DOI 10.31891/2307-5732-2024-331-38
УДК 004.896:004.42

ЛЬОВКІН ВАЛЕРІЙ
Національний університет «Запорізька політехніка»
https://orcid.org/0000-0002-6890-2807
e-mail: vliovkin@gmail.com

# ОСОБЛИВОСТІ ПРОГРАМУВАННЯ АЛГОРИТМІВ І СТРУКТУР ДАНИХ ПРИ ПРОГНОЗУВАННІ АВТОМОБІЛЬНОГО ТРАФІКУ

*У роботі проаналізовано можливі варіанти організації структур і сховищ даних при побудові системи, яка передбачає моніторинг і прогнозування автомобільного трафіку. Враховано особливості реалізації цих структур даних мовою Python на основі використання пакетів NumPy, Pandas. Приділено особливу увагу використанню переліків, багатовимірних масивів та словників у якості структур даних та файлів різного типу і баз даних в якості сховищ даних. Створено вебдодаток на основі використання фреймворку розробки вебдодатків Django. Запропоновано і представлено відповідну структуру реляційної бази даних для підтримки організації сховища. Розроблений вебдодаток дозволив визначити час виконання різних сценаріїв використання структур даних на основі відповідних алгоритмів у складі програмної системи. Експерименти виконувались з заповненням сховищ даних даними спостережень за автомобільним трафіком за 59 станціями з 1.01.2019 року по 30.09.2022 року в місті Мадрид. Базові сценарії відслідковування включають читання даних, відбір даних. Прогнозування автомобільного трафіку заплановано виконувати для умов обмеженості ресурсів. Передбачено підготовку даних як у вигляді часових рядів, так і для реалізації необхідного в такому випадку відбору релевантних станцій. Враховано можливості створення індексів для пришвидшення пошуку. Надано рекомендації щодо вибору структур даних для підтримки різних варіантів застосування таких програмних систем. Попередньо структуровані дані у вигляді переліків і словників за умов швидкого доступу є достатньо ефективною основою для організації вибору і підготовки даних для виконання відповідних функцій у програмі. Загалом у випадку необхідності читання даних з файлів перед виконанням відповідних операцій звернення до бази даних є більш доцільним варіантом підтримки таких функцій.*

*Ключові слова: алгоритм, структура даних, програмування, програмне забезпечення, база даних, прогнозування автомобільного трафіку.*

LOVKIN VALERII
National University «Zaporizhzhia Polytechnic»

## FEATURES OF PROGRAMMING ALGORITHMS AND DATA STRUCTURES FOR TRAFFIC FORECASTING

*Development of software system for monitoring and forecasting of vehicle traffic is considered in the paper. Possible ways of data structures and storages organization for such a system were analysed. Implementation features of these data structures for Python programming language were considered based on the NumPy, Pandas packages. Special attention was paid to the application of lists, multidimensional arrays and dictionaries as data structures and files of different types as data storages. Web application was developed based on the Django web development framework. The corresponding database structure was proposed and presented for data storage organization. The developed web application supported calculation of time necessary for realization of different scenarios. These scenarios define implementation of data structures based on the corresponding algorithms in software system. Data storages were filled with observation data, accumulated in Madrid through the period from 1.01.2019 to 30.09.2022. 59 stations were used to measure vehicle traffic. Base monitoring scenarios include data reading and selection. Vehicle traffic forecasting is done under resource constraints. Data is prepared in the form of time series, as well as in the form necessary for selection of informative stations. Search speed acceleration was provided through the creation of indexes. Recommendations were given for design decision making concerning choice of data structures for support of different variants of software system application. Data, previously structured as lists and dictionaries, makes an effective foundation for data selection and preparation. If data has to be read from a file at the beginning of a corresponding operation, then software communication with database is a more appropriate approach.*

*Keywords: algorithm, data structure, programming, software, database, traffic forecasting.*

### Постановка проблеми

Рух автомобілів є активною складовою життя будь-якого міста, який важливо враховувати на різних етапах пов’язаних процесів. Збільшення або зменшення автомобільного трафіку призводить до впливу на ці пов’язані процеси та є результатом інших пов’язаних процесів. Наприклад, збільшення трафіку є результатом збільшення ділової активності в місті, але одночасно і додатковим джерелом забруднення повітря у місті, потенційним чинником впливу на активність громадян. Так, якщо не здійснюється регулювання кількості автомобілів, враховуючи створену для них інфраструктуру, якщо інфраструктура не узгоджується в свою чергу з наявними і запланованими потребами, це може призводити до неефективного використання часу (як ділового, так і персонального) громадянами, компаніями.

