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

Факультет інформаційних технологій
(факультет)
Кафедра системного аналізу та управління
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеню магістра

Студента Сердюка Дмитра Олеговича
академічної групи 124м-21-1
спеціальності 124 Системний аналіз
на тему: «Аналіз та прогнозування товарообігу у мережі торгових підприємств»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	Коряшкіна Л.С.			
розділів:				
Інформаційно-аналітичний	Коряшкіна Л.С.			
Спеціальній	Коряшкіна Л.С.			
Рецензент				
Нормоконтролер	Хом’як Т.В.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри
Системного аналізу та управління
(повна назва)
_____ к.т.н., доц. Желдак Т.А.
(підпис) (прізвище, ініціали)
«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *магістра*

студенту Сердюку Д.О. **академічної групи** 124м-21-1
спеціальності: 124 Системний аналіз
на тему «Аналіз та прогнозування товарообігу у мережі торгових підприємств»
затвердженому наказом ректора НТУ «Дніпровська політехніка»
від 31.10.2022 р. №1200-С

Розділ	Зміст	Терміни виконання
Інформаційно-аналітичний розділ	Проблеми аналізу статистичних даних в задачі прогнозування виторгу торгових підприємств. Обґрунтування вибору методів досліджень	10.11.2022
Спеціальний розділ	Розробка і програмна реалізація нового підходу планування виторгу кожного магазину мережі на наступний місяць	07.12.2022

Завдання видано _____ Коряшкіна Л.С.
(підпис керівника) (прізвище, ініціали)

Дата видачі: 06.09.2022 р.

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Сердюк Д.О.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 67 с., 17 рис., 6 табл., 5 додатків, 12 джерел.

Об'єктом дослідження є мережа торгових підприємств

Предметом дослідження є прогнозування товарообігу кожного підприємства мережі, а також їх класифікація

Метою даної кваліфікаційної роботи є підвищення якості планування товарообігу торгівельних підприємств за рахунок залучення та порівняння методів прогнозу

Методи дослідження: прогнозні методи розрахунку, таких як середнє згладжування, вагові коефіцієнти, експоненційні згладжування (подвійне та потрійне) та моделі ARIMA (auto), для класифікації метод бінарного дерева, кластеризація за найбільшою кореляцією.

В *інформаційно–аналітичному розділі* розглянуто методи класифікації, кластеризації об'єктів, прогнозування вихідних даних.

У *спеціальному розділі* обґрунтовано доцільність врахування під час прогнозування товарообігу суб'єкта торгової діяльності не тільки динаміки продажів, але й таких факторів, як його локація, тип, місткість, та інші.

Практична цінність отриманих результатів полягає в тому, що запропонований новий підхід щодо планування товарообігу для торгових підприємств мережі, який допомагає особі, що приймає рішення про відкриття нового підприємства, зважено ухвалити вибір міста, формат та інші параметри для нового торгового об'єкта, який планується відкрити.

Ключові слова: КЛАСТЕРИЗАЦІЯ, КЛАСИФІКАЦІЯ, ARIMA, ЕКСПОНЕНЦІЙНІ ЗГЛАДЖУВАННЯ, КОЕФІЦІЄНТ, ТОРГОВА МЕРЕЖА.

ABSTRACT

Explanatory note: 67 pp., 17 figures, 6 tables, 5 appendices, 12 sources.

The object of the research is trade enterprises network

The subject of the research is the forecasting of the commodity distribution of the skin approach, as well as its classification

The purpose of this qualification work is to make the planning process more qualitative using the methods of intellectual analysis

Research methods: methods for calculating time series, such as average smoothing, weighting factors, exponential smoothing (double and triple) and ARIMA models (auto), for classification, the binary tree method, clustering by the highest correlation.

In *the informational and analytical sectio*n, the methods of classification, clustering and forecasting, which will be used to solve the set tasks, are considered

The implementation of solving the time series problem and the clustering and classification of the enterprise network are considered in *a special section*.

The practical value of the obtained results is that the proposed developed system improves the process of planning the turnover of goods for all trading enterprises of the network

Keywords: CLUSTERIZATION, CLASSIFICATION, ARIMA, EXPONENTIAL SMOOTHING, COEFFICIENT, NETWORK.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ТО – товарообіг

ТМ – торгова мережа

ТП – торгове підприємство

ВСТУП

Товарообіг є одним з основних економічних показників господарської діяльності будь-якого підприємства. Прогнозування і планування товарообігу завжди ґрунтується на аналізі результатів роботи за минулий період часу. При цьому визначаються тенденції зміни, оцінка приросту товарообігу й аналіз його структури. Методика прогнозування товарообігу включає також проведення розрахунково-аналітичної роботи з метою визначення і оцінювання ряду параметрів, що суттєво впливають на зріст чи падіння цього фінансового показника.

Прогноз продажів включає як суб'єктивний (думка фахівців), так і об'єктивний (економіко-математичні розрахунки) елементи. Процеси планування – визначення ключових факторів економічної ситуації (зовнішні та внутрішні фактори), підбір вихідної інформації, аналіз розвитку товарообігу (продажів) у минулому періоді. Складовими елементами плану продажів у свою чергу є: концепція основного та альтернативного плану (на випадок непередбачених обставин), стратегія досягнення цілей, розробка організаційних заходів щодо здійснення стратегії; розрахунок витрат, валових доходів та прибутку, доведення планових завдань до конкретних виконавців, перевірка плану на реальність (контроль прогнозу).

У практичній діяльності поєднуються довго-, середньо- та короткострокові прогнози. Довго-, середньостроковий прогноз стосується стратегічних рішень: виходу новий ринок, інвестування, планів грошових потоків та інших. Короткостроковий прогноз – тактика фірми. Вона стосується плану товарообігу, фінансів, обсягу закупівлі товарів.

Планування доходів – це неодмінно важлива частина будь-якої великої торгівельної мережі, яка дозволяє зробити погляд на майбутнє, розробити комплекс дій, з метою максимізації прибутку та розширення мережі. Точний план, це така константа у часі, на яку розраховує компанія виходячи з багатьох факторів впливу.

Також важливо зазначити що зазвичай компанія розраховує на те, щоб план виконався на сто відсотків, бо саме цей показник дає змогу компанії йти стратегічним шляхом.

Але нажаль трапляються випадки коли план не виконується, або навпаки перевиконується. Так виходить якщо у плані не враховані якісь показники, також можуть траплятись події які неможливо передбачити, такі як відключення світла, повітряна тривога, або жахлива погода.

З плином часу мережа знає все більше і більше факторів, які впливають на розрахунок прибутку, також можна додати що з плином часу компанія накопичує данні(знання), які можна використати для прогнозування.

Такі знання можна збирати за день, за місяць, за рік, для подальшої побудови та дослідження, таку модель можна назвати часовим рядом. На основі часових рядів розробляється модель, завдяки якій можна прогнозувати товарообіг підприємства мережі.

При побудові плану для нового підприємства мережі виникає проблема в тому, що дня нього не існує часового ряду, завдяки якому можна зробити план, тому рішенням цієї проблеми може служити групування вже існуючих підприємств мережі в групи, таким чином щоб можна було б використати їх часові ряди для плану нового підприємства. Тобто процес вже буде називатись класифікацією, оскільки існує показник, за яким нове підприємство увійде в існуючу групу, і для підприємства буде зроблений план.

Роботу присвячено розробці методики аналізу даних про підприємства торгової мережі, у тому числі фінансових, зібраних продовж п’яти років, з метою виявлення закономірностей, які дозволять прогнозувати товарообіг існуючих підприємств на найближчий часовий період або оцінювати фінансові характеристики нових об’єктів торгової діяльності.

1. ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

В даному розділі розглянуться методи, які застосовуються у роботі для аналізу та прогнозування товарообігу підприємств мережі.

1.1. Кластерний аналіз

Кластерний аналіз або кластеризація — це завдання групування набору об’єктів таким чином, щоб об’єкти в одній групі (званій кластером) були більш схожі (в певному сенсі) один на одного, ніж об’єкти в інших групах (кластерах). Це головне завдання дослідницького аналізу даних і загальна техніка статистичного аналізу даних, яка використовується в багатьох галузях, включаючи розпізнавання образів, аналіз зображень, пошук інформації, біоінформатику, стиснення даних, комп’ютерну графіку та машинне навчання.

Сам по собі кластерний аналіз – це не один конкретний алгоритм, а загальне завдання, яке необхідно вирішити. Це може бути досягнуто різними алгоритмами, які суттєво відрізняються у своєму розумінні того, що являє собою кластер, і як їх ефективно знайти. Популярні поняття кластерів включають групи з малими відстанями між членами кластерів, щільні області простору даних, інтервали або певні статистичні розподіли. Тому кластеризацію можна сформулювати як багатоцільову задачу оптимізації. Відповідний алгоритм кластеризації та налаштування параметрів (включно з такими параметрами, як функція відстані для використання, поріг щільності або кількість очікуваних кластерів) залежать від індивідуального набору даних і передбачуваного використання результатів. Кластерний аналіз як такий не є автоматичним завданням, а повторюваним процесом виявлення знань або інтерактивної багатоцільової оптимізації, яка передбачає спроби та невдачі. Часто необхідно змінювати попередню обробку даних і параметри моделі, поки результат не досягне бажаних властивостей.[1]

Основні моделі кластеризації:

- Моделі підключення: наприклад, ієрархічна кластеризація будує моделі на основі віддаленого підключення.
- Центроїдні моделі: наприклад, алгоритм k-means представляє кожен кластер одним вектором середнього.
- Моделі розподілу: кластери моделюються за допомогою статистичних розподілів, таких як багатофакторні нормальні розподіли, які використовуються алгоритмом очікування-максимізації.
- Моделі щільності: наприклад, DBSCAN[2] і OPTICS[3] визначають кластери як пов’язані щільні області в просторі даних.
- Моделі підпростору: у бікластеризації (також відомій як спільна кластеризація або дворежимна кластеризація) кластери моделюються як членами кластера, так і відповідними атрибутами.
- Групові моделі: деякі алгоритми не надають уточненої моделі для своїх результатів і лише надають інформацію про групування.
- Моделі на основі графів: кліку, тобто підмножину вузлів у графі, так що кожні два вузли в підмножині з’єднані ребром, можна розглядати як прототипну форму кластера. Послаблення повної вимоги зв’язності (частка ребер може бути відсутнім) відомі як квазікліки, як в алгоритмі кластеризації HCS.
- Моделі графів зі знаком: кожен шлях у графі зі знаком має знак із добутку знаків на ребрах. Згідно з припущеннями теорії балансу, ребра можуть змінити знак і призвести до роздвоєного графа. Слабша "аксіома кластерності" (жоден цикл не має точно одного негативного ребра) дає результати з більш ніж двома кластерами або підграфами лише з позитивними ребрами.
- Нейронні моделі: найвідомішою неконтрольованою нейронною мережею є карта самоорганізації, і ці моделі зазвичай можна охарактеризувати як подібні до однієї або кількох із зазначених вище моделей, включаючи моделі

підпростору, коли нейронні мережі реалізують форму аналізу головних компонентів або незалежного аналізу. Аналіз компонентів.

В роботі розглядаються дані, які містять більше двох компонентів, через це візуалізація даних є неможливою, а розрахунки громіздкі, тому для вирішення проблеми використовується метод головних компонентів, метод факторного аналізу в статистиці, який використовує ортогональне перетворення множини спостережень з можливо пов'язаними змінними (сутностями, кожна з яких набуває різних числових значень) у множину змінних без лінійної кореляції, які називаються головними компонентами.[4]

1.2. Класифікація

Задача класифікації — формалізована задача, яка містить множину об'єктів (ситуацій), поділених певним чином на класи. Задана скінченна множина об'єктів, для яких відомо, до яких класів вони належать. Ця множина називається вибіркою. До якого класу належать інші об'єкти невідомо. Необхідно побудувати такий алгоритм, який буде здатний класифікувати довільний об'єкт з вихідної множини. Класифікація об'єкта — номер або найменування класу, що видається алгоритмом класифікації в результаті його застосування до цього об'єкта.

В математичній статистиці задачі класифікації називаються також задачами дискретного аналізу. В машинному навчанні завдання класифікації вирішується, як правило, за допомогою методів штучної нейронної мережі при постановці експерименту у вигляді навчання з учителем.

Простір характеристик

Характеристикою називається відображення $f: X \to D_f$, де D_f — множина допустимих значень характеристики. Якщо задані характеристики$f_1 \dots, f_n$, то вектор $x = (f_1(x), \dots, f_n(x))$ називається характеристичним описом об'єкта $x \in X$.

Характеристики можна ототожнювати із самими об'єктами. При цьому множину $X = D_{f_1} \times ... \times D_{f_n}$ називають простором характеристик.

Залежно від множини D_f характеристики поділяються на такі типи:

- Бінарні характеристики: $D_f = \{0,1\}$;
- Номінальні характеристики: D_f — скінченна множина;
- Порядкові характеристики: D_f— скінченна впорядкована множина;
- Кількісні характеристики: D_f — множина дійсних чисел.

Часто зустрічаються прикладні задачі з різнотипними характеристиками, для їх вирішення підходять далеко не всі методи.

