вимірювання схожості, а також інші налаштування. Додатково, вона може включати в себе функціонал для виявлення аномалій у часових рядах та надавати зручні засоби для користувачів для взаємодії з візуалізованими даними, такі як збільшення, зменшення, експорт графіків тощо.

Мета програмного забезпечення - спростити процес вивчення та аналізу часових рядів, дозволяючи користувачам легко виявляти та розуміти часткову схожість між різними даними. Також, програма може забезпечити можливість збереження результатів аналізу для подальшого використання та обміну даними.

Не вимагає ознайомлення із будь-якою додатковою експлуатаційною документацією.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

При математичній обробці масивів експериментальної інформації виникає необхідність у підборі емпіричних формул, що встановлюють зв'язок одного виміряного параметра з іншим.

Завдання визначення точного виду виявленої взаємозалежності параметрів вирішується за допомогою регресійного аналізу.

Мета проєкту:

Розробка програмного забезпечення для аналізу та дослідження часових рядів з метою виявлення трендів та прогнозування майбутніх значень.

Експлуатаційне призначення програми:

- створення більш точного засобу аналізу статистичних даних;
- збільшення продуктивності при оцінці регресійних характеристик набору даних, візуалізація числових характеристик, вибір експериментальних даних для аналізу.

Для забезпечення сталого функціонування програми користувачеві і програмісту необхідно дотримуватися таких умов:

- програма повинна використовуватись у приміщеннях, які відповідають умовам роботи ЕОМ [1];
- кваліфікація робочого персоналу з даною програмою повинна бути на рівні досвідченого користувача ЕОМ, операційною системою Windows. Робітник повинен бути ознайомлений з керівництвом користувача;
- програмний комплекс повинен використовуватись в приміщеннях, з наступними кліматичними умовами: температура 21-25 °С, відносна вологість повітря 40-60%.

Програма, що розробляється, розрахований на використання на IBM-сумісних персональних комп’ютерах, що мають наступні характеристики:

- процесор з тактовою частотою не нижче 1,6 ГГц;
- не менше 512 МБ (для 32-розрядної системи) або 1024 МБ (для 64-

розрядної системи) оперативної пам’яті;

- відеоадаптер з підтримкою OpenGL 3.0 та відеопам’яттю не менше 256 МБ;
- не менше 512 МБ (для 32-розрядної системи) або 1 ГБ (для 64-розрядної системи) простору на жорсткому диску;
- монітор з роздільною здатністю екрану 1024x768 та більше, з підтримкою не менш ніж 16,7 млн. кольорів;
- клавіатура, маніпулятор миша;
- наявність CD/DVD приводу, USB роз’єму або LAN-адаптеру для встановлення необхідного ПЗ.

Для функціонування програмного продуту необхідні:

- програма розрахована для роботи у будь-якій операційній системі, де встановлений .NET Framework 4.8;
- встановлений Microsoft .Net Framework 4.8 і вище.

2 ПІДГОТОВКА ДО РОБОТИ

Для початку роботи програмного засобу його необхідно запустити за допомогою пускового файлу «Components.exe».

3 ОПИС ОПЕРАЦІЙ

Після запуску програмного засобу «C:\Users\User\Desktop\Components «Components.exe» на екрані з'являється меню (рис.4.1).