Відслідковувати трафік потрібно як стратегічно для розуміння наявних процесів, визначення способів регулювання трафіку, так і тактично: визначати, як можна вплинути на трафік у короткій перспективі для того, щоб врегулювати наявні тенденції, а також як поточна або очікувана ситуація може вплинути на пов’язані процеси та чи потрібно для цього заподіяти певний керуючий вплив, щоб негативний вплив трафіку знизити. Усе це говорить про те, що важливими є інструменти не тільки моніторингу поточного або історичного стану, але і інструменти прогнозування. Довгострокове прогнозування потрібне при плануванні процесів у місті і пов’язане зі стратегічними процесами, а ось короткострокове дозволяє визначити той керуючий вплив, який можна або необхідно заподіяти в тій чи іншій ситуації. Тож у підсумку це вказує на

важливість об'єднання в одній програмній системі інструментів відслідковування автомобільного трафіку та його прогнозування.

Важливо також окреслити область застосування такої системи, адже це може значно впливати на процеси використання, на необхідні функції та акцент використання таких функцій. Зважаючи на вплив трафіку на інші процесі у місті, в даній роботі вирішено зробити акцент на створенні основ програмної системи, яка може використовуватися не централізовано міською радою (або не тільки таким чином), а децентралізовано для різних організацій, в діяльності яких можуть використовуватися дані стосовно автомобільного трафіку. Наприклад, сервіс з надання медичних послуг може використовувати ці дані для визначення рекомендацій для власних пацієнтів. Однак, особливістю такого використання є те, що наявні ресурси для роботи такої програмної системи можуть значно відрізнятися. Тому при реалізації програмної системи моніторингу і прогнозування автомобільного трафіку необхідно забезпечити ефективність обраних проєктних рішень щодо обраних структур даних, використаних алгоритмів, на що і направлена дана робота.

**Аналіз останніх джерел**

Оскільки при створенні описаної системи центральним механізмом є прогнозування автомобільного трафіку, то був проведений аналіз останніх джерел, концентруючись на вирішенні цієї проблеми. У підсумку основні результати були зведені до робіт [1–8]. При цьому роботи [1–7] мають дещо різні інструменти розв'язання проблеми прогнозування автомобільного трафіку, але загальну спільну ідею, яка полягає в тому, щоб виконувати прогнозування на основі згорткових нейронних мереж, найчастіше графічних, та об'єднанні цієї моделі з рекурентними нейронними мережами. Тоді дані об'єднуються з різних станцій спостереження, дозволяючи враховувати співвідношення, які пов'язані зі взаємним розташуванням цих станцій. Проте це потребує великої кількості ресурсів, що охоплюють широкий набір станцій спостереження, що не завжди наявно.

Зважаючи на окреслену необхідність прогнозування автомобільного трафіку та потребу в реалізації прогнозування в умовах обмежених ресурсів, важливою роботою є робота [8], в якій представлено метод прогнозування автомобільного трафіку. Цей метод враховує необхідність роботи в умовах обмежених ресурсів. Модель прогнозування побудована на основі двонаправленої довгої короткострокової пам'яті. Саме ж прогнозування виконується не на основі всіх наявних станцій спостереження за трафіком, а на основі обраних станцій для створення моделі.

**Метою роботи є** дослідження особливостей програмної реалізації системи моніторингу і прогнозування автомобільного трафіку стосовно ефективності прийнятих рішень для роботи в умовах обмежених ресурсів.

**Виклад основного матеріалу**

Основні функції програмної системи моніторингу і прогнозування автомобільного трафіку пов'язані з наданням у різному представленні накопичених даних про трафік. Вони можуть стосуватися демонстрації трафіку за певний період на певній станції, включати порівняння трафіку, однак все це так чи інакше пов'язано з функціями інформаційної системи. Проте, як було визначено в постановці проблеми, важливо не тільки накопичувати дані за минулі періоди, але і виконувати прогнозування майбутнього стану. Саме це і має дозволити використовувати таку програмну систему в умовах різних організацій, які мають потребу в даних, пов'язаних зі станом міста.