Типи вхідних даних

- Характеристичний опис — найпоширеніший випадок. Кожен об'єкт описується набором своїх характеристик, які називаються ознаками. Ознаки можуть бути числовими або нечисловими.
- Матриця відстаней між об'єктами. Кожен об'єкт описується відстанями до всіх інших об'єктів навчальної вибірки. З цим типом вхідних даних працюють деякі методи, зокрема, метод найближчих сусідів, метод потенційних функцій.
- Часовий ряд або сигнал є послідовність вимірів у часі. Кожен вимір може представлятися числом, вектором, а в загальному випадку — характеристичним описом досліджуваного об'єкта в цей час часу.
- Зображення або відеоряд.
- Зустрічаються і складніші випадки, коли вхідні дані представляються у вигляді графів, текстів, результатів запитів до бази даних, і т. д. Як правило, вони приводяться до першого або другого випадку шляхом попередньої обробки даних та вилучення характеристик.[5]

Класифікацію сигналів та зображень називають також розпізнаванням образів.

Дерево прийняття рішень (також можуть називатися деревами класифікацій або регресійними деревами) — використовується в галузі статистики та аналізу даних для прогнозних моделей. Структура дерева містить такі елементи: «листя» і «гілки». На ребрах («гілках») дерева прийняття рішення записані атрибути, від яких залежить цільова функція, в «листі» записані значення цільової функції, а в інших вузлах — атрибути, за якими розрізняються випадки. Щоб класифікувати новий випадок, треба спуститися по дереву до листа і видати відповідне значення. Подібні дерева рішень широко використовуються в інтелектуальному аналізі даних. Мета полягає в тому, щоб створити модель, яка пророкує значення цільової змінної на основі декількох змінних на вході.

Дерево рішень — це графічне зображення послідовності рішень і станів середовища з указівкою відповідних імовірностей і виграшів для будь-яких комбінацій альтернатив і станів середовища.

Використання цього методу передбачає, що вся необхідна інформація про очікувані виграші для кожної альтернативи та імовірності виникнення всіх ситуацій була зібрана заздалегідь.

Метод "дерева рішень" застосовують на практиці у ситуаціях, коли результати одного рішення впливають на подальші рішення, тобто, для прийняття послідовних рішень.[6]

Коефіцієнт Джині – це метрика, яка вказує на дискримінаційну силу моделі, а саме на ефективність моделі в розрізненні «поганих» позичальників, які зазнають дефолту в майбутньому, і «хороших» позичальників, які не зазнають дефолту в майбутньому. Цей показник часто використовується для порівняння якості різних моделей і оцінки їхньої потужності прогнозування.[7]

1.3. Прогнозування

В роботі розглядаються методи прогнозування часових рядів для всіх підприємств мережі.

Часовий ряд — це ряд точок даних, проіндексованих (або перелічених, або відкладених на графіку) в хронологічному порядку. Найчастіше часовий ряд є послідовністю, взятою на рівновіддалених точках в часі, які йдуть одна за одною. Таким чином, він є послідовністю даних дискретного часу.[8]

Методи які розглядаються: рухомого середнього, зваженого середнього, подвійного та потрійного експоненційного згладжування та ARIMA.

У статистиці ковзне середнє (ковзне середнє або поточне середнє) — це обчислення для аналізу точок даних шляхом створення серії середніх різних підмножин повного набору даних. Його також називають рухомим середнім (РС) або рухомим середнім і є різновидом фільтра кінцевої імпульсної характеристики. Варіанти включають: просту, кумулятивну або зважену форми.[9]

$$
\begin{aligned}
SMA_{k,\text{next}} &= \frac{1}{k}\sum_{i=n-k+2}^{n+1} p_i \\
&= \frac{1}{k}\Big(\underbrace{p_{n-k+2} + p_{n-k+3} + \cdots + p_n + p_{n+1}}_{\sum_{i=n-k+2}^{n+1} p_i} + \underbrace{p_{n-k+1} - p_{n-k+1}}_{=0}\Big) \\
&= \underbrace{\frac{1}{k}\Big(p_{n-k+1} + p_{n-k+2} + \cdots + p_n\Big)}_{=SMA_{k,\text{prev}}} - \frac{p_{n-k+1}}{k} + \frac{p_{n+1}}{k} \\
&= SMA_{k,\text{prev}} + \frac{1}{k}\Big(p_{n+1} - p_{n-k+1}\Big)
\end{aligned}
$$

Середнє зважене – це середнє, яке має коефіцієнти множення, щоб надати різні ваги даним у різних позиціях у вікні вибірки. З математичної точки зору зважене ковзне середнє — це згортка даних із фіксованою ваговою функцією.[10]

$$\begin{aligned}
\mathrm{Total}_{M+1} &= \mathrm{Total}_M + p_{M+1} - p_{M-n+1} \\
\mathrm{Numerator}_{M+1} &= \mathrm{Numerator}_M + np_{M+1} - \mathrm{Total}_M \\
\mathrm{WMA}_{M+1} &= \frac{\mathrm{Numerator}_{M+1}}{n + (n-1) + \cdots + 2 + 1}
\end{aligned}$$

Експоненціальне згладжування — це емпіричне правило для згладжування даних часових рядів за допомогою функції експоненціального вікна. Тоді як у простому ковзному середньому минулі спостереження зважуються однаково, експоненціальні функції використовуються для призначення експоненційно зменшуваних ваг з часом. Це процедура, яку легко освоїти та легко застосувати, щоб зробити певне визначення на основі попередніх припущень користувача, таких як сезонність:[11]

$s_0 = x_0$

$s_t = \alpha x_t + (1-\alpha)s_{t-1}, \quad t > 0$

$0 < \alpha < 1$

ARIMA – частина AR ARIMA вказує на те, що змінна, яка цікавить, регресує на її власні відсталі (тобто попередні) значення. Частина MA вказує на те, що помилка регресії насправді є лінійною комбінацією членів помилки, значення яких мали місце одночасно та в різний час у минулому. I (для «інтегрованого») вказує на те, що значення даних було замінено на різницю між їхніми значеннями та попередніми значеннями (і цей процес розрізнення міг виконуватися більше одного разу). Мета кожної з цих функцій полягає в тому, щоб модель якомога краще відповідала даним.[12]

2. СПЕЦІАЛЬНИЙ РОЗДІЛ

У спеціальному розділі розглядаються проблематика обраних задач з подальшим їх рішенням

2.1. Постановка задачі дослідження

В існуючій мережі торгових підприємств досліджуються два основних питання, перше це групування існуючих ТМ за загальними ознаками та за ознакою тижневої сезонності, друге – це розрахунок для кожного підприємства мережі плану товарообігу на наступний місяць використовуючи методи розрахунку часових рядів, та порівняння результатів методів з реальними даними за місяць на предмет найбільш точного методу.

Метою даної роботи є обґрунтування доцільності врахування під час прогнозування товарообігу суб'єкта торгової діяльності не тільки динаміки продажів, але й таких факторів, як його локація, тип, місткість, та інші.

Вихідні дані для дослідження: база даних містить інформацію про торгові підприємства мережі з наступними атрибутами: місто, в якому функціонує підприємство; кількість населення, яку покриває підприємство; формат організації, площа та кількість стелажів; локація (street, ТЦ, ТРЦ); тип локації (офіс чи студенти, спальний район, транзит, центр міста); щоденний обсяг продажів протягом року.

Постановка задачі. Для досягнення поставленої мети цієї роботи, були поставлені наступні задачі:

- на основі фінансових показників отримати вторинні дані про середньомісячний товарообіг для кожного підприємства, а також фактор денної сезонності;
- провести кластеризацію підприємств за фактором денної сезонності і виявити унікальні ознаки кожної групи;

- розширити заданий набір даних додатковими ознаками й побудувати правила або дерево рішень щодо класифікації нових торгівельних підприємств задля прогнозування його товарообігу;
- Провести тестування, зробити висновки.

2.2. Програмна реалізація поставлених задач

2.2.1. Кластеризація підприємств за фактором денної сезонності

Структура вихідних даних: база даних містить інформацію про 370 торгових підприємств мережі.

На основі цих фінансових показників отримано вторинні дані про середньомісячний товарообіг для кожного підприємства, а також фактор денної сезонності. Для оцінки цього показника з даних про обсяги продажів було видалено аномалії, пов’язані зі святами, поганими погодними умови (наприклад снігопад, через який підприємство недоотримало прибуток), а також днями, в яких підприємство, з різних причин, не працювало. Далі обчислюється середні за днями тижня продажі і розраховується так званий коефіцієнт денної сезонності, який показує, на скільки відсотків продажі в певний день (понеділок, вівторок і т.д.) відхиляються від середніх продажів за тиждень.

Приклад розрахованих коефіцієнтів денної сезонності наведений в табл. 2.1.

Таблиця 2.1- Коефіцієнти денної сезонності підприємств

Номер підприємства	Понеді-лок	Вівторок	Середа	Четвер	П'ятниця	Субота	Неділя
1	1,12	1,17	1,12	1,09	1,07	0,73	0,7
2	1,07	1,12	1,11	1,1	1,12	0,81	0,67
3	1,02	1,03	1,04	1,01	1,03	0,96	0,9
4	1,02	1,07	1,06	1,06	1,07	0,9	0,82
5	1,01	1,03	1,02	1,02	1,04	0,97	0,92
6	1,02	1,02	1,05	1,02	1,07	0,94	0,88

Візуалізація даних отриманої таблиці сезонності за допомогою методу головних компонент дозволила зробити висновок про можливість розбиття 327 підприємств мережі на сім груп (решта підприємств не функціонувала на момент збору даних). Далі проводимо початкове групування на основі аналізу кореляційної матриці за такими правилами: кожна група повинна містить не менш ніж 15 підприємств; якщо кореляція між підприємствами більше за 0.8, то вони відносяться до одною групи. З кожним новим доданим підприємством у групі перераховується центр кластеру. Для решти підприємств, яку не було віднесено до жодного з кластерів, створюємо окрему групу. Потім, ігноруючи зазначену вище умову кореляційного зв’язку, розкидаємо всі підприємства по існуючим групам за найкоротшою відстанню до центру кластеру. Перераховуючи кожного разу центри кластерів, виявляємо можливість перенесення кожного підприємства з одного кластеру до іншого до тих пір, поки кластери не устаткуються. Результат кластеризації наведений на рис. 1.

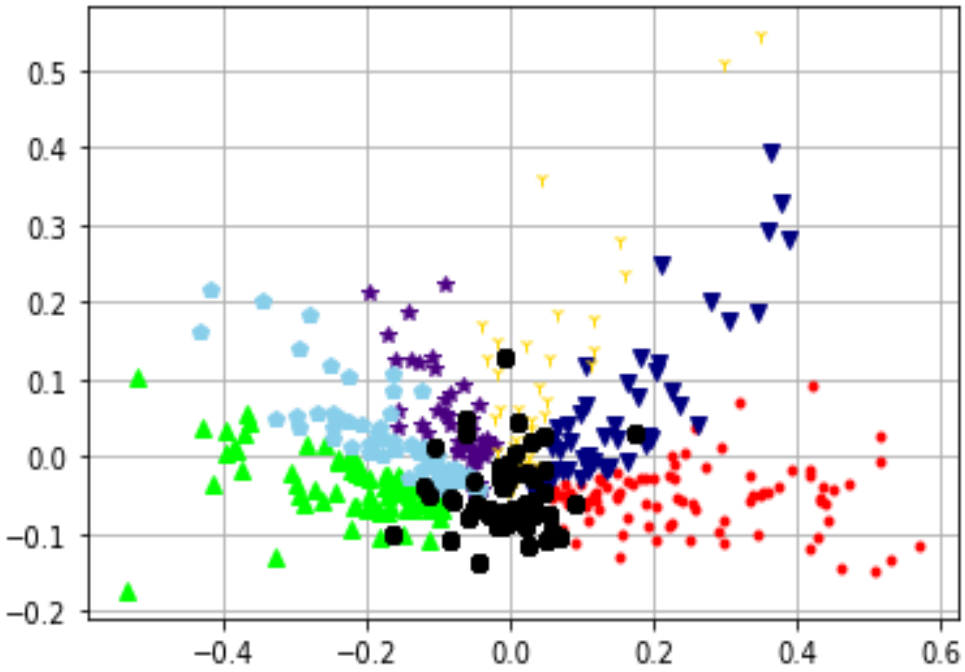

Рисунок 2.1 – Результат кластеризації

З рис. 2.1 можна побачити сформовані кластери та кластер в якому знаходяться магазини які не мають зазначеного раніше кореляційного зв’язку (виділені чорним кольором).

На рис. 2.2 представлена тижнева сезонність для кожної виділеної групи. Розглядаючи кожну групу окремо, можна визнати деякі ознаки які їх характеризують.

Наприклад, в групі 0 наявний низький прибуток у вихідні дні, особливо в неділю, в групі 1 можна побачити зріст у п’ятницю і середній показник у суботу, але неділя теж найгірший день.

У групі 2 наявна зворотна сезонність, у графіку можна побачити що у вихідний день найбільші прибутки.

Відміна груп 2 і 3 – це неділя, яка в групі 3 йде на сильний спад після суботі, при тому що п’ятниця має майже такий результат як і субота.

Група 4 виділяється дуже великим зростанням в суботу і гарною неділею.

В п’ятій групі вихідні як два абсолютно різні дні, субота найкращий, а неділя найгірший.