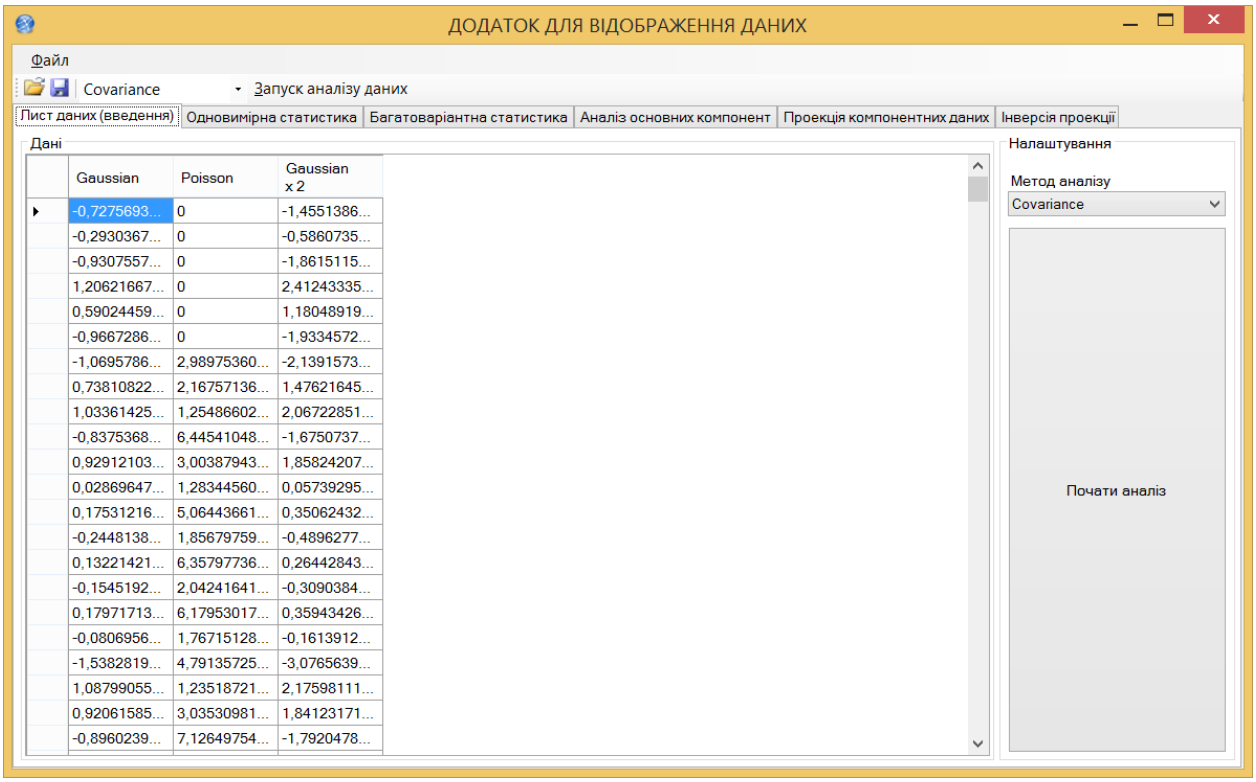


Рисунок 4.1 – Головне меню

У підсистемі оцінки набору даних кількісних показників статистичних даних представлений механізм статистичної оцінки вхідних параметрів, що імпортуються за допомогою формату Excel.

Після відкриття достатньо натиснути кнопку «Запуск аналізу». Результати визначення статистичних показників формуються автоматично та розміщуються у вкладках головного вікна.

Основні режими роботи:

– Побудова гістограми розподілу
– Розрахунок багатоваріантної статистики
– Виконання проекції компонентних даних
– Дослідження діаграм розрахунків на тестових даних

Основні дії в послідовності, що вимагається – користувач натискає на компонент кнопку. В залежності від вибору кнопки користувачем, буде викликана відповідна функція.

Заключні дії – викликання необхідної функції, яка описана в обробнику подій.

Необхідні ресурси для виконання операції – подія натискання на пункт вкладки.

Для завершення виконання аналіз в програмі необхідно обрати останній пункт меню на вкладці «Інверсія проекції» (рис. 4.2).