Для виконання безпосередньо прогнозування автомобільного трафіку вирішено використовувати в підсумку метод [8], зважаючи на його придатність у окреслених в роботі умовах. Проте цей метод має певні особливості, які пов'язані з тим, що він передбачає не тільки навчання і використання моделі, але і визначення релевантних для прогнозування станцій. Тому навіть якщо навчання моделі може бути рідкісним процесом, але визначення релевантних станцій має виконуватися, по-перше, раніше, адже потребує часу, а по-друге, результати даного процесу можуть бути свідченням необхідності змін у самій моделі прогнозування. Тоді визначення релевантних станцій має бути більш частим процесом, а фактично може виконуватися в режимі моніторингу: за накопичення постійних змін має змінюватися модель. Тож стосовно програмної системи моніторингу автомобільного трафіку це вказує на збільшення кількості запитів, які не тільки пов'язані з витяганням певних даних зі сховища, але і з формуванням часових рядів. Ці вимоги стосовно роботи з даними мають бути досліджені в цій роботі стосовно ефективності використаних засобів для структурування даних.

Для простоти і зручності використання програмної системи було вирішено реалізувати її у вигляді вебдодатку на основі використання фреймворку Django. Такий вибір обумовлений тим, що програмна система вимагає, як було описано, ефективної роботи з даними, підтримки прогнозування автомобільного трафіку на основі двонаправленої короткочасної пам'яті, відбору ознак на основі методу Random Forest, наявності універсального доступу для роботи в умовах обмежених ресурсів. Об'єднати всі ці механізми разом дозволяє мова програмування Python, а фреймворк Django дозволяє створювати актуальні вебдодатки, що має сприяти практичному використанню створеної системи.

Зважаючи на необхідність проведення експериментів, у цій роботі було окремо створено базу даних, структура якої виключно пов'язана з забезпеченням основ роботи з даними щодо моніторингу та прогнозування трафіку. В якості основи для організації бази даних використано систему керування базами даних SQLite. Враховуючи описану концепцію застосування програмної системи, значної початкової необхідності проведення навантажувального тестування не було виявлено, тому цієї системи керування

базами даних достатньо для проведення експериментів. Структура бази даних (рис. 1) складається з 3 базових таблиць:

– trafficapp_station, що містить дані станцій спостереження, за якими збираються дані про автомобільний трафік, включаючи номер станції, позначення, розташування на мапі за широтою та довготою;

– trafficapp_stationpair, що містить дані пар станцій, які є релевантними, включаючи номер запису, станцію, для якої визначено релевантну станцію, станцію, яка визначена як релевантна, час, коли цю пару було створено, застосовуваність цих даних, тобто чи ця пара є актуальною для створення моделі прогнозування;

– trafficapp_traffic. що містить дані здійснених вимірювань автомобільного трафіку за станціями протягом години, включаючи номер вимірювання, станцію, за якою таке вимірювання здійснено, час, коли здійснено вимірювання, кількість автомобілів, які зафіксовані за станцією спостереження, стан спостереження, що вказує на те, чи ці дані актуальні, чи вони вже були змінені (необхідно для позначення помилково внесених позицій), позначка про прогнозування, що вказує на те, чи це значення було виміряно, чи воно є результатом використання моделі прогнозування.

Під час проведення експериментів, описаних нижче, було використано дані [9], які є результатами вимірювання автомобільного трафіку на 59 станціях міста Мадрид [10] у період з 1.01.2019 до 30.09.2022, та відповідні супутні дані. У даному випадку дані про спостереження представлені окремими CSV-файлами.