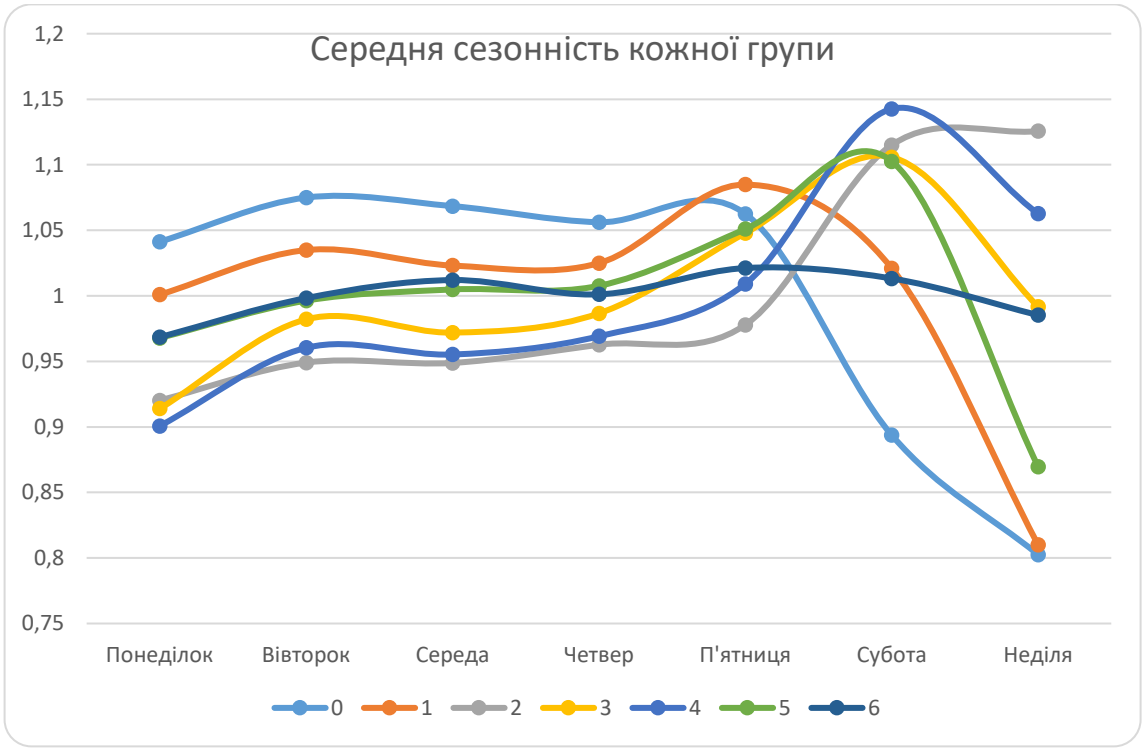

Рисунок 2.2 – Середня сезонності за кожною групою

Шоста група виділяється окремо, оскільки для неї не характерна сезонність по дням тижня. Це зумовлено тим, що всі підприємства групи мають різну сезонність. Тому на рис. 3, групу розглянуто окремо.

Як можна побачити з рис. 3, між всіма членами групи 6 немає жодної кореляції, тому її можна далі розбивати на нові групи, більш малі, та дивитись результати, але це не є предметом даного дослідження.

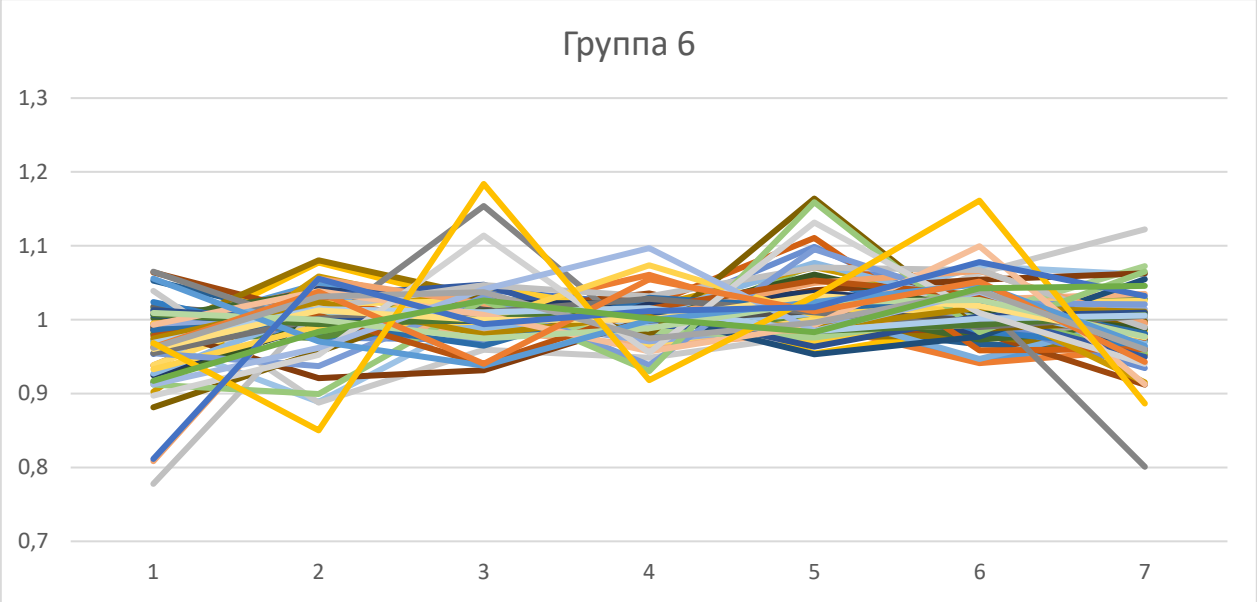

Рисунок 2.3 –Сезонності шостої групи

Для порівняння двох описаних підходів щодо оцінювання сезонності підприємств і кластеризації за цією ознакою підприємств, кластеризуємо вибірку за методом k-найменших.

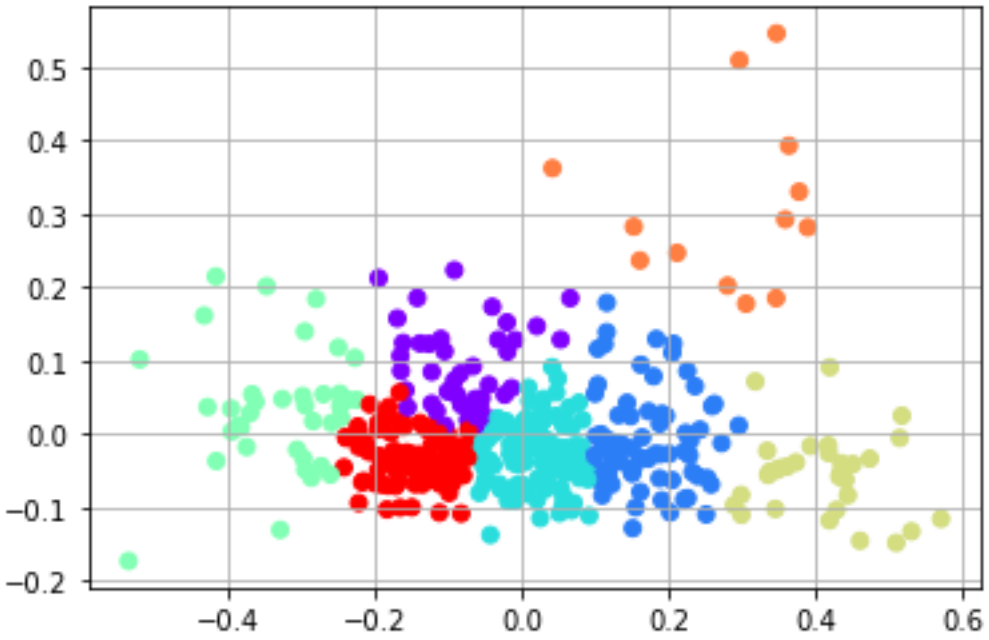

Рисунок 2.4 – Результат кластеризації методом k-середніх

Як видно з рис. 2.4, використаний метод поділив всі значення на 7 кластерів, в даному випадку немає такого кластера, який у себе приймає всіх які не підходять до якоїсь із груп. Не зважаючи на відмінності в графічному представленні результатів кластеризації підприємств за двома підходами, проведемо детальний аналіз отриманих груп.

На рис. 5 можна простежити середні значення за кожною з груп.

Група 4, має виражену спадну сезонність у вихідні дні, як і перша, але перша має не такий сильний спад як нульова.

Група 2 не має якоїсь вираженої сезонності, його пізніше розглянемо окремо.

Групи 3 і 6 мають висхідну сезонність по вихідним дням, різницю полягає лише в силі підйому.

Група 5 має близьку тенденцію з групою 0, різниця в силі спаду, у п'ятої групи великий спад на неділю, а в 0 групи спад не є значним.

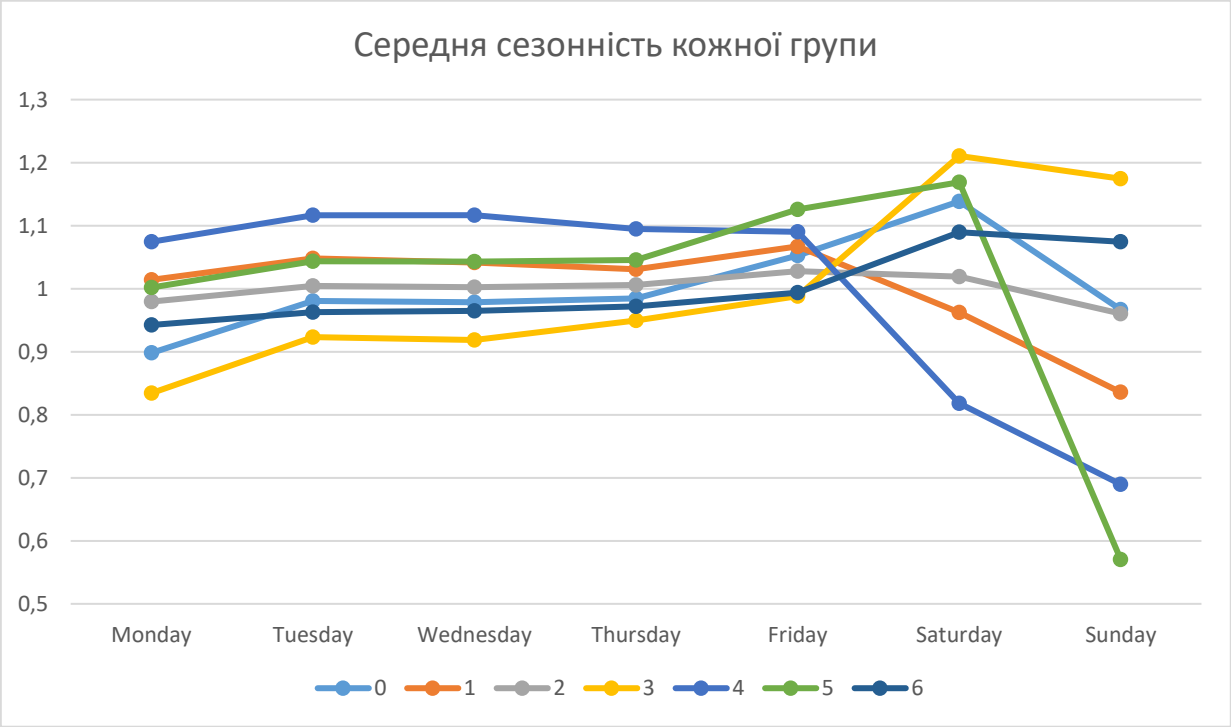

Рисунок 2.5 – Результат кластеризації методом k-середніх

На рис. 2.6 можна побачити детальний огляд всіх підприємств мережі, які попали в групу 2. На рис. 2.4 вона знаходиться посередині і має бірюзовий колір. З графіку групи 2, можна побачити що всі підприємства групи мають різну тижневу сезонність, тобто як і в попередньому методі. Для більш детального опису магазинів даної групи треба робити розбивку саме цієї групи.

Тож, розглянувши два методи можна зробити висновок, що їх відповіді є близькими але не однаковими. В обидвох випадках утворюється група, всі члени якої не корелюються між собою, тобто цю групу можна виключити з подальшого розгляду.

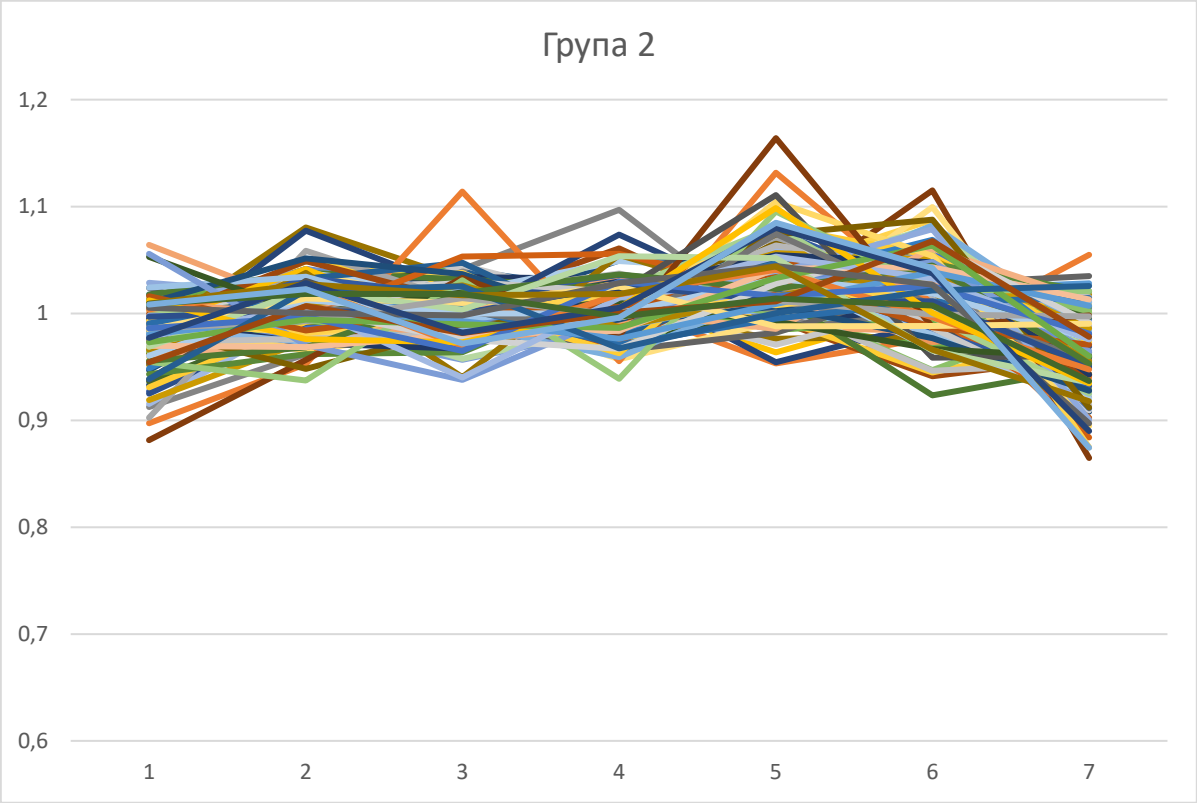

Рисунок 2.6 – Тижнева сезонність підприємств групи 2

Проведена кластеризація дозволяє виявити унікальні ознаки кожної групи. Відтак, нові дані про належність підприємства до певного кластеру можна надалі

використовувати як вихідний параметр під час класифікації об’єктів мережі для прогнозування сезонності і товарообігу нових підприємств.