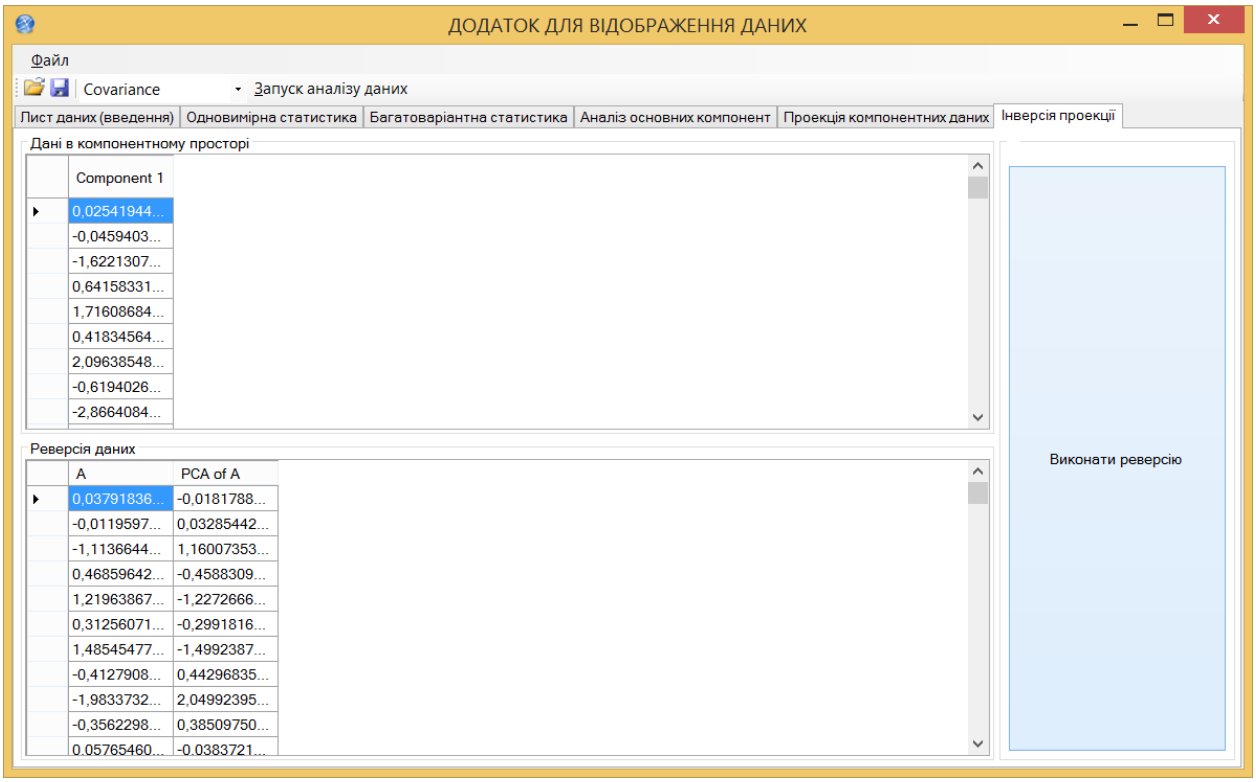


Рисунок 4.2 – Пункт меню «Інверсія проекції»

4 АВАРІЙНІ СИТУАЦІЇ

Якщо програмний засіб буде запускатися з пристрою який не відповідає необхідним технічним характеристикам буде виведено необхідне повідомлення та викликане завершення роботи програмного засобу.

Якщо користувач зачинить поточний екран розрахунків наборів даних, він буде повернений до попереднього екрану меню програмного засобу.

Якщо програмний засіб під час роботи буде поводити некоректно чи із помилкою, від користувача потребується перезапустити програмний засіб для відновлення коректної роботи програмного засобу.

5 РЕКОМЕНДАЦІЇ ЩОДО ЗАСТОСУВАННЯ

Після встановлення програмного засобу на комп’ютер, для його запуску необхідно натиснути на «.exe» файл.

У підсистемі оцінки набору даних кількісних показників статистичних даних представлений механізм статистичної оцінки вхідних параметрів, що імпортуються за допомогою формату Excel. Статистичні дані для виконання аналізу розташовані в табличному файлі, із сукупністю листів, які можна обирати окремо в програмі:

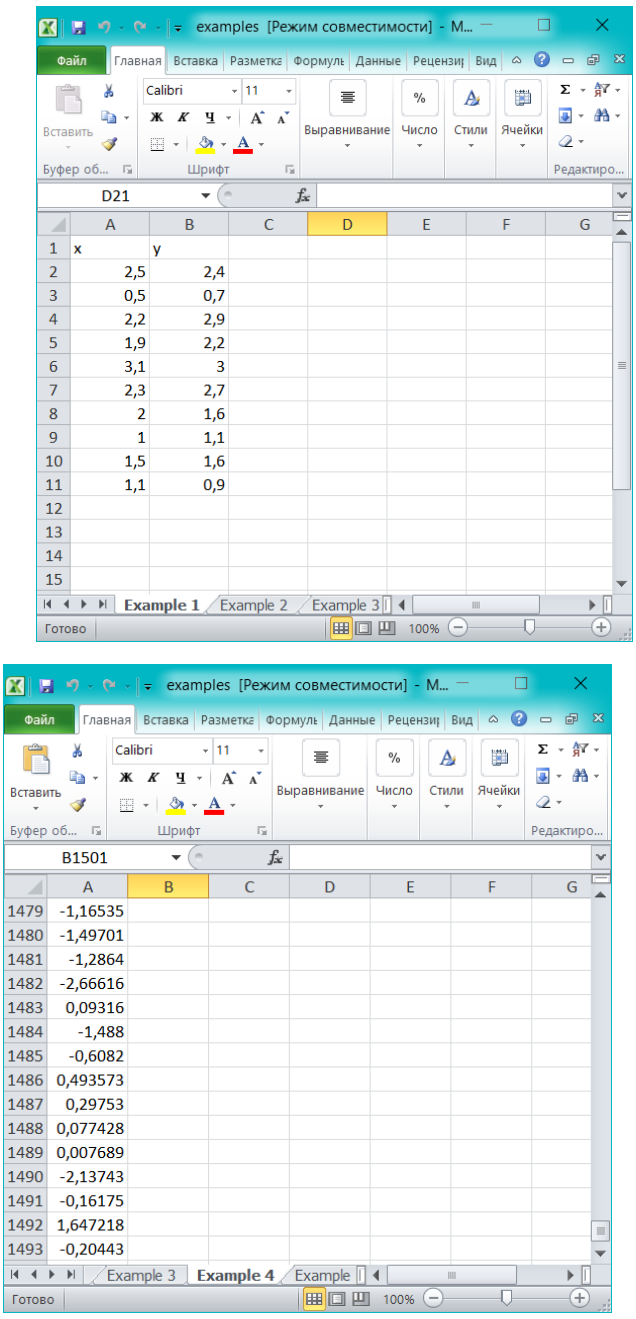


Рисунок 5.1 – Вміст файлу даних

Вибрав лист із даними, слід вибрати напрямок аналізу (Covariance або Covariation) та натиснути на кнопку «Почати аналіз». Результати обробки автоматично оновлюються у вкладках:

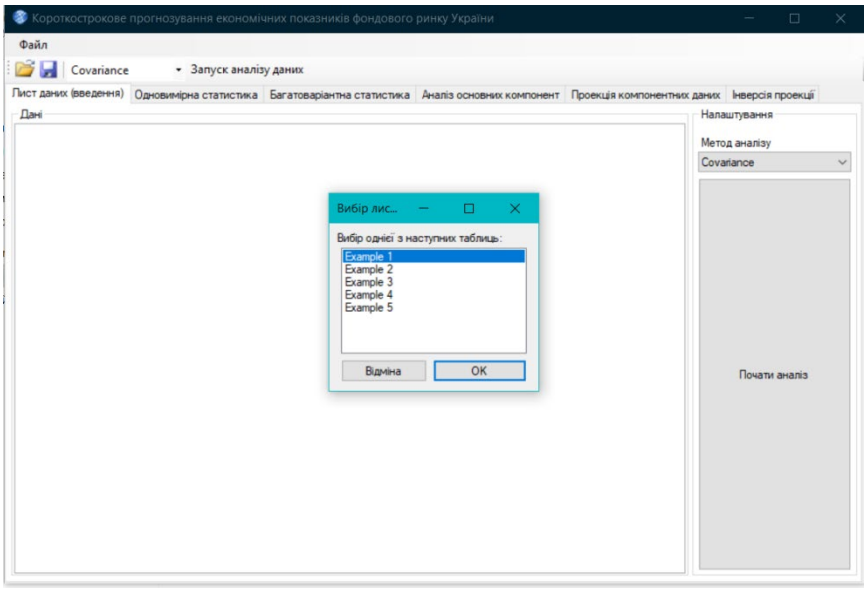


Рисунок 5.2 – Відкриття файлу даних

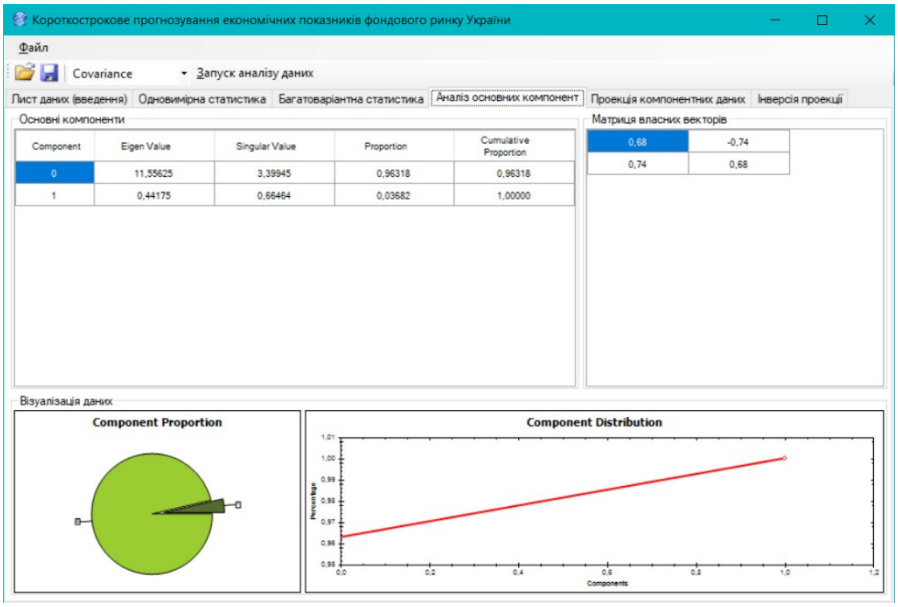


Рисунок 5.3 – Виконання аналізу даних

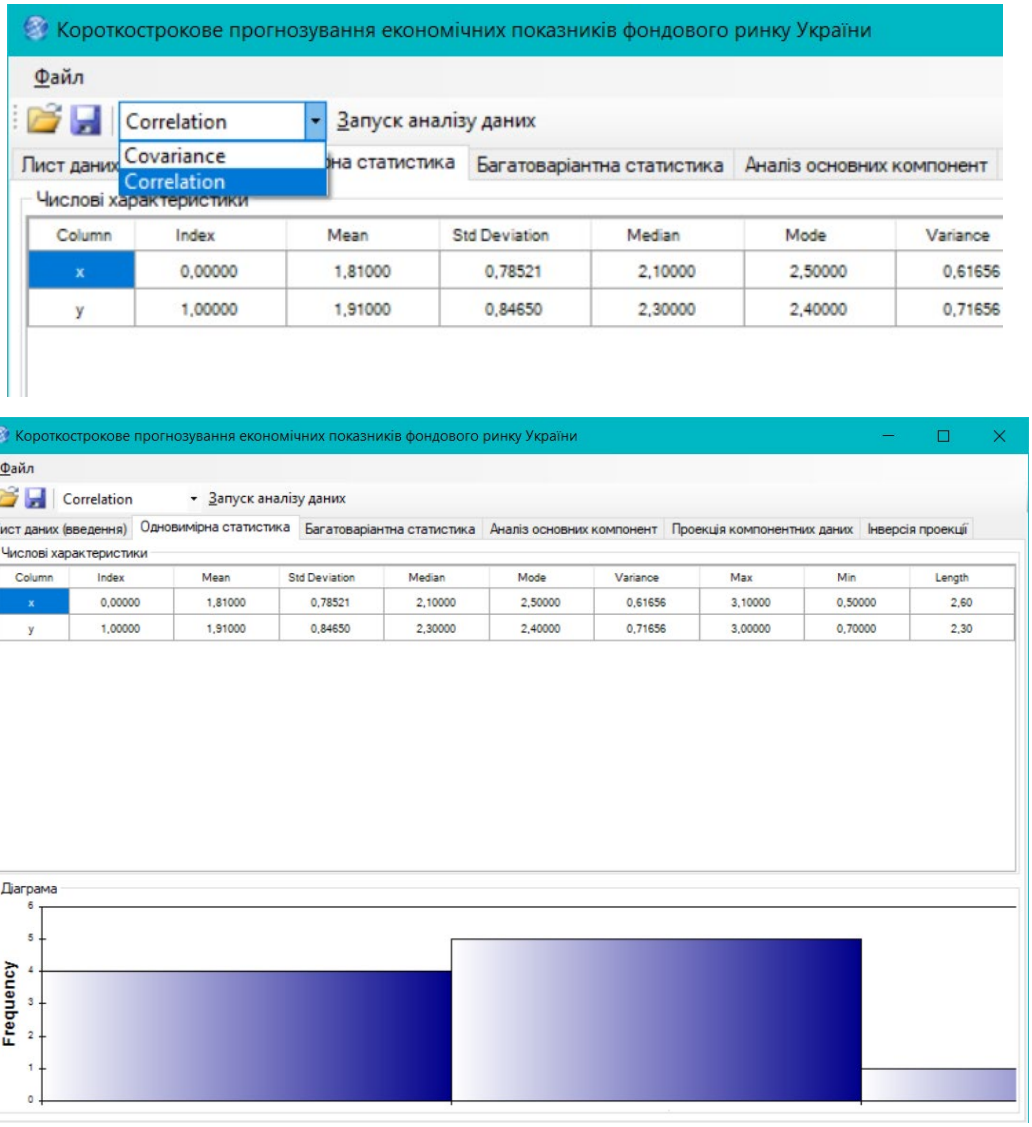


Рисунок 5.4 – Виконання аналізу даних (2)

6 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.:Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн.трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

ДОДАТОК В

Текст програми

ЗАТВЕРДЖУЮ

Проректор
Українського державного університету науки і технології
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЧАСОВИХ РЯДІВ ЗА КРИТЕРІЄМ ЧАСТКОВОЇ СХОЖОСТІ

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1329-01 12 1

Завідувач кафедри КІТ

_____Вадим ГОРЯЧКІН

Керівник розробки

_____Віктор ШИНКАРЕНКО

Виконавець

_____Тімур КДИРОВ

Нормоконтролер

_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

44165850.1329-01 12 01

ДОСЛІДЖЕННЯ ЧАСОВИХ РЯДІВ ЗА КРИТЕРІЄМ ЧАСТКОВОЇ СХОЖОСТІ

Текст програми

Листів 9

2024

ЗМІСТ

Текст програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

using ZedGraph;

using Accord.Statistics.Analysis;
using Accord.Statistics.Controls;


namespace Components
{

   public partial class MainForm : System.Windows.Forms.Form
   {

      private readonly Color[] colors = { Color.YellowGreen, Color.DarkOliveGreen, Color.DarkKhaki, Color.Olive,
         Color.Honeydew, Color.PaleGoldenrod, Color.Indigo, Color.Olive, Color.SeaGreen };



      private PrincipalComponentAnalysis pca;
      private DescriptiveAnalysis sda;

      string[] sourceColumns;



      public MainForm()
      {
         InitializeComponent();

         dgvAnalysisSource.AutoGenerateColumns = true;
         dgvDistributionMeasures.AutoGenerateColumns = false;
```

```
    dgvFeatureVectors.AutoGenerateColumns = true;
    dgvPrincipalComponents.AutoGenerateColumns = false;
    dgvProjectionComponents.AutoGenerateColumns = false;
    dgvProjectionResult.AutoGenerateColumns = true;
}

private void DataAnalyzer_Load(object sender, EventArgs e)
{
    Array methods = Enum.GetValues(typeof(PrincipalComponentAnalysis.AnalysisMethod));
    this.tscbMethod.ComboBox.DataSource = methods;
    this.cbMethod.DataSource = methods;
}