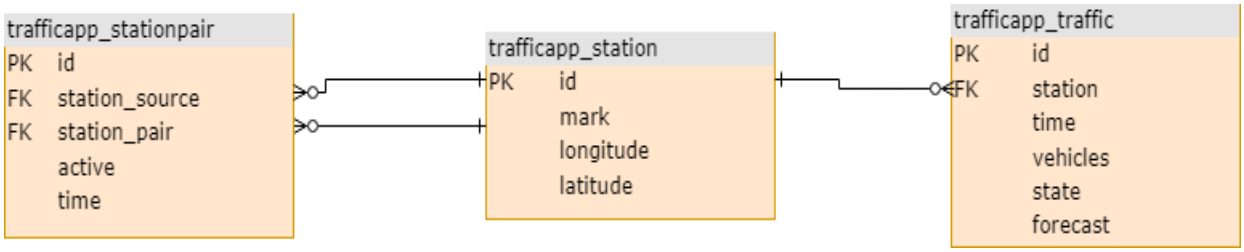


**Рис. 1. Схема бази даних**

Тож враховуючи наявні сховища, можна визначити, що для роботи з даними може використовуватися декілька варіантів: зберігати всі дані в базі даних, зберігати всі дані в файлах відповідного типу, які дозволять зберігати структуру даних, що обрана, зберігати дані для моніторингу в базі даних, а дані для роботи з моделями, представлені часовими рядами, зберігати у файлах, що передбачають структуризацію, зберігати дані для моніторингу в базі даних, а дані для роботи з моделями у зручному для користувача представленні з подальшим формуванням часових рядів кожен раз, коли дані використовуються для створення моделей.

Враховуючи описані варіанти, сценарії використання, можна розглянути наступні варіанти структурування даних у програмі: за допомогою бази даних, на основі структури даних DataFrame з пакета Pandas, на основі багатовимірного масиву з пакета NumPy з визначенням вимірювань за кожною станцією окремо та подальшим об'єднанням цих вимірювань у вигляді словника, переліку або єдиного багатовимірного масиву NumPy (для останнього варіанта базової структуризації). При цьому початково дані можуть зберігатися у вигляді різних форматів або поступати з різних джерел. Це може бути або прикладний програмний інтерфейс для збору даних з іншого сервера, безпосередньо датчиків (що менш ймовірно у випадку окресленої концепції програми), або відкриті дані, які зазвичай представлені відповідними файлами, що знаходяться на певному сервері. У даному випадку характерним є як раз другий підхід. Тоді програмне забезпечення має створюватися таким чином, щоб періодично прочитувати дані з сервера, вносити їх у певне сховище (для моніторингу надто витратним є для кожного звернення запитувати і обробляти заново дані для кожного запиту кожного користувача), а далі тільки з певним кроком оновлювати дані, зчитуючи нові та додаючи їх до сховища.

У подальшій частині роботи проведено ряд експериментів, які направлені на встановлення більш ефективної організації структур даних та алгоритмів для роботи з ними, а також сховищ даних.

Спочатку було порівняно час читання даних у випадку необхідності відслідковування. Тоді з бази даних зчитуються всі записи за заданою станцією, не обробляючи їх далі або створюючи з них структуру DataFrame. Тобто реалізовувалось фільтрування даних за номером станції, а потім виконувалось сортування за часом спостереження. Ці дії виконувались для кожної з 59 станцій 20 разів, а в таблиці 1 представлені середні значення. За аналогічною процедурою виконувались і наступні сценарії в подальших експериментах.

Для відслідковування зчитувався трафік разом з датою і часом, а для формування ряду – тільки трафік. При роботі з файлами дані вже є підготовленими у вигляді відповідно словника, переліку або масиву пакета NumPy, що містить одразу дані всіх станцій. Тоді робота полягає тільки у читанні готового файлу та виділенні потрібного ключа або індексу, що відповідають заданій станції, тобто фактично у копіюванні даних за зазначеним індексом.

Таблиця 1

**Середній час читання даних автомобільного трафіку за станцією**

| Сценарій | Час виконання, с | | | |
|---|---|---|---|---|
| | База даних | Файл зі словником | Файл з переліком | Файл з масивом |
| Читання даних для відслідковування | 0,00019 | 0,025498 | 0,025398 | 0,027892 |
| Читання даних з формуванням структури для відслідковування | 0,238689 | | | |
| Читання даних для подальшого формування ряду | 0,000176 | 0,004875 | 0,004915 | 0,005585 |
| Читання даних з формуванням структури для подальшого формування ряду | 0,043968 | | | |

Результати в таблиці 1 демонструють, що робота, пов'язана з виділенням даних, що в подальшому можуть використовуватися для відслідковування або для подальшого формування часового ряду, виконується значно швидше на основі роботи з базою даних. В залежності від структури, яка зберігається в файлі, перевага бази даних у швидкості виконання складає від 134 до майже 147 разів у випадку відслідковування, а у випадку формування ряду перевага складає від 27 до майже 32 разів. Така перевага є значною, однак варто зазначити, що сформована вона головним чином через необхідність читання файлу, що є доволі витратною операцією.