2.2.2. Побудова правил рішень щодо класифікації торгівельних підприємств

В даній частині роботи вирішується питання доцільності врахування під час прогнозування товарообігу суб’єкта торгової діяльності не тільки динаміки продажів, але й таких факторів, як його локація, тип, місткість, та інші.

Отже, оцінювання денної сезонності проведено за наступною схемою: з даних про обсяги продажів було видалено аномалії, пов’язані зі святами, поганими погодними умови, а також днями, в яких підприємство, з різних причин, не працювало. Далі обчислювалися середні за днями тижня продажі і розраховувався так званий коефіцієнт денної сезонності, який показує, на скільки відсотків продажі в певний день (понеділок, вівторок і т. ін.) відхиляються від середніх продажів за тиждень.

Для задачі класифікації існує вибірка в якій вихідний параметр ранжовано у формат від 1 до 7 (рис. 2.7), тобто це будуть наші 7 груп на які розбиваються підприємства. В даних присутні 3 числові та 4 текстові ознаки. Існують два підходи роботи з не цифровими ознаками, перший це перевести ці ознаки у бінарний формат, тоді кількість колонок збільшиться на кількість унікальних значень ознаки, другий підхід це перевести текст у число за допомогою кодування, кожній текстовій ознаці поставити у відповідність кодове число. В даному випадку було вибрано другий варіант, оскільки від кількості ознак класифікація стане складнішою.

Number	Rang	Format_cc	City_code	Area_koef	Area	Racks	Location_(	Location_T
1	3	4	6	31958.84	129.82	81	1	3
2	4	5	6	31958.84	151.47	96	2	3
3	4	4	6	31958.84	158.42	81	1	2
4	2	5	6	31958.84	180.47	104	1	3
5	5	5	6	31958.84	144.12	97	2	3
6	3	5	6	31958.84	172.68	95	1	3

Рисунок 2.7 – Вхідні значення для класифікації

Отже дані містять сім ознак і групи, але в одному зі стовпців з ознаками відсутні дані, тому перед основною класифікацією було зроблено класифікацію для заповнення пробілів у даних. Для цього потрібно дві вибірки, навчальна і прогнозна. Перша має в собі 6 ознак і стовпець з відповідями, друга має лише 6 ознак. Вирішуємо задачу за допомогою методу дерева рішень. На рис 4, зображено правила за якими вирішується дана задача

Number	Format_cc	Rang	City_code	Area_koef	Area	Racks	Location_(	Location_T
1	4	3	6	31958.84	129.82	81	1	3
2	5	4	6	31958.84	151.47	96	2	3
3	4	4	6	31958.84	158.42	81	1	2
4	5	2	6	31958.84	180.47	104	1	3
5	5	5	6	31958.84	144.12	97	2	3
6	5	3	6	31958.84	172.68	95	1	3
7	6	5	6	31958.84	171.75	114	2	3

Рисунок 2.8 – Вхідні значення для тестування

На рис. 2.8 наведені дані які будуть використовуватись для отримання результату після навчання, саме в цих були пропуски у даних.

На рис. 2.9 наведені дані на яких вибірка буде навчатись. Ця таблиця буде ділитись у пропорції 15% до 85 %, що означає що 15 відсотків з вибірки буде використовуватись для тестування, а 85 відсотків вибірки для навчання.

Після підрахунку пропусків, усі значення формуються у вихідну таблицю, як це було на початку, але вже повністю заповнена.

Number	Rang	City_code	Area_koef	Area	Racks	Location_code	Location_Type_code
8	7	6	31958.84	181.52	85	3	1
9	7	6	31958.84	165.26	84	2	2
10	4	6	31958.84	166.78	87	1	4
11	6	6	31958.84	146.38	90	3	1
12	3	63	43591	217.33	119	1	3
13	4	80	70357	182	108	2	4
14	1	3	151948	195.31	108	1	3

Рисунок 2.9 – Вхідні значення для навчання

В процесі розрахунку пропусків, програма формує правило, дерево рішень, наведено нижче(рис. 2.10) показує як воно робить процес класифікації. З дерева можна бачити що всі рішення приймаються за допомогою коефіцієнта джині.

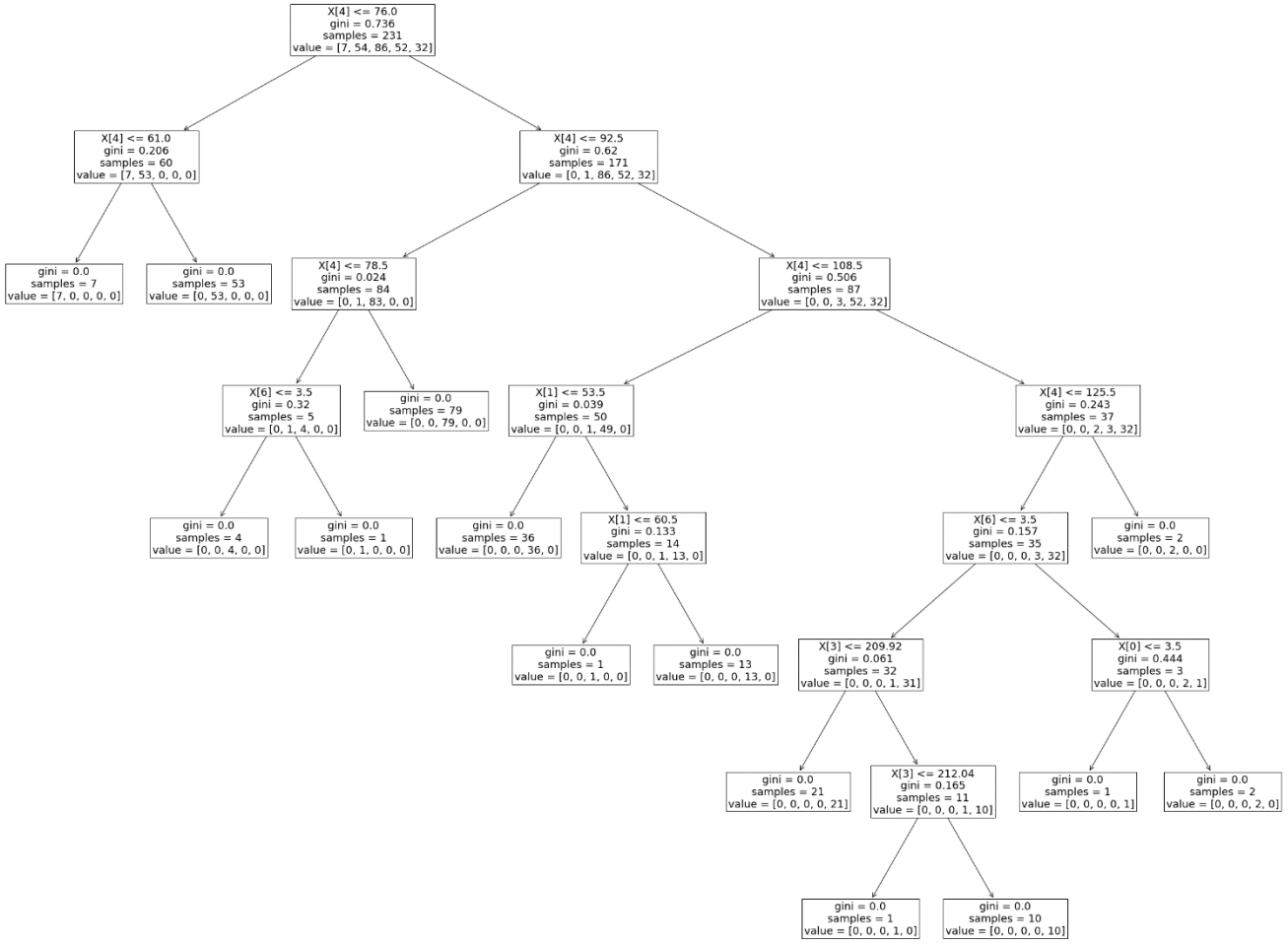

Рисунок 2.10 –Дерево рішень

З рис. 2.10 можна побачити що програма розглядає стовпець 5 як найбільш інформаційний, тобто кількість стелажів. Перше правило це якщо стелажів більше ніж 76, тоді вірогідність попадання у початкові ранги, перший і другий, якщо більше 76 то розглядається далі. Потім система розглядає випадки коли стелажів може бути менше ніж 92.5, якщо менше, то вірогідність середнього рангу дуже висока, далі по цій гілці розглядається фактор 78.5 стелажів, і якщо цей параметр задовольняється, то далі розглядається параметр тип локації і так далі поки всі можливі варіанти не будуть розглянуті.

Отже надалі розглядається повністю заповнена таблиця яка не має ніяких пропусків на предмет класифікації всієї торгової мережі та визначення фактору впливу при додаванні стовпця груп після кластерного аналізу.

Number	Rang	Format_cc	City_code	Area_koef	Area	Racks	Location_(	Location_T
1	3	4	6	31958.84	129.82	81	1	3
2	4	5	6	31958.84	151.47	96	2	3
3	4	4	6	31958.84	158.42	81	1	2
4	2	5	6	31958.84	180.47	104	1	3
5	5	5	6	31958.84	144.12	97	2	3
6	3	5	6	31958.84	172.68	95	1	3
7	7	4	6	31958.84	181.52	85	3	1
8	5	6	6	31958.84	171.75	114	2	3

Рисунок 2.11 – Вхідні значення для класифікації

З рисунку 2.11 можна бачити що вхідні дані мають номер підприємства мережі, ранг який буде як стовпець з відповідями, формат магазину, код міста, коефіцієнт густини магазину у місті це можна інтерпретувати як кількість людей у місті, яка ходить до одного магазину. Торгова площа торгового підприємства, кількість стелажів, код локації де саме знаходиться магазин(ТЦ, ТРЦ, Стрит ритейл), та тип локації(житловий масив, транзит, офіси, центр міста). Також друга вибірка має додатковий стовпець, це буде номер кластеру.

При класифікації вибірки з стовпцем номера кластеру, було отримано наступні результати: отримана точність є 37.65%., при класифікації вибірки без стовпця номера кластера, отримано результат: 43.01%.

Обидва результати є недостатніми, і це може бути пов’язано з декількома факторами, такими як недостатність даних для навчання, завелика кількість стовпців для навчання, але висновок з отриманих даних можна зробити такий, що додавання стовпця група кластерів на вибірці у 370 підприємств мережі, не дає ніякого збільшення точності класифікації, тому в даному випадку треба його прибрати, і працювати з основними стовпцями.

Результати проведених експериментів підтвердили дієвість розробленої методика аналізу і прогнозування сезонності і товарообігу підприємств.

2.2.3. Прогнозування

По кожному підприємству є дані за місячний товарообіг з моменту запису статистики(з січня 2017) або з моменту його відкриття. Постановка задачі, треба розробити методику планування на наступний місяць, використовуючи дані за попередні місяці.

В розробці плану використовувались методи: рухомого середнього, зваженого середнього, подвійного та потрійного експоненційного згладжування, ARIMA з сталими коефіцієнтами та auto Arima – функція яка може підібрати оптимальні значення.

Таблиця 2.2- Вхідні дані для прогнозування

TT	01.01.2017	01.02.2017	01.03.2017	01.04.2017
1	320870	327043	408027	300046
2	476851	451535	588144	450442
3	445154	503613	651096	554114
4	336334	327997	425323	352532
5	559529	560869	740987	631959

В табл. 2.2 знаходяться всі дані по всім підприємствам мережі. Дані складаються з: номера підприємства у мережі та з місячного товарообігу. Для розрахунків було взято дані в період з 1.1.2017 по 1.9.2022. Програмна реалізація дозволяє отримувати графік функції для кожного підприємства мережі(рис. 12).