#region Buttons
private void btnRunAnalysis_Click(object sender, EventArgs e)
{
    dgvAnalysisSource.EndEdit();

    double[,] sourceMatrix = ToMatrix(dgvAnalysisSource.DataSource as DataTable, out sourceColumns);

    sda = new DescriptiveAnalysis(sourceMatrix, sourceColumns);

    dgvStatisticCenter.DataSource = new ArrayDataView(sda.DeviationScores, sourceColumns);
    dgvStatisticStandard.DataSource = new ArrayDataView(sda.StandardScores, sourceColumns);

    dgvStatisticCovariance.DataSource = new ArrayDataView(sda.CovarianceMatrix, sourceColumns);
    dgvStatisticCorrelation.DataSource = new ArrayDataView(sda.CorrelationMatrix, sourceColumns);
    dgvDistributionMeasures.DataSource = sda.Measures;


    pca = new PrincipalComponentAnalysis(sda.Source,
        (PrincipalComponentAnalysis.AnalysisMethod)cbMethod.SelectedValue);


    pca.Compute();

    dgvFeatureVectors.DataSource = new ArrayDataView(pca.ComponentMatrix);
    dgvPrincipalComponents.DataSource = pca.Components;
```

```
    dgvProjectionComponents.DataSource = pca.Components;
    numComponents.Maximum = pca.Components.Count;
    numComponents.Value = 1;
    numThreshold.Value = (decimal)pca.Components[0].CumulativeProportion*100;

    CreateComponentCumulativeDistributionGraph(graphCurve);
    CreateComponentDistributionGraph(graphShare);

}

private void btnProject_Click(object sender, EventArgs e)
{
    string[] colNames;
    int components = (int)numComponents.Value;
    double[,] projectionSource = ToMatrix(dgvProjectionSource.DataSource as DataTable, out colNames);
    double[,] m = pca.Transform(projectionSource, components);
    dgvProjectionResult.DataSource = new ArrayDataView(m, GenerateComponentNames(components));
    dgvReversionSource.DataSource = dgvProjectionResult.DataSource;
}

private void btnRevert_Click(object sender, EventArgs e)
{
    double[,] reversionSource = (double[,])(dgvReversionSource.DataSource as ArrayDataView).Data;
    double[,] m = pca.Revert(reversionSource);
    dgvReversionResult.DataSource = new ArrayDataView(m, sourceColumns);
}
#endregion


#region Menus
private void MenuFileOpen_Click(object sender, EventArgs e)
{
    if (openFileDialog.ShowDialog(this) == DialogResult.OK)
    {
        string filename = openFileDialog.FileName;
        string extension = Path.GetExtension(filename);
        if (extension == ".xls" || extension == ".xlsx")
        {
```

```
                ExcelReader db = new ExcelReader(filename, true, false);
                TableSelectDialog t = new TableSelectDialog(db.GetWorksheetList());

                if (t.ShowDialog(this) == DialogResult.OK)
                {
                    this.dgvAnalysisSource.DataSource = db.GetWorksheet(t.Selection);
                    this.dgvProjectionSource.DataSource = db.GetWorksheet(t.Selection);
                }
            }
            else if (extension == ".xml")
            {
                DataTable dataTableAnalysisSource = new DataTable();
                dataTableAnalysisSource.ReadXml(openFileDialog.FileName);

                this.dgvAnalysisSource.DataSource = dataTableAnalysisSource;
                this.dgvProjectionSource.DataSource = dataTableAnalysisSource.Clone();
            }
        }
    }

    private void saveToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (saveFileDialog1.ShowDialog(this) == DialogResult.OK)
        {
            (dgvAnalysisSource.DataSource as DataTable).WriteXml(saveFileDialog1.FileName,
XmlWriteMode.WriteSchema);
        }
    }
    #endregion


    #region Graphs
    public void CreateComponentCumulativeDistributionGraph(ZedGraphControl zgc)
    {
        GraphPane myPane = zgc.GraphPane;

        myPane.CurveList.Clear();