Отримані результати таким чином вказують на те, що у випадку, коли дані у вигляді словника, переліку або масиву знаходиться в файлі, витягання даних з бази даних є пріоритетним, якщо дані одразу передаються на сторінку для відображення користувачу. Якщо дані після цього обробляються в програмі далі, тобто необхідно сформувати структуру даних, щоб потім вже працювати з нею, то пріоритетним варіантом стає вже робота з файлами. Не зважаючи на необхідність читання даних, перетворення даних на масив NumPy з переліку, який отримується після читання з бази даних, є достатньо витратним, тому в підсумку робота з базою даних у такому випадку вимагає у 7,9-9,4 разів більше часу ніж робота з файлами, де такі структури вже підготовлені: зокрема у 7,9 разів більше за таку саму структуру (масив) за формування ряду та у 8,6 разів більше у випадку відслідковування.

При цьому слід враховувати, що фактично велику кількість операцій обробки даних можна достатньо швидко виконати на основі вбудованих засобів Django, а тому структурування даних найчастіше тоді є надмірним, тобто виділені дані одразу передаються в шаблон вебсторінки для подальшого відображення її користувачу. Однак, оскільки щонайменше при відслідковуванні автомобільного трафіку читання даних з файлу у вигляді певної структури недостатньо, тому що їх потрібно далі певним чином опрацьовувати для виконання потрібного відображення цих даних, то окремо було розглянуто сценарій фільтрації даних. Через те, що таке фільтрування виконується вже для виведення у вигляді сторінки, реалізацію виконано через фільтрування даних з бази даних або через фільтрування отриманих з файлу даних у структурі.

Дані у файлі розглядались у двох варіантах. У першому варіанті дані в файлі представляють часовий ряд, тобто кожен елемент для кожної станції складається з 30 значень з додаванням часової мітки, з якої розпочато фіксування цього часового ряду. Кожне значення відповідає кількості автомобілів за відповідною станцією. 30 значень представляють 30 послідовних годин. Оскільки навчання моделей за обраним методом прогнозування автомобільного трафіку виконується на основі 24 послідовних значень на вхід з прогнозуванням наступних 6 значень, тобто значень за наступні 6 годин, то часовий ряд у розроблюваній програмі складається саме з 30 годин. У другому варіанті дані в файлі представляють тільки послідовність значень за вказаною станцією, тобто значення за 1 годину, та часову мітку, коли це значення було зафіксовано. У всіх випадках у даному дослідженні час зафіксований як тип Timestamp з вказуванням року, місяця, числа, години та хвилин.

Отримані результати представлені в таблиці 2.

Таблиця 2

**Середній час відбору даних автомобільного трафіку за станцією**

| Сценарій | Час виконання, с | |
|---|---|---|
| | За часом | За кількістю |
| База даних | 0,000257 | 0,000222 |
| Словник з елементом типу DataFrame, що представляє часовий ряд | 0,000163 | 0,000146 |
| Словник з елементом типу NumPy, що представляє часовий ряд | 0,00225 | 0,00016 |
| Словник з елементом типу DataFrame | 0,000031 | 0,00004 |
| Словник з елементом типу NumPy | 0,002108 | 0,00014 |
| Перелік з елементом типу DataFrame, що представляє часовий ряд | 0,000112 | 0,000161 |
| Перелік з елементом типу NumPy, що представляє часовий ряд | 0,002126 | 0,00031 |
| Перелік з елементом типу DataFrame | 0,000032 | 0,000042 |
| Перелік з елементом типу NumPy | 0,002089 | 0,000163 |
| Масив | 0,002054 | 0,000098 |
| Масив, що представляє часовий ряд | 0.002252 | 0,000245 |

Відбір даних реалізовано за двома окремими критеріями. У першому випадку відбір здійснювався за заданим проміжком часу спостереження (позначено в таблиці 2 «за часом»). Фактично відбір здійснювався за заданою станцією за заданим проміжком у місяць. У другому випадку відбір здійснювався за кількістю автомобілів. Тоді був заданий інтервал з кількістю автомобілів від 10 до 500.