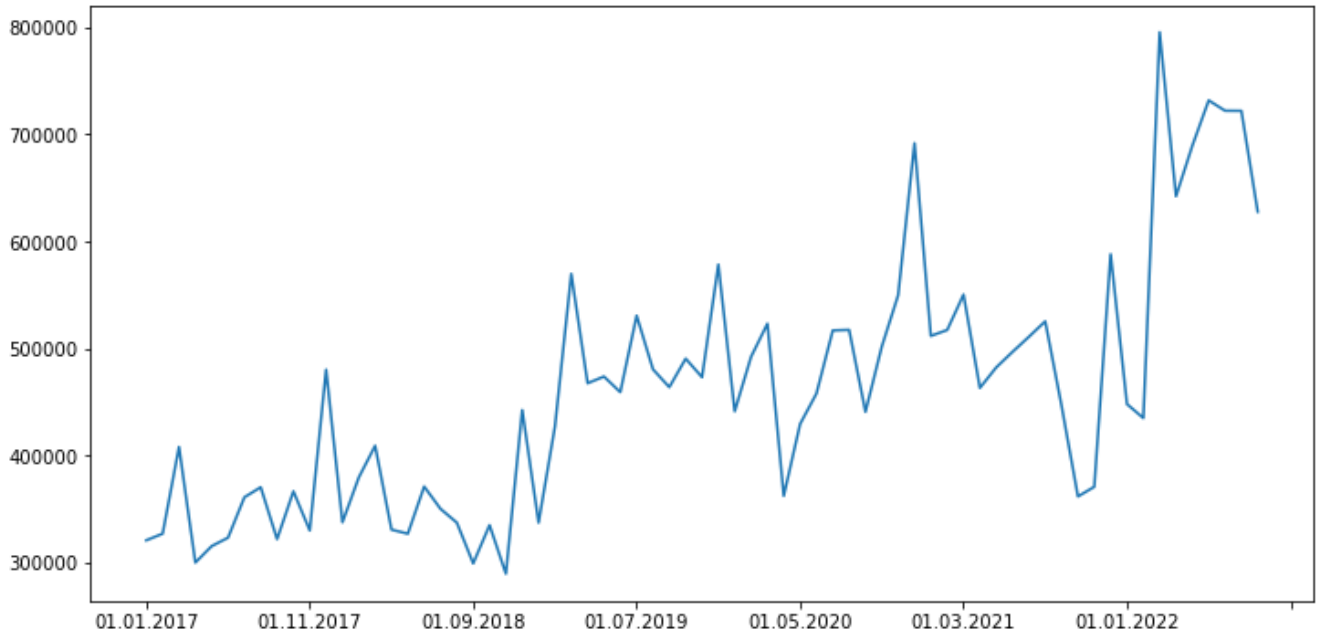

Рисунок 2.12 – Дані по підприємству мережі

На рис. 2.13 зображено реалізацію метода середнього зваженого, можна бачити реальні дані, метод, та нижню і верхню границі. Роблячи висновок з методу можна зазначити що в даному випадку метод згладжує за період в 12 місяців, саме

на такий період наявна сезонність, особливо сезонність можна помітити звернувши увагу на піки по груднях. Ще на графіку можна помітити 2 фактори підвищення товарообігу, які можна пов’язати з фактором інфляції, тим самим можна умовно поділити на чотири групи дані з графіку: перше – це період до початку короно вірусних обмежень, друга група – це їх впровадження, метод показує тенденцію на збільшення виручки підприємством, третя група це плато, тобто швидка тенденція на збільшення зникла, але можна бачити що в середньому за 12 місяців товарообіг не збільшився, остання - це військові дії і як факт підвищення цін.

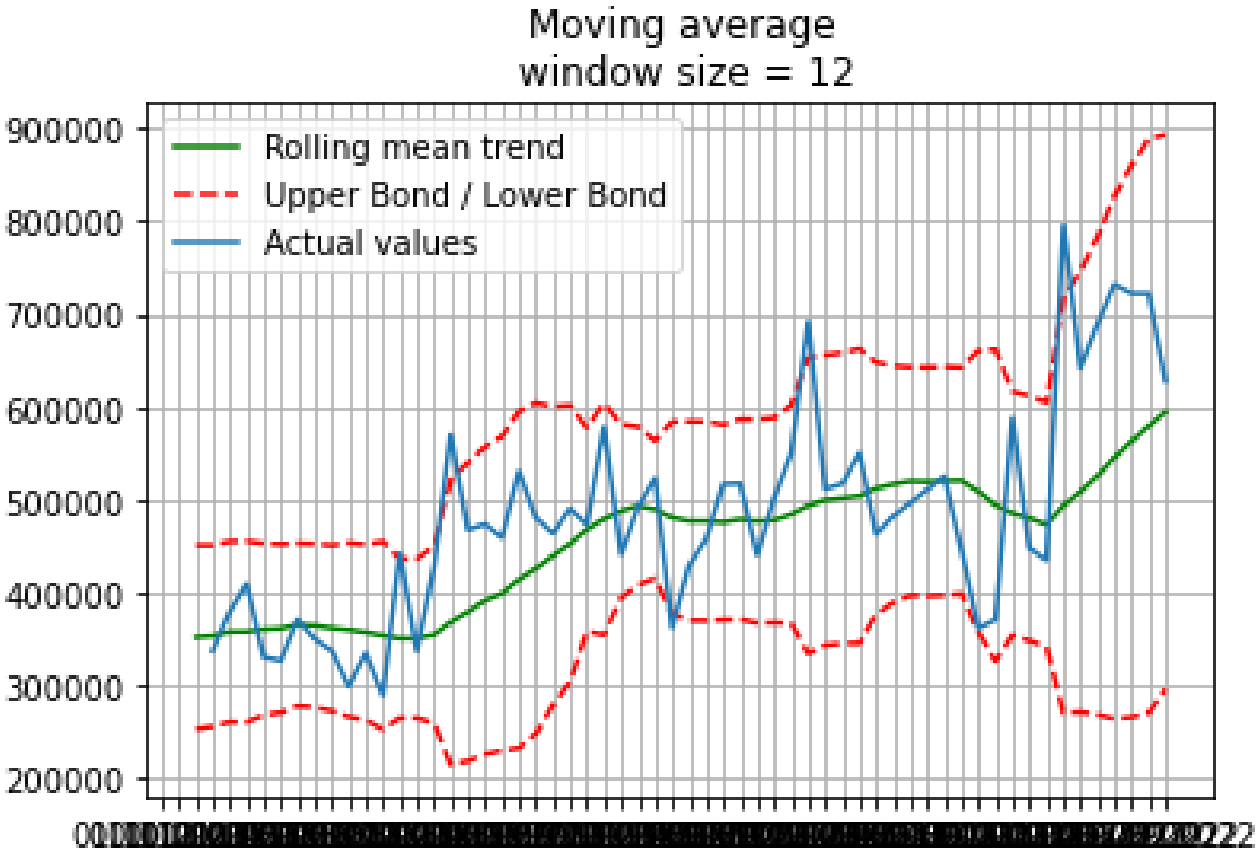

Рисунок 2.13 – Метод середнього зваженого

Наступний метод який розглядається і має візуалізацію це експоненційне згладжування, яке зображено на рис. 2.14. На рисунку зображено графік підприємства мережі, яке розглядалось вище, та методи згладжування. Як можна

побачити на графіку, чим більше коефіцієнт α, тим більша точність згладжування, тобто при α=0, отримаємо середнє зважене.

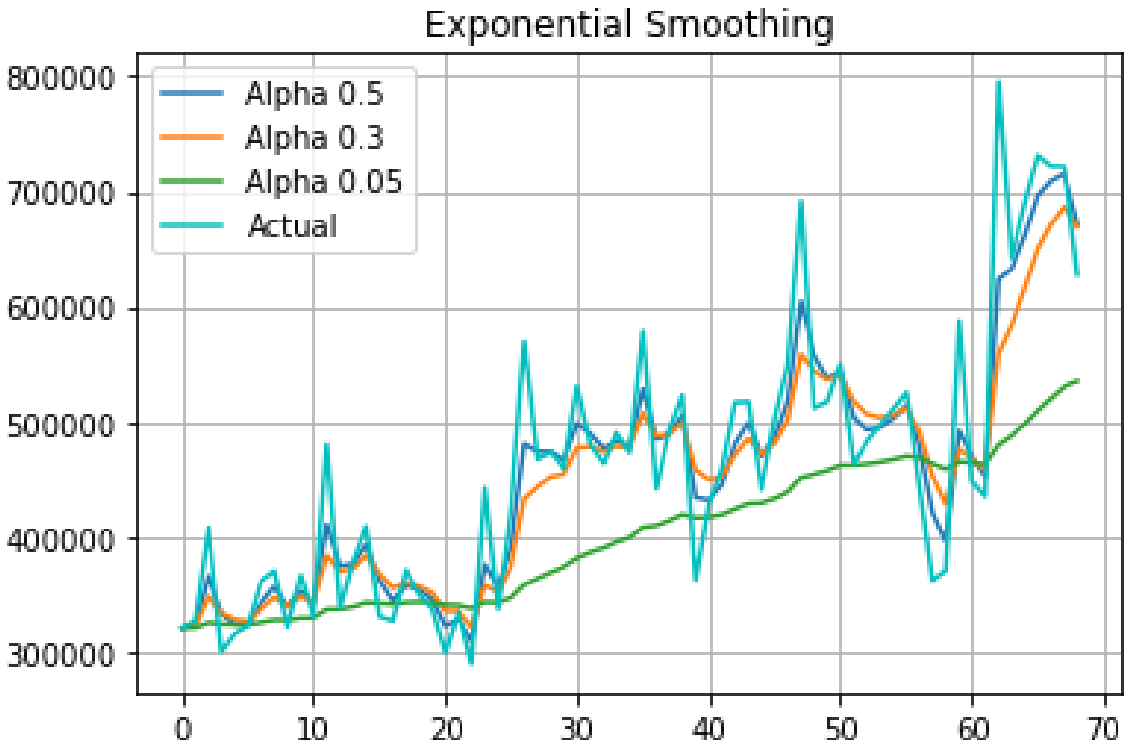

Рисунок 2.14 – Метод експоненційного згладжування

На рисунку 2.15, зображено метод подвійного експоненційного згладжування який приймає два коефіцієнта, α та β. В даному випадку розглядались чотири варіанти комбінацій цих коефіцієнтів з 0.9 та 0.5. Як можна побачити з графіку усі варіанти більш менш точно згладжують графік, але є і неточності, наприклад при обидвох коефіцієнтах які дорівнюють 0.9, прогнозоване значення дуже сильно наближається до аномальних значень, в випадку з даним підприємством – передбачення є сильно заниженим. У випадку коли обидва значення α та β дорівнюють 0.5, то передбачення є на середньому рівні. Тобто можна зробити висновок що коли коефіцієнти різні, то передбачення є найбільш точним.

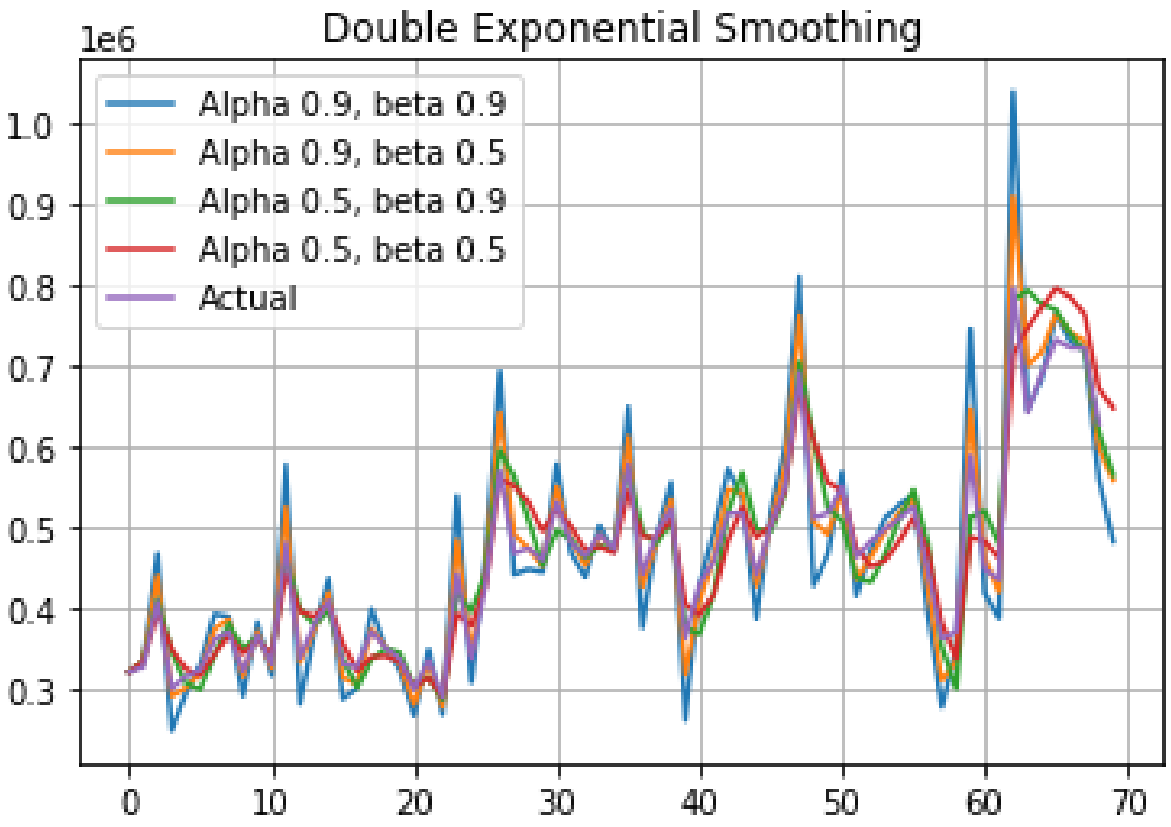

Рисунок 2.15 – Дані по підприємству мережі

Розглянувши одне підприємство мережі, можна зробити висновок що до кожного метода є питання щодо їх точності, тобто в випадках з експоненційними згладжуваннями можна ще підбирати коефіцієнти, щоб вийти на оптимальне значення. Але коли в мережі більше трьох ста підприємств, то немає ніякої можливості розглядати кожне підприємство окремо, бо це дуже трудомістко, тому в подальшому розгляді приймаються стандартні коефіцієнти, які були вибрані експериментальним шляхом, і які мають більш менш однакову точність для всіх підприємств торгової мережі, тобто середні значення для всіх коефіцієнтів це 0.5.