        // Set the titles and axis labels
```

```
    myPane.Title.Text = "Component Distribution";
    myPane.Title.FontSpec.Size = 24f;
    myPane.Title.FontSpec.Family = "Tahoma";
    myPane.XAxis.Title.Text = "Components";
    myPane.YAxis.Title.Text = "Percentage";

    PointPairList list = new PointPairList();
    for (int i = 0; i < pca.Components.Count; i++)
    {
        list.Add(pca.Components[i].Index, pca.Components[i].CumulativeProportion);
    }

    // Hide the legend
    myPane.Legend.IsVisible = false;

    // Add a curve
    LineItem curve = myPane.AddCurve("label", list, Color.Red, SymbolType.Circle);
    curve.Line.Width = 2.0F;
    curve.Line.IsAntiAlias = true;
    curve.Symbol.Fill = new Fill(Color.White);
    curve.Symbol.Size = 7;

    myPane.XAxis.Scale.MinAuto = true;
    myPane.XAxis.Scale.MaxAuto = true;
    myPane.YAxis.Scale.MinAuto = true;
    myPane.YAxis.Scale.MaxAuto = true;
    myPane.XAxis.Scale.MagAuto = true;
    myPane.YAxis.Scale.MagAuto = true;


    // Calculate the Axis Scale Ranges
    zgc.AxisChange();
    zgc.Invalidate();
}

public void CreateComponentDistributionGraph(ZedGraphControl zgc)
{
    GraphPane myPane = zgc.GraphPane;
    myPane.CurveList.Clear();
```

```
            // Set the GraphPane title
            myPane.Title.Text = "Component Proportion";
            myPane.Title.FontSpec.Size = 24f;
            myPane.Title.FontSpec.Family = "Tahoma";

            // Fill the pane background with a color gradient
            //  myPane.Fill = new Fill(Color.White, Color.WhiteSmoke, 45.0f);
            // No fill for the chart background
            myPane.Chart.Fill.Type = FillType.None;

            myPane.Legend.IsVisible = false;

            // Add some pie slices
            for (int i = 0; i < pca.Components.Count; i++)
            {
                myPane.AddPieSlice(pca.Components[i].Proportion, colors[i%colors.Length], 0.1,
pca.Components[i].Index.ToString());
            }

            myPane.XAxis.Scale.MinAuto = true;
            myPane.XAxis.Scale.MaxAuto = true;
            myPane.YAxis.Scale.MinAuto = true;
            myPane.YAxis.Scale.MaxAuto = true;
            myPane.XAxis.Scale.MagAuto = true;
            myPane.YAxis.Scale.MagAuto = true;


            // Calculate the Axis Scale Ranges
            zgc.AxisChange();

            zgc.Invalidate();
        }
        #endregion


        #region Events
        private void numComponents_ValueChanged(object sender, EventArgs e)
        {
```

```
    if (rbComponents.Checked)
    {
        int num = (int)numComponents.Value;
        numThreshold.Value = (decimal)pca.CumulativeProportions[num - 1]*100;

        dgvProjectionComponents.ClearSelection();
        for (int i = 0; i < num && i < dgvProjectionComponents.Rows.Count; i++)
            dgvProjectionComponents.Rows[i].Selected = true;
    }
}

private void numThreshold_ValueChanged(object sender, EventArgs e)
{
    if (rbThreshold.Checked)
    {
        int num = pca.GetNumberOfComponents((float)numThreshold.Value/100);
        numComponents.Value = num;

        dgvProjectionComponents.ClearSelection();
        for (int i = 0; i < num && i < dgvProjectionComponents.Rows.Count; i++)
            dgvProjectionComponents.Rows[i].Selected = true;
    }
}

private void bindingSource1_CurrentChanged(object sender, EventArgs e)
{
    if (dgvDistributionMeasures.CurrentRow != null)
    {
        DataGridViewRow row = (DataGridViewRow)dgvDistributionMeasures.CurrentRow;
        dataHistogramView1.DataSource =
            ((DescriptiveMeasures)row.DataBoundItem).Samples;
    }
}
#endregion
```