Порівняння включало використання бази даних або словника, переліку та масиву, які були попередньо завантажені з файлу в пам'ять. Словник та перелік для кожного ключа або відповідно індексу мають в якості базового елемента структуру DataFrame або NumPy. У випадку масиву вся структура представлена багатовимірним масивом NumPy.

Результати в таблиці 2 вказують на те, що найчастіше відбір даних за датою реалізується довше ніж за кількістю автомобілів. У такому випадку відбору порівняння відбувається за всіма складовими дати, на відміну від відбору за кількістю автомобілів, коли порівняння здійснюється між цілими числами. Також помітно, що використання в якості базового елемента масиву NumPy для таких дій є більш витратним, тобто відбір на такому масиві реалізується довше порівняно зі структурою DataFrame. Перевага при цьому в залежності від загальної структури, критерія відбору та використання часового ряду або даних за годину складала від 1,1 до 68 разів. Найбільший розрив зафіксований у випадку відбору за датою за словником з даними за годину.

Якщо порівнювати результати, отримані на основі переліку та на основі словника, то в цілому вони є достатньо близькими між собою, що загалом підтверджують і попередні результати читання з файлу з вибором всіх даних за станцією. Це вказує на ефективність використаного гешування в основі побудови словника, що є логічним, адже кількість ключів є відносно невеликою, а фактично структура ключів подібна до індексів, адже кожна зі станцій представлена кодом, що включає номер, який фактично і є ідентифікатором.

Використання даних, що представляють часовий ряд, призводить до збільшення часу, необхідного для відбору на таких даних, порівняно з використанням даних, що відповідають 1 годині спостереження. Збільшення складає від 1,02 до 5,26 разів. При цьому слід розуміти, що це також і додаткові витрати пам'яті на збереження такої структури. Тож у випадку передбачення виконання відбору даних як однієї з часто використовуваних функцій і вибору на користь збереження даних у вигляді структури в оперативній пам'яті доцільно створити окрему структуру, в яку виділити дані за 1 годину, замість використання загального часового ряду.

Окрім представлених спостережень варто також визначити, що база даних не дозволяє за такого сценарію використання, як описано, мати постійну перевагу в часі виконання. Це можна пояснити тим, що в базі даних зберігаються дані одразу за всіма станціями, тому пошук відбувається на більшому просторі, ніж у підготовленій структурі. Якщо структура вже наявна в пам'яті, то з неї потрібно тільки вибрати відповідний елемент, а тоді вже на меншому просторі пошуку виконувати потрібний відбір. Оскільки доступ до елемента в переліку відбувається за постійний час (у термінах часової складності алгоритму), а у випадку словника це також постійний час (або лінійний у найгіршому випадку), то логічним є те, що база даних дозволяє відібрати дані за довший час ніж перелік або словник, елементи якого містять час спостереження і відповідну кількість автомобілів. Тому у випадку, коли можливим у програмі є зберігання в оперативній пам'яті відповідного набору даних, використання переліку або словника з елементами типу DataFrame має перевагу. Проте якщо важливим є застосування додаткових можливостей системи керування базами даних, обсяг даних збільшується (що загалом є неминучим у подальшому), кількість одночасних звернень від різних користувачів збільшується, то використання системи керування базами даних все ж є більш доцільним. Якщо ці критерії не є домінуючими (наприклад, спостереження відбуваються за результатами побудови моделі або чітко відомі варіанти відображення даних користувачу і вони можуть бути опрацьовані для того, щоб потім відправляти за запитом готові результати), то використання переліків або словників описаним чином можна вважати більш доцільним.

Окремо було також перевірено відповідні випадки для роботи з базою даних, словниками та переліками за створення індексів. Індекси створювались за часом для DataFrame у словнику або переліку та за станцією і часом у випадку бази даних. Отримані результати продемонстровані в таблиці 3.