Для метода вагових коефіцієнтів було вибрано наступну систему [0.12, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8], такий розподіл коефіцієнтів означає, що кожен 12 місяць, це сезонність, а отже він має найвищих показник.

Таблиця 2.3- Вихідні дані розрахунку

TT	Moving average	Weighted average	Double exponential smoothing	Triple exponential smoothing	Arima	Real
1	713418	702248	776881	799901	806171	766680
2	1113758	1104119	974653	1129131	1137065	1119856
3	1084210	1085886	918817	1073866	1004896	953331
4	675635	671080	523309	614502	653267	660691
5	1261292	1252345	1314752	1378426	1343840	1378877
6	913945	908383	802721	919765	868416	1005587
7	2458228	2442450	2379850	2618882	2540196	2596477
8	1280988	1273864	1123179	1261421	1272123	1378006
9	881499	872855	803486	928822	888575	957552
10	1517350	1511654	1688474	1810434	1624935	1892492
11	1391068	1358938	1246437	1574586	1425722	1486585
12	1010663	1001136	867948	982456	1021591	987637
13	1827509	1811439	2128723	2218428	2062494	2105926
14	1179835	1177003	1133756	1304644	1124299	1112128
15	1867128	1856883	2030041	2249601	2059874	2299329
16	1686458	1676293	1635606	1793937	1692609	1855750

В таблиці 2.3 наведено розрахунки п’ятьма методами, з ліва на право це згладжування за 12 місяців, вагові коефіцієнти, подвійне далі потрійне експоненційні згладжування, та стандартна ARIMA. Останній стовпець таблиці, це справжні дані за місяць, з якими всі методи і порівнюються.

Таблиця 2.4- Порівняння точності методів

ТТ	Moving average, %	Weighted average	Double exponential smoothing	Triple exponential smoothing	Arima
1	7%	9%	1%	4%	5%
2	1%	1%	15%	1%	2%
3	12%	12%	4%	11%	5%
4	2%	2%	26%	8%	1%
5	9%	10%	5%	0%	3%
6	10%	11%	25%	9%	16%
7	6%	6%	9%	1%	2%
8	8%	8%	23%	9%	8%
9	9%	10%	19%	3%	8%

Для порівняння адекватності отриманих даних візьмемо різницю між прогнозним значенням по кожному методу і дійсні значення

В таблиці 2.4 наведені процент точності кожного метода для кожного підприємства торгової мережі

Таблиця 2.5- Середні значення точності методів прогнозу

Moving average	Weighted average	Double exponential smoothing	Triple exponential smoothing	Arima
7,84%	8,06%	1,19%	5,09%	3,85%
30,93%	31,99%	18,03%	12,57%	10,26%

В таблиці 2.5 наведені середні значення по кожному методу, які розраховувались за двома різними методами, верхнє значення розраховується як відношення суми всіх значень обраного методу, до суми значень дійсних даних, а нижнє значення це середнє значення серед відсотків розбіжності по кожному методу.

З табл. 2.4 можна зробити такі висновки, найточніший метод по сумі за мережу це метод подвійного експоненційного згладжування, розбіжність в 1.19%, найточніший метод за середньою розбіжністю це ARIMA. Цей результат означає що в середньому розбіжність між дійсними даними та прогнозом є 10%. Найгірший результат по двом категоріям це вагові коефіцієнти. Слід зазначити що вагові коефіцієнти можуть давати і кращі результати, тільки якщо підбирати правильні коефіцієнти для кожного підприємства мережі, але такий процес є доволі громістким.

Окремо слід зазначити auto ARIMA, це той же метод, але з автопідбором всіх параметрів.

З табл. 2.6 можна зробити висновок про те, що даний метод дає найбільш точний прогноз, за всі інші методи, але треба зазначити один момент, це те, що відносно часу цей метод рахує довше ніж усі попередні разом, тому що окрім розрахунку прогнозу, рахується також і коефіцієнти.

Таблиця 2.6- Точність методу auto Arima

Auto Arima
2,94%
9,82%

Наукова новизна цією роботи полягає у розробці методики аналізу і прогнозування товарообігу підприємств торгової мережі , базуючись е тільки на фінансово-економічних розрахунках, але й застосовуючи методи інтелектуального аналізу даних для оцінки впливу інших параметрів суб’єкту торгівельної діяльності на його товарообіг.

2.3. Опис програми

Програмна реалізація написана на мові програмування python 3.9.5, у середі розробки PyCharm Community Edition 2022.1.4 та Spyder (Anaconda3)

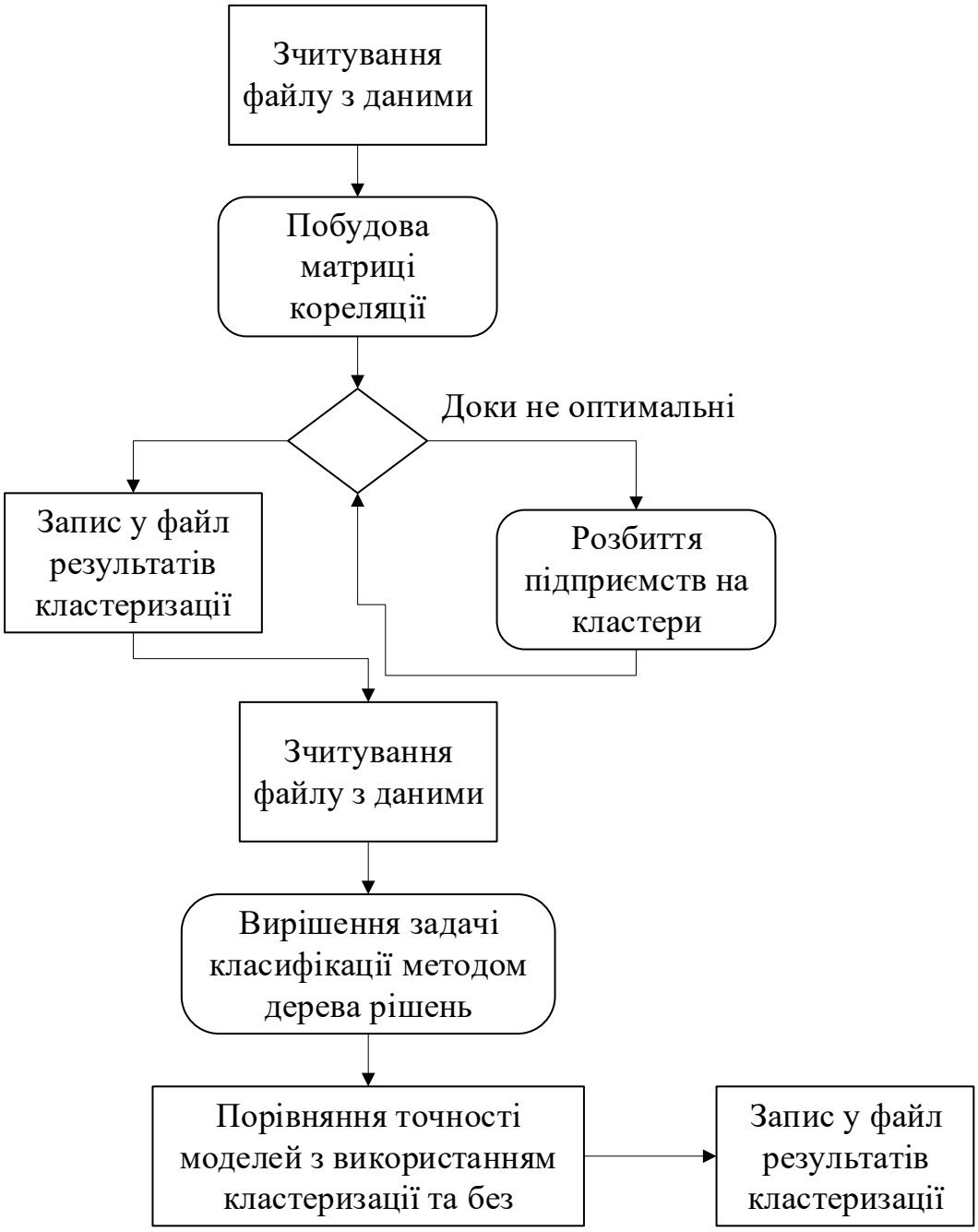

Рисунок 2.16 – Блок схема реалізації кластеризації та класифікації

На рис. 2.16 зображено блок-схему програмної реалізації двох задач, кластеризації та класифікації, для реалізації програми кластеризації (ДОДАТОК А), були використані наступні програмні модулі: **numpy** – математична бібліотека, **pandas** – для роботи з даними, **sklearn.decomposition** – для роботи з методом головних компонент, **sklearn.cluster** – для рішення задачі кластеризації методом k-середніх, **matplotlib.pyplot** – модуль візуалізації даних, **csv** – модуль для запису даних у форматі csv.

Для реалізації програми кластеризації (ДОДАТОК Б) були використані наступні модулі: **pandas** - для роботи з даними, **sklearn** – для побудови і вирішення задачі дерева рішень, **sklearn.model_selection** – цей модуль дозволяє використати функцію для розбиття вибірки на тестову та тренувальну, **matplotlib.pyplot** – модуль візуалізації даних.

На рис. 2.17 наведено блок-схему реалізації задачі прогнозування. Програма складається з трьох частин: перша це головна частина в якій задаються параметри для прогнозування (ДОДАТОК В), друга частина це реалізація методів середнього згладжування, подвійного та потрійного експонційних згладжувань, та ARIMA з заданим значенням(ДОДАТОК Г), третя програма це реалізація ARIMA з автопідбором параметрів.(ДОДАТОК Д)

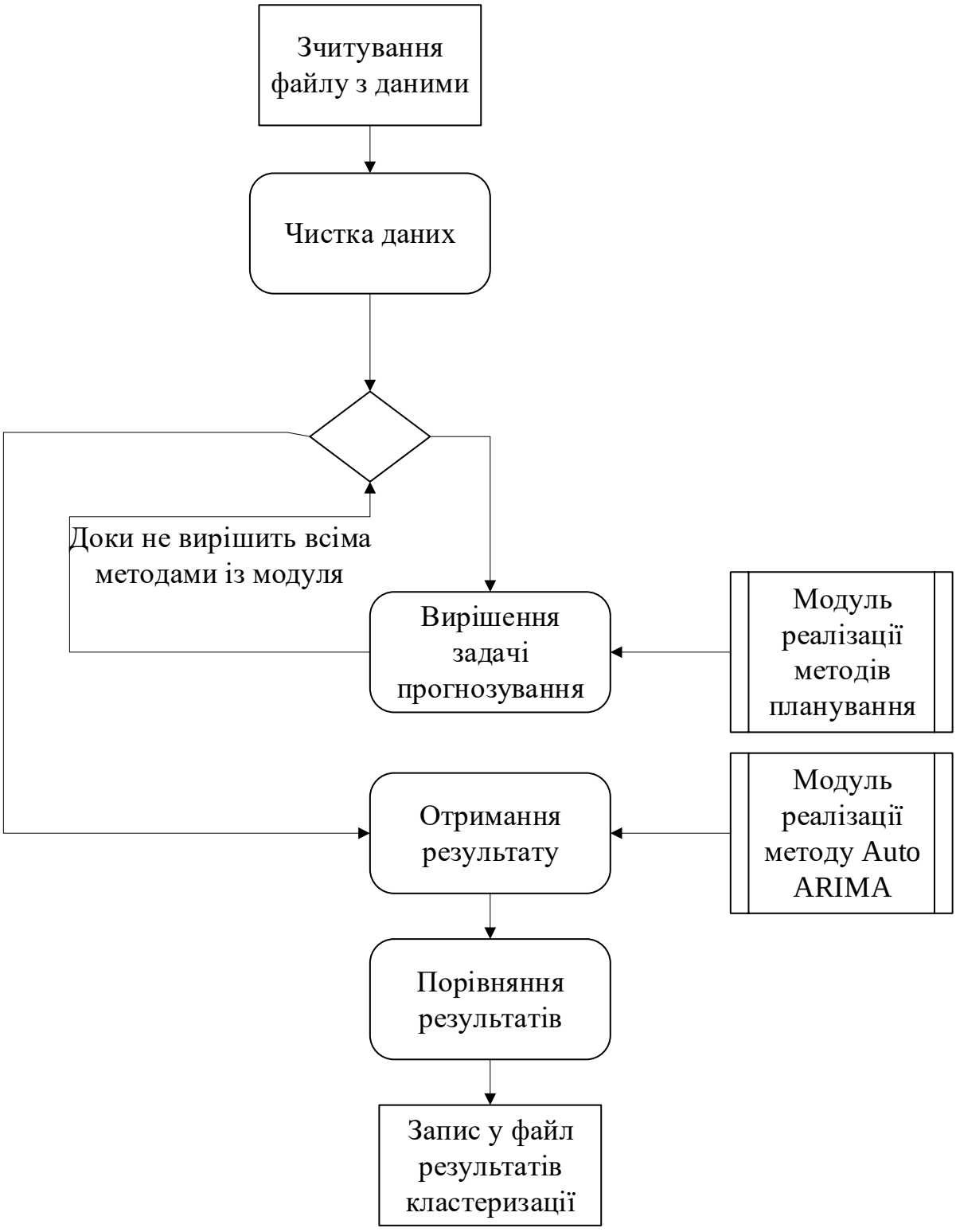

Рис. 2.17 Блок схема реалізації методів прогнозування

ВИСНОВКИ

В роботі запропоновано методику аналізу даних про підприємства торгової мережі, у тому числі відомості про локацію підрозділу, його тип, місткість, фінансові показники, зібрані продовж одного року, з метою виявлення закономірностей, які дозволять прогнозувати товарообіг існуючих підприємств на найближчий часовий період або оцінювати фінансові характеристики нових суб'єктів торгової діяльності.

На основі цих фінансових показників отримано вторинні дані про середньомісячний товарообіг для кожного підприємства, а також фактор денної сезонності.

Проведена кластеризація дозволяє виявити унікальні ознаки кожної групи. Відтак, нові дані про належність підприємства до певного кластеру можна надалі використовувати як вихідний параметр під час класифікації об'єктів мережі для прогнозування сезонності і товарообігу нових підприємств.

Наукова новизна цією роботи полягає у розробці методики аналізу і прогнозування товарообігу підприємств торгової мережі , базуючись е тільки на фінансово-економічних розрахунках, але й застосовуючи методи інтелектуального аналізу даних для оцінки впливу інших параметрів суб'єкту торгівельної діяльності на його товарообіг.

Результати проведених експериментів підтвердили дієвість розробленої методика аналізу і прогнозування сезонності і товарообігу підприємств.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cluster analysis URL: https://en.wikipedia.org/wiki/Cluster_analysis (дата звернення: 20.11.2022).
2. sklearn.cluster.DBSCAN URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html (дата звернення: 20.11.2022).
3. sklearn.cluster.OPTICS URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html (дата звернення: 20.11.2022).
4. Метод головних компонент URL: https://uk.wikipedia.org/wiki/Метод_головних_компонент (дата звернення: 20.11.2022).
5. Задача класифікації URL: https://uk.wikipedia.org/wiki/Задача_класифікації (дата звернення: 20.11.2022).
6. Дерево прийняття рішень URL: https://studfile.net/preview/10096776/ (дата звернення: 20.11.2022).
7. Using the Gini coefficient to evaluate the performance of credit score models URL: https://towardsdatascience.com/using-the-gini-coefficient-to-evaluate-the-performance-of-credit-score-models-59fe13ef420 (дата звернення: 20.11.2022).
8. Time series URL: https://en.wikipedia.org/wiki/Time_series (дата звернення: 20.11.2022).
9. Moving Average (MA): Purpose, Uses, Formula, and Examples URL: https://www.investopedia.com/terms/m/movingaverage.asp (дата звернення: 20.11.2022).
10. Weighted arithmetic mean URL: https://en.wikipedia.org/wiki/Weighted_arithmetic_mean (дата звернення: 20.11.2022).

11. Exponential smoothing URL: https://en.wikipedia.org/wiki/Exponential_smoothing (дата звернення: 20.11.2022).
12. Autoregressive integrated moving average URL: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average (дата звернення: 20.11.2022).

ДОДАТОК А

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jan 31 20:14:20 2022
@author: admin
"""
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import csv
work_data = pd.read_csv("days_of_week_csv.csv", index_col = 0, delimiter=(";"))
#days_of_week_csv
new_data=np.array(work_data)
corr = np.corrcoef(work_data)
#метод главных компонентов
pca = PCA(n_components=2)
pca_TT = pca.fit_transform(work_data)
fig = plt.figure()
plt.plot(pca_TT[:,0],pca_TT[:,1],'bo')
plt.grid()
plt.show()
group = []
temp = []
full_temp = []
for i in range(len(corr)):
```

```
    if i in full_temp:
        continue
    else:
        temp.append(i)

    for j in range(i + 1, len(corr[i])):
        if corr[i][j] >= 0.8 and j not in full_temp:
            temp.append(j)

    for k in range(len(temp)):
        full_temp.append(temp[k])

    group.append(temp.copy())
    temp.clear()
del(full_temp, temp, i, j, k)
big_group = []

other_group = []
for i in range(len(group)):
    if len(group[i]) >= 15: # Значение колличеситва элементов в группе
        big_group.append(group[i])
    else:
        for k in range(len(group[i])):
            other_group.append(group[i][k])
colors = ['red', 'navy', 'lime', 'indigo', 'skyblue', 'gold', 'chocolate', 'crimson']
markers = ['.', 'v', '^', '*', 'p', '1', 'P', 'x']
```

```
for i in range(len(big_group)):
    for j in range(len(big_group[i])):
        plt.plot(pca_TT[big_group[i][j], 0], pca_TT[big_group[i][j], 1], marker = markers[i], color = colors[i])
for i in range(len(other_group)):
    plt.plot(pca_TT[other_group[i],0], pca_TT[other_group[i],1], marker = 'p', color = 'black')
plt.grid()
plt.show()


big_group_mean = []


temp = []


for i in range(len(big_group)):
    for j in range(len(new_data[i])):
        temp.append(np.median(new_data[big_group[i],j]))#!!!
    big_group_mean.append(temp.copy())
    temp.clear()


dif_other_corr = []
for i in other_group:
    for j in range(len(big_group_mean)):
        temp.append(np.corrcoef(new_data[i],big_group_mean[j])[0][1])
    dif_other_corr.append(temp.copy())
    temp.clear()
```

```
for i in range(len(other_group)):
    big_group[np.argmax(dif_other_corr[i])].append(other_group[i])



for i in range(len(big_group)):
    for j in range(len(big_group[i])):
        plt.plot(pca_TT[big_group[i][j],0], pca_TT[big_group[i][j],1], marker = markers[i], color = colors[i])
plt.grid()
plt.show()
number_of_group = []
new_big_group = []
special_case = []
for i in range(len(big_group)):
    new_big_group.append(number_of_group.copy())
n = 0
while(n <= 10):
    big_group_mean = []


    temp = []


    if n == 0:
        big_group_pre = big_group.copy()
    else:
        big_group_pre = new_big_group.copy()


    for i in range(len(big_group)):
```

```
        for j in range(len(new_data[i])):
            temp.append(np.median(new_data[big_group_pre[i],j]))
        big_group_mean.append(temp.copy())
        temp.clear()
    number_of_group = []

    new_big_group = []

    for i in range(len(big_group)):
        new_big_group.append(number_of_group.copy())

    for i in range(len(big_group_pre)):
        for j in range(len(big_group_pre[i])):
            for k in range(len(big_group_mean)):

temp.append(np.corrcoef(new_data[big_group_pre[i][j]],big_group_mean[k])[0][1])
            #new_big_group.append(temp.copy())
            if np.max(temp) <= 0.8:
                special_case.append(big_group_pre[i][j])
            else:
                new_big_group[np.argmax(temp)].append(big_group_pre[i][j])
            temp.clear()

    for i in special_case:
        for k in range(len(big_group_mean)):
            temp.append(np.corrcoef(new_data[i],big_group_mean[k])[0][1])
        if np.max(temp) > 0.8:
```

```
            new_big_group[np.argmax(temp)].append(i)
            special_case.remove(i)
        temp.clear()
    n = n + 1
for i in range(len(new_big_group)):
    for j in range(len(new_big_group[i])):
        plt.plot(pca_TT[new_big_group[i][j], 0], pca_TT[new_big_group[i][j], 1], marker = markers[i], color = colors[i])


    for i in range(len(special_case)):
        plt.plot(pca_TT[special_case[i], 0], pca_TT[special_case[i], 1], marker = 'o', color = 'k')
plt.grid()
plt.show()


to_csv_data = []
print('Кластер: TT: пн вт ср чт пт сб вс')
for i in range(len(new_big_group)):
    for j in range(len(new_big_group[i])):
        print(f'{i}; {work_data.index[new_big_group[i][j]]}; {new_data[new_big_group[i][j]]}')
        to_csv_data.append([ int(work_data.index[new_big_group[i][j]]), int(i), new_data[new_big_group[i][j]][0], new_data[new_big_group[i][j]][1], new_data[new_big_group[i][j]][2], new_data[new_big_group[i][j]][3], new_data[new_big_group[i][j]][4], new_data[new_big_group[i][j]][5], new_data[new_big_group[i][j]][6]])
 for i in range(len(special_case)):
    print(f'{len(big_group)}; { work_data.index[special_case[i]]}; {new_data[special_case[i]]}')
```

```
    to_csv_data.append([ work_data.index[special_case[i]], len(big_group),
new_data[special_case[i]][0], new_data[special_case[i]][1],
new_data[special_case[i]][2], new_data[special_case[i]][3],
new_data[special_case[i]][4], new_data[special_case[i]][5],
new_data[special_case[i]][6]])


with
open('D:\Дима\Системный_аннализ\Диплом\Classififcation\cluster_new_out_data.csv
', 'w', newline='') as fh:
    field_names = ['Number_of_TT', 'Number_of_group', 'Monday',
                   'Tuesday', 'Wednesday', 'Thursday',
                   'Friday', 'Saturday', 'Sunday']
    writer = csv.DictWriter(fh, fieldnames=field_names, delimiter=';')
    writer.writeheader()
    for row in to_csv_data:
        writer.writerow({field_names[0]: row[0],
                         field_names[1]: row[1],
                         field_names[2]: row[2],
                         field_names[3]: row[3],
                         field_names[4]: row[4],
                         field_names[5]: row[5],
                         field_names[6]: row[6],
                         field_names[7]: row[7],
                         field_names[8]: row[8]})
```

ДОДАТОК Б

```
import pandas as pd
from sklearn import tree
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
def series_cutting(series):
    X, y = series.values[:, 1:], series.values[:, 0]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.15, random_state=100)

    return X, y, X_train, X_test, y_train, y_test

def decision_tree(series):
    X, y = series.values[:, 1:], series.values[:, 0]
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(X, y)
    #clf.predict([[65000, 150, 95, 1,      5,       2,       1]])
    #tree.plot_tree(clf)


    plt.figure(figsize=(40,32))  # set plot size (denoted in inches)
    tree.plot_tree(clf, fontsize=18)
    plt.show()

    return clf
```

```
def predic_series(predict_series, decision_of_series):
    predict = []
    for line in range(len(predict_series)):
        predict.append(int(decision_of_series.predict([predict_series.values[line]])[0]))
        #print(predict_series.values[line])
    return predict

def main():
    dataset = pd.read_csv('clasification_format_training.csv', index_col=0, delimiter=';')
    dataset_predict = pd.read_csv('clasification_format.csv', index_col=0, delimiter=';')
    dataset_c = pd.read_csv('clasification_format_c.csv', index_col=0, delimiter=';')
    #dataset = pd.get_dummies(dataset)

    tree_solved = decision_tree(dataset)

    predict = predic_series(dataset_predict, tree_solved)

    dataset_predict.insert(loc=0, column='Format_code', value=predict)
    dataset = pd.concat([dataset, dataset_c, dataset_predict])

    cols = list(dataset)
    cols[1], cols[0] = cols[0], cols[1]
    dataset = dataset.reindex(columns=cols)
    dataset = dataset.sort_values(by='Number')

    return dataset
```

```
def nan_value(series):
    drop_list = []
    dataset_index = series.index
    for group in dataset_index:
        if str(series[group]) == 'nan':
            series[group] = -1
            drop_list.append(group)
    return series, drop_list
if __name__ == "__main__":
    dataset = main()
    #X, y, X_train, X_test, y_train, y_test = series_cutting(dataset)
    dataset.to_csv('dataset_without_cluster_group.csv', sep=';')

    cluster = pd.read_csv('cluster_new_out_data.csv', sep=';', index_col=(0))
    #cluster = nan_value(cluster['Number_of_group'])
    dataset['Number_of_group'] = cluster['Number_of_group']
    _, drop_group = nan_value(dataset['Number_of_group'])
    dataset = dataset.drop(drop_group)
    dataset.to_csv('dataset_with_cluster_group.csv', sep=';')
```

ДОДАТОК В

```
from planning import prepare_data, moving_average, weighted_average
from planning import double_exponential_smoothing, plot_exponential_smoothing
from holt_winters import HoltWinters
from arimapy import arima
import pandas as pd
import csv
import warnings
warnings.filterwarnings('ignore')
if __name__=='__main__':
    dataset = pd.read_csv('raw_data.csv', index_col=0, delimiter=';').T
    dataset = prepare_data(dataset.copy())

    dataset_columns = dataset.columns
    dataset_index = dataset.index

    forecast_TT_dict = dict()
    predict_method_dict = dict()

    print('{:>5}|{:^12}|{:^12}|{:^12}|{:^12}|{:^12}|'.format('TT number', 'Moving
average', 'Weighted average', 'Double exponential smoothing', 'Triple exponential
smoothing', 'Arima'))

    for TT in dataset_columns:
        ma = moving_average(dataset[TT], 12)
        wa = weighted_average(dataset[TT], [0.12,
                                            0.08,
```

```
                0.08,
                0.08,
                0.08,
                0.08,
                0.08,
                0.08,
                0.08,
                0.08,
                0.08,
                0.08,])

des = double_exponential_smoothing(dataset[TT], 0.5, 0.5)[-1]

holtwinters = HoltWinters(dataset[TT], 12, 0.5, 0.5, 0.5, 1)
holtwinters.triple_exponential_smoothing()
tes = holtwinters.result[-1]