Таблиця 3

**Середній час відбору даних автомобільного трафіку за станцією з використанням індексування за датою**

| Сценарій | Час виконання, с | |
|---|---|---|
| | За часом | За кількістю |
| База даних | 0,000254 | 0,000219 |
| Словник, що представляє часовий ряд | 0,000083 | 0,000128 |
| Словник | 0,000022 | 0,00003 |
| Перелік, що представляє часовий ряд | 0,000059 | 0,000127 |
| Перелік | 0,000024 | 0,00003 |

Створення індексу дозволило пришвидшити процес відбору, зокрема для словника від 1,14 до 1,96 разів, а для переліку – від 1,27 до 1,9 разів, тож його варто вважати доцільним. У випадку використання

переліку або словника у вигляді часу та 1 години покращення визначено в 1,33-1,41 рази.

Оскільки метод, який обрано для прогнозування автомобільного трафіку, передбачає визначення для кожної моделі, що відповідає конкретній станції, релевантних станцій, підготовка даних для роботи методу відбору ознак є достатньо типовою операцією, якщо процес визначення релевантних станцій передбачає повторне визначення. Тож доцільним є визначення середнього часу підготовки даних для цього.

Підготовка даних полягає в тому, щоб відділити дві структури, кожна з яких представляє собою масив NumPy (адже саме така структура має бути подана в результаті на вхід). Перший масив містить часовий ряд за попередню годину для всіх станцій окрім тієї, для якої будується модель. Другий масив містить часовий ряд за поточну годину для однієї станції, для якої будується модель. Тобто різниця в даних при цьому складає годину.

Важливим моментом також є те, що читання файлу відбувалось поза вимірюванням часу, тобто в такому випадку весь часовий ряд повинен містить у оперативній пам'яті. У випадку використання в основі елементу структури даних типу DataFrame такий файл займає 240 Мб. Відповідно зі збільшенням кількості станцій, зі збільшенням періоду спостереження цей обсяг буде збільшуватись і вимагати більше оперативної пам'яті в підсумку або більше часу для того, щоб прочитати дані з файлу.

Отримані в підсумку результати обчислення часу представлені в таблиці 4.

Таблиця 4

**Середній час підготовки даних для вибору релевантних станцій**

| **Сценарій** | **Час виконання, с** | |
|---|---|---|
| | **Елемент типу DataFrame** | **Елемент типу масиву NumPy** |
| База даних | 4,731292 | |
| База даних з індексуванням | 3,285113 | |
| Словник | 0,001447 | 0,004562 |
| Перелік | 0,016012 | 0,139129 |
| Масив | 0,194369 | |

Як помітно з результатів у таблиці 4, час підготовки даних для вибору релевантних станцій з бази даних є значно більшим ніж за використання словника, переліку або масиву. При аналізі потрібно враховувати специфіку цих сценаріїв у контексті вибору релевантних станцій: по-перше, ці структури даних вже є прочитаними з файлів, відповідно виконання даного завдання фактично вимагає копіювання даних з готової структури, по-друге, база даних вимагає виконання цілого ряду дій, які включають вибір даних за кожною станцією, що аналізується, відкидання частини цих даних про організації зсуву та перетворення цих даних на масив. У випадку, якщо ці структури даних не знаходяться в пам'яті, а їх потрібно прочитати з файлу, такі дії вже будуть вимагати значно більшого часу. Для того щоб мати можливість врахувати і такий випадок, окремо було проаналізовано і такі сценарії. Результати представлені в таблиці 5.

Таблиця 5

**Середній час виконання сценаріїв, пов'язаних з роботою з часовими рядами**

| **Сценарій** | **Час виконання, с** | |
|---|---|---|
| | **Без дати** | **З датою** |
| Формування часового ряду у словнику на основі даних XLSX-файлів | 2,340111 | |
| Формування часового ряду у переліку на основі даних XLSX-файлів | 2,394887 | |
| Формування часового ряду з бази даних | 0,859649 | |
| Формування часового ряду з бази даних з індексацією | 0,794949 | |
| Нормування даних з бази даних | 0,774783 | |
| Нормування даних з бази даних з індексацією | 0,71698 | |
| Читання з файлу зі словником | 0,250242 | 10,249701 |
| Читання з файлу з переліком | 0,253388 | 10,381432 |
| Читання з файлу з масивом | 0,296898 | 9,056661 |
| Запис у файл зі словником | 0,217074 | 8,051293 |
| Запис у файл з переліком | 0,187853 | 8,166063 |
| Запис у файл з масивом | 2,137535 | 7,576285 |