_, a = arima(dataset[TT])

predict_method_dict = {'Moving average': round(ma, 2),
                       'Weighted average': round(wa, 2),
                       'Double exponential smoothing': round(des, 2),
                       'Triple exponential smoothing': round(tes, 2),
                       'Arima': round(a, 2)}

forecast_TT_dict.update({f'{TT}': predict_method_dict.copy()})
```

```
        print('{:>5}|{:^12}|{:^12}|{:^12}|{:^12}|{:^12}\n'.format(TT, round(ma, 2),
round(wa, 2), round(des, 2), round(tes, 2), round(a, 2)))


    with open('predict.csv', 'w', newline='') as fh:
        field_names = ['TT', 'Moving average', 'Weighted average',
                       'Double exponential smoothing',
                       'Triple exponential smoothing', 'Arima']
        writer = csv.DictWriter(fh, fieldnames=field_names)
        writer.writeheader()
        for TT in dataset_columns:
            writer.writerow({field_names[0]: f'{TT}',
                             field_names[1]:
f'{forecast_TT_dict.get(f"{str(TT)}").get(f"{field_names[1]}")}',
                             field_names[2]:
f'{forecast_TT_dict.get(f"{str(TT)}").get(f"{field_names[2]}")}',
                             field_names[3]:
f'{forecast_TT_dict.get(f"{str(TT)}").get(f"{field_names[3]}")}',
                             field_names[4]:
f'{forecast_TT_dict.get(f"{str(TT)}").get(f"{field_names[4]}")}',
                             field_names[5]:
f'{forecast_TT_dict.get(f"{str(TT)}").get(f"{field_names[5]}")}'})
```

ДОДАТОК Г

```
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from sklearn.model_selection import TimeSeriesSplit
from holt_winters import HoltWinters
def show_all(series):
    #dataset_columns = series.columns
    dataset_index = series.index
    #for TT in dataset_columns:
    for index, date in enumerate(dataset_index):
        print(f'{index}: {date}, {series[date]}')
def prepare_data(series):
    dataset_columns = series.columns
    dataset_index = series.index
    for TT in dataset_columns:
        for date in dataset_index:
            if str(series[TT][date]) == 'nan':
                series[TT][date] = 0
    return series


def plotMovingAverage(series, n):
    """
```

```
    series - dataframe with timeseries
    n - rolling window size
    """
    rolling_mean = series.rolling(window=n).mean()
    # При желании, можно строить и доверительные интервалы для сглаженных значений
    rolling_std =  series.rolling(window=n).std()
    upper_bond = rolling_mean + 1.96 * rolling_std
    lower_bond = rolling_mean - 1.96 * rolling_std
    #plt.figure(figsize=(120,100))
    plt.title(f"Moving average\n window size = {n}")
    plt.plot(rolling_mean, "g", label="Rolling mean trend")
    plt.plot(upper_bond, "r--", label="Upper Bond / Lower Bond")
    plt.plot(lower_bond, "r--")
    plt.plot(series[n:], label="Actual values")
    plt.legend(loc="upper left")
    plt.grid(True)
    plt.show()


def moving_average(series, n):
    return np.average(series[-n:])
def weighted_average(series, weights):
    result = 0.0
    weights.reverse()
    for index, weight in enumerate(weights):
        result += series[- index - 1] * weight
```

```
    return result
def exponential_smoothing(series, alpha):
    result = [series[0]] # first value is same as series
    for n in range(1, len(series)):
        result.append(alpha * series[n] + (1 - alpha) * result[n - 1])
    return result
def double_exponential_smoothing(series, alpha, beta):
    result = [series[0]]
    for n in range(1, len(series) + 1):
        if n == 1:
            level, trend = series[0], series[1] - series[0]
        if n >= len(series): # прогнозируем
            value = result[-1]
        else:
            value = series[n]
        last_level, level = level, alpha * value + (1 - alpha) * (level + trend)
        trend = beta * (level - last_level) + (1 - beta) * trend
        result.append(level + trend)
    return result
def timeseriesCVscore(x, data):
    # вектор ошибок
    errors = []
    values = data.values
    alpha, beta, gamma = x
    # задаём число фолдов для кросс-валидации
    tscv = TimeSeriesSplit(n_splits=2)
```

```
    # идем по фолдам, на каждом обучаем модель, строим прогноз на отложенной выборке и считаем ошибку
    for train, test in tscv.split(values):
        model = HoltWinters(series=values[train], slen = 12, alpha=alpha, beta=beta, gamma=gamma, n_preds=len(test))
        model.triple_exponential_smoothing()
        predictions = model.result[-len(test):]
        actual = values[test]
        error = mean_squared_error(predictions, actual)
        errors.append(error)
    # Возвращаем средний квадрат ошибки по вектору ошибок
    return np.mean(np.array(errors))
def plot_exponential_smoothing(series, alpha):
    plt.style.context('seaborn-white')
    for alpha in [alpha, 0.3, 0.05]:
        plt.plot(exponential_smoothing(series, alpha), label="Alpha {}".format(alpha))
    plt.plot(series.values, "c", label = "Actual")
    plt.legend(loc="best")
    plt.axis('tight')
    plt.title("Exponential Smoothing")
    plt.grid(True)
    plt.show()


def plot_double_exponential_smoothing(series, alpha, beta):
    plt.style.context('seaborn-white')
    for alpha in [0.9, alpha]:
```

```
        for beta in [0.9, beta]:
            plt.plot(double_exponential_smoothing(series, alpha, beta), label="Alpha {}, beta {}".format(alpha, beta))
    plt.plot(series.values, label = "Actual")
    plt.legend(loc="best")
    plt.axis('tight')
    plt.title("Double Exponential Smoothing")
    plt.grid(True)
    plt.show()



def plotHoltWinters(data, model):
    Anomalies = np.array([np.NaN] * len(data))
    Anomalies[data.values < model.LowerBond] = data.values[
        data.values < model.LowerBond
    ]
    #plt.figure(figsize=(25, 10))
    plt.plot(model.result, label="Model")
    plt.plot(model.UpperBond, "r--", alpha=0.5, label="Up/Low confidence")
    plt.plot(model.LowerBond, "r--", alpha=0.5)
    plt.fill_between(
        x=range(0, len(model.result)),
        y1=model.UpperBond,
        y2=model.LowerBond,
        alpha=0.5,
        color="grey",
    )
```

```
    plt.plot(data.values, label="Actual")
    plt.plot(Anomalies, "o", markersize=10, label="Anomalies")
    plt.axvspan(len(data) - 128, len(data), alpha=0.5, color="lightgrey")
    plt.grid(True)
    plt.axis("tight")
    plt.legend(loc="best", fontsize=13)


if __name__=='__main__':
    dataset = pd.read_csv('raw_data.csv', index_col=0, delimiter=';').T
    dataset = prepare_data(dataset.copy())
    #new = map(lambda x: 0 if x == 'nan', dataset)
    tt_number = 107
    print(f'Moving average: {moving_average(dataset[tt_number], 12)}')
    plotMovingAverage(dataset[tt_number], 12)
    print(f'Weighted average: {weighted_average(dataset[tt_number], [0.24, 0.19, 0.17, 0.15, 0.15, 0.10])}')
    #print(exponential_smoothing(dataset[tt_number], 0.5))
    plot_exponential_smoothing(dataset[tt_number], 0.5)
    double_exp_smoot = double_exponential_smoothing(dataset[tt_number], 0.9, 0.5)[len(double_exponential_smoothing(dataset[tt_number], 0.5, 0.5)) - 1]
    #print(double_exponential_smoothing(dataset[tt_number], 0.5, 0.5))
    print(f'Double exp smoothing: {double_exp_smoot}')
    plot_double_exponential_smoothing(dataset[tt_number], 0.5, 0.5)
    #plt.plot(dataset[109])
    holtwinters = HoltWinters(dataset[tt_number], 12, 0.5, 0.5, 0.5, 1)
    holtwinters.triple_exponential_smoothing()
```

```
    result = holtwinters.result

    print(f'Triple exp smoothing: {result[len(result) - 1]}')

    #params = timeseriesCVscore([0.5, 0.5, 0.5], dataset[tt_number])
    '''
    x = [0.5, 0.5, 0.5]
    # Минимизируем функцию потерь с ограничениями на параметры
    opt = minimize(timeseriesCVscore, x0=x, args=(dataset[tt_number]),
method="TNC", bounds = ((0, 1), (0, 1), (0, 1)))

    alpha_final, beta_final, gamma_final = opt.x
    print(alpha_final, beta_final, gamma_final)

    holtwinters = HoltWinters(dataset[tt_number], 12, alpha_final, beta_final,
gamma_final, 1)
    holtwinters.triple_exponential_smoothing()
    result = holtwinters.result

    print(f'Triple exp smoothing fitted data: {result[-1]}')
    '''
    dataset = dataset[tt_number]
    #dataset = dataset.resample('W')
    dataset.plot(figsize=(12,6))
    total = dataset.describe()
```

ДОДАТОК Д

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pmdarima.arima import auto_arima
from time import time

import warnings

warnings.filterwarnings('ignore')

INFLATION = 12 / 12
INFLATION = -100

def prepare_data(series):
    dataset_columns = series.columns
    dataset_index = series.index
    for TT in dataset_columns:
        for date in dataset_index:
            if str(series[TT][date]) == 'nan':
                series[TT][date] = 0
    return series


def prepare_data_to_nan(series):
    dataset_columns = series.columns
```

```
    dataset_index = series.index
    for TT in dataset_columns:
        for date in dataset_index:
            if str(series[TT][date]) == 0:
                series[TT][date] = 'nan'
    return series


def moving_average(series, n):
    return np.average(series[-n:])


def double_exponential_smoothing(series, alpha, beta):
    result = [series[0]]
    for n in range(1, len(series) + 1):
        if n == 1:
            level, trend = series[0], series[1] - series[0]
        if n >= len(series):  # прогнозируем
            value = result[-1]
        else:
            value = series[n]
        last_level, level = level, alpha * value + (1 - alpha) * (level + trend)
        trend = beta * (level - last_level) + (1 - beta) * trend
        result.append(level + trend)
    return result
```

```
def auto_arima_forecast(array):
    arima_model = auto_arima(array, start_p=0, d=1, start_q=0,
                             max_p=3, max_d=3, max_q=3, start_P=0,
                             D=1, start_Q=0, max_P=3, max_D=3,
                             max_Q=3, error_action='warn', trace=True,
                             suppress_warnings=True, stepwise=True,
                             random_state=15, n_fits=12, return_valid_fits=False)

    return arima_model


def find_season_value(seasons, TT, month):
    for i in range(len(seasons)):
        if seasons.values[i][0] == TT and seasons.values[i][1] == month:
            return seasons.values[i][2]


def forecast(series, seasons, month, reduction):
    forecast_data_dict = {}
    dataset_columns = series.columns
    dataset_index = series.index
    for TT in dataset_columns:
        print(TT, reduction)
        if 1 <= series[TT].count() - series[TT].value_counts(sort=False).values[0]:
            # des = round(double_exponential_smoothing(series[TT], 0.73, 0.73)[-1],
reduction)
```

```
            arm = round(auto_arima_forecast(series[TT]).predict(1)[len(series[TT])])
        else:
            # des = round(double_exponential_smoothing(series[ANALOGS[TT]], 0.73, 0.73)[-1], reduction)
            arm = round(auto_arima_forecast(series[ANALOGS[int(TT)]]).predict(1)[len(series[TT])])

        if find_season_value(seasons, TT, month) is None:
            season = find_season_value(seasons, ANALOGS[int(TT)], month)
        else:
            season = find_season_value(seasons, TT, month)

        #result = round(arm * season, reduction)
        result = round(arm, reduction)
        print(f'{arm}: {season}: {result}')

        forecast_data_dict.update({TT: result})
    return forecast_data_dict


def add_to_data_frame(add_dict, frame, date):
    frame.loc[date] = 0
    dataset_columns = frame.columns
    for TT in dataset_columns:
        TT = int(TT)
        frame[TT][date] = add_dict[TT] + (1 + (INFLATION / 100))
    return frame
```

```
def main():
    timer = time()
    #dataset_checks = prepare_data(pd.read_csv('Checks_for_year.csv', index_col=0, delimiter=';').T)
    #dataset_srh = prepare_data(pd.read_csv('SRH_for_year.csv', index_col=0, delimiter=';').T)  # .T
    #dataset_TO = dataset_checks * dataset_srh
    dataset_TO = prepare_data(pd.read_csv('raw_data.csv', index_col=0, delimiter=';').T)

    season_check = pd.read_csv('Season_all_retail_check.csv', delimiter=';')
    season_srh = pd.read_csv('Season_all_retail_srh.csv', delimiter=';')
    season_TO = season_check.copy()
    season_TO['Value'] = season_check['Value'] * season_srh['Value']

    dates = ['01.10.2022']

    for date in dates:
        _, month, _ = date.split('.')
        '''
        timer = time()
        main_checks = forecast(dataset_checks, season_check, int(month), 0)
        #print(f'Done main_checks: {round(time() - timer, 4)}')
        dataset_checks = add_to_data_frame(main_checks, dataset_checks, date)

        #timer = time()
```

```
        main_srh = forecast(dataset_srh, season_srh, int(month), 2)
        print(f'Done: {round(time() - timer, 4)}')
        dataset_srh = add_to_data_frame(main_srh, dataset_srh, date)

        dataset_TO = dataset_checks * dataset_srh
        '''

        main_TO = forecast(dataset_TO, season_TO, int(month), 2)
        dataset_TO = add_to_data_frame(main_TO, dataset_TO, date)

        #dataset_TO.to_csv('TO_for_new_year_november.csv', sep=';')
        dataset_TO.to_csv('raw_data_predict.csv', sep=';')
        # dataset_checks.to_csv('Check_for_new_year.csv', sep=';')
        # dataset_srh.to_csv('SRH_for_new_year.csv', sep=';')
    print(f'Done: {round(time() - timer, 4)}')


if __name__ == '__main__':
    main()
```