У таблиці 5 порівняно час роботи з даними, які представляють часовий ряд, у випадку з'єднання з джерелом даних, яким є база даних або файл. Якщо джерелом даних є XLSX-файл, тобто дані накопичуються оператором або програмно зовнішнім чином у таких файлах, то процес формування часового ряду є витратним. За такого використання більш доцільним є застосування бази даних. При цьому потрібно зазначити, що у всіх випадках формування часового ряду виконувалось також і нормування. Сам процес читання даних з файлу може бути достатньо витратним, особливо у випадках наявності в структурі даних різного типу, зокрема часових міток. При цьому витратним є і процес запису даних у файл. Тому за сценаріїв,

що передбачають постійну взаємодію з файлами, доцільно застосовувати бази даних і обробляти отримані дані в програмі для формування відповідних структур даних під прогнозування автомобільного трафіку.

**Висновки**

Представлені результати дослідження дозволяють порівняти різні варіанти використання структур даних у розрізі застосування відповідних алгоритмів стосовно швидкості виконання цих сценаріїв при створенні системи, яка передбачає моніторинг і прогнозування автомобільного трафіку.

Загалом можна зробити висновок, що у випадку неможливості постійного зберігання структурованих даних у оперативній пам'яті, зокрема коли передбачається робота з даними, що відповідають значному періоду часу і/або великій кількості станцій спостереження, доцільною є організація роботи з даними як стосовно підготовки до прогнозування, так і моніторингу в розрізі продуктивності на основі бази даних. У випадку, коли період часу є невеликим, кількість станцій значно обмежена і можливим є зберігання відповідних підготовлених структур даних у оперативній пам'яті, переліки і словники є достатньо продуктивними при побудові базових елементів на основі табличної структури.

**References**

1. Lu J., Wei Z., Tian D. Urban Traffic Flow Prediction Model Based on Spatio-temporal Graph Convolutional Network. Journal of Wuhan University of Technology (Transportation Science and Engineering). 2023. Volume 47, Issue 2. p. 234–238. DOI: 10.3963/j.issn.2095-3844.2023.02.007.

2. Lei Bai, Lina Yao, Can Li, Xianzhi Wang, Can Wang. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. 34th Conference on Neural Information Processing Systems (NeurIPS2020), Vancouver, Canada. Vancouver, 2020. P. 17804–17815. DOI: 10.48550/arXiv.2007.02842.

3. Chen R., Yao H. Hybrid Graph Models for Traffic Prediction. Applied Sciences. 2023. № 13. Article 8673. DOI: 10.3390/app13158673.

4. Kumar R., Panwar R., Chaurasiya V. K. Urban traffic forecasting using attention based model with GCN and GRU. Multimedia Tools and Applications. 2023. DOI: 10.1007/s11042-023-17248-y.

5. Zhizhu Wu, Mingxia Huang, Aiping Zhao and Zhixun lan. Traffic prediction based on GCN-LSTM model. Journal of Physics: Conference Series. 2021. Volume 1972. Article 012107. DOI: 10.1088/1742-6596/1972/1/012107.

6. Zhao L., Song Y., Zhang C. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction / // IEEE Transactions on Intelligent Transportation Systems. 2020. Volume 21, Issue 9. P. 3848–3858. DOI: 10.1109/TITS.2019.2935152.

7. Hu X., Liu T., Hao X. et al. Attention-based Conv-LSTM and Bi-LSTM networks for large-scale traffic speed prediction. The Journal of Supercomputing. 2022. Volume 78. P. 12686–12709. DOI: 10.1007/s11227-022-04386-7.

8. Lovkin V. M., Subbotin S. A., Oliinyk A. O. Method for Agent-Oriented Traffic Prediction under Data and Resource Constraints. Radio Electronics, Computer Science, Control. 2023. № 4. P. 99-110. DOI: https://doi.org/10.15588/1607-3274-2023-4-10.

9. Aforos de trafico en la ciudad de Madrid permanentes – Portal de datos abiertos del Ayuntamiento de Madrid. https://datos.madrid.es/sites/v/index.jsp?vgnextoid=fabbf3e1de124610VgnVCM2000001f4a900aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD.

10. En portada – Portal de datos abiertos del Ayuntamiento de Madrid. https://datos.madrid.es/portal/site/egob.