Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Відновлення конструктору руху міського громадського автотранспорту»
за освітньою програмою: **12 Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студент групи ПЗ1912 _____ /Михайло ВЄТЛУЖСЬКИХ/
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /Віктор ШИНКАРЕНКО/
(підпис) (Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /Світлана ВОЛКОВА/
(підпис) (Ім'я ПРІЗВИЩЕ)

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

Студент _____
(підпис)

Дніпро – 2023

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: «Restoration of the constructor of the traffic of the city public transport»

according to educational curriculum **12 Software engineering**

in the Speciality: **121 Software engineering**

Done by the student of the group PZ1912: ____ /Mykhailo VIETLUZHSKYKH/

(підпис)

Scientific Supervisor: ____ /Victor SCHINKARENKO/

(підпис)

Normative controller: ___ /Svitlana VOLKOVA/

(підпис)

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерні технології і системи
Кафедра: Комп'ютерні інформаційні технології
Рівень вищої освіти: бакалавр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____/Вадим ГОРЯЧКІН/

_____ 2023 р.

ЗАВДАННЯ

На кваліфікаційну роботу бакалавра
студенту Вєтлужських Михайло Вячеславовича

1. Тема роботи: «Відновлення конструктору руху міського громадського автотранспорту»
Керівник роботи: Шинкаренко Віктор Іванович
затверджені наказом № 1196 ст від 07.12.2022 року

2. Строк подання студентом роботи: 22.06.2023 року

3. Вихідні дані до роботи: _____.

4. Зміст пояснювальної записки (перелік питань до розробки): реферат, вступ, аналіз вимог та передпроєктні дослідження, проектування, розробка програми, тестування та відлагодження, висновки, список використаних джерел.

5. Перелік демонстраційного матеріалу:
   5.1. доповідь;
   5.2. презентація;
   5.3. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вступ	12.09.22 – 26.10.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	27.10.22 – 04.03.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	05.03.23 – 31.04.23	
4	Постановка задачі, технічне завдання	01.05.23 – 07.05.23	30%
5	Узгодження та затвердження ТЗ	08.05.23 – 15.05.23	
6	Розробка інструментальних засобів дослідження	16.05.23 – 21.05.23	
7	Виконання досліджень	29.05.23 – 01.06.23	60%
8	Оформлення тез доповідей	21.03.2023 – 24.03.2023	
9	Оформлення статті у фаховий журнал	02.06.23 – 06.06.23	
10	Оформлення пояснювальної записки	07.06.23 – 11.06.23	
11	Розробка демонстраційних матеріалів	12.06.23 – 18.06.23	100%
12	Подання кваліфікаційної роботи до кафедри	22.06.23	
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісі	27.06.23	

Студент _____ Михайло ВЄТЛУЖСЬКИХ

Керівник роботи _____ Віктор ШИНКАРЕНКО

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається з 180 с., 25 рис., 4 табл., 4 додатки, 26 джерел.

Об’єктом дослідження є дані руху громадського автотранспорту. Предметом дослідження є прогнозування швидкості руху громадського автотранспорту на певних ділянках маршруту.

Метою дослідження є накопичення, аналіз та підготовка даних для відновлення конструктору руху громадського автотранспорту.

В результаті дослідження накопичено дані, побудовано систему аналізу даних, відновлено конструктор та розроблено систему візуалізації даних.

Пояснювальна записка складається з наступних 8 розділів.

1. Вступ – розділ, у якому описується мета роботи, актуальність, та експлуатаційне призначення. Цей розділ складається з 1 сторінки.
2. Аналіз вимог та передпроєктні дослідження– розділ, у якому описуються та аналізуються аналоги програми, досліджується література з обраної теми та формуються вимоги до програмного забезпечення. Цей розділ складається з 7 сторінок.
3. Проектування – розділ, у якому відбувається визначення функціональних та нефункціональних вимог до програмного забезпечення, його призначення, зовнішнє проектування та внутрішнє проектування системи. Цей розділ складається з 23 сторінки.
4. Розробка програми – розділ, у якому описується процес аналізу даних та розробки складових програми. Цей розділ складається з 35 сторінок.
5. Тестування та відлагодження – розділ, в якому описується тестування програми. Цей розділ складається з 7 сторінок.
6. Висновки – розділ, у якому приведено висновки до дослідження. Цей розділ складається з 1 сторінки.

7. Список використаних джерел – розділ, який включає в себе бібліографічний список використаної літератури. Цей розділ складається з 4 сторінок.
8. Додатки – розділ, який містить робочий проект.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Найменування	Значення
API	З англ. Application Programming Interface, інтерфейс програмування додатку.
URL	Уніфікований лока́тор ресурсів або адреса ресурсу (англ. Uniform Resource Locator — єдиний вказівник на ресурс, URL)
SSE інтерфейс	SSE (від англ. Server-Sent Events — «події, що посилаються сервером») є технологією відправлення повідомлень від сервера до веб-браузера у вигляді DOM-подій.
UX & UI	Досвід користування (англ. User Experience, UX) — це те, що людина відчуває при користуванні продуктом, системою чи сервісом (послугою). Інтерфейс користувача (скорочено ІК), (англ. user interface, UI) — засіб зручної взаємодії користувача (людини) з інформаційною системою.

ВСТУП

Під час подорожування громадським автотранспортом виникає проблема непостійності інтервалів руху. Тобто, людина не може прогнозувати, скільки потрібно чекати на наступний транспорт. Найбільш актуальна ця проблема під час заторів на дорогах, які можуть виникнути в результаті ремонтних робіт, свят, тощо.

Сьогодні, для прогнозування використовується сервіс EasyWay. Головними перевагами цього сервісу є:

- відстежування руху громадського транспорту у реальному часі з точністю менше одної хвилини;
- можливість відстежування громадського електротранспорту, а саме тролейбусів та трамваїв.

Метою роботи є накопичення даних для подальшого їх дослідження за допомогою сучасних засобів і відновлення конструктору.

Експлуатаційні призначення: наступна робота допоможе подорожувачам точніше планувати свій час на поїздку громадським транспортом.

1 АНАЛІЗ ВИМОГ ТА ПЕРЕДПРОЄКТНІ ДОСЛІДЖЕННЯ

1.1 Аналіз аналогів

Google Maps надає функцію прогнозування руху громадського транспорту для деяких міст. Вона використовує дані про розклад руху транспорту, а також інформацію про поточну дорожню ситуацію, щоб оцінити час прибуття автобусів, трамваїв або метро до певної зупинки.

Moovit – це мобільний додаток, який надає прогнози руху громадського транспорту в багатьох містах по всьому світу. Він оновлюється в реальному часі та надає користувачам інформацію про прибуття транспорту, розклад руху, оповіщення про зміни у руховому графіку та іншу корисну інформацію. Також, підтримує кілька міст в Україні. Зокрема, він надає інформацію про рух громадського транспорту в Києві, Львові, Харкові, Одесі, Дніпрі та інших містах

Citymapper – це ще один популярний додаток для навігації громадським транспортом. Він надає користувачам інформацію про найкоротший шлях до пункту призначення з використанням різних видів транспорту. Додаток також прогнозує час прибуття транспорту, враховуючи поточний руховий графік і дані про затримки.

Transit – це додаток для прогнозування руху громадського транспорту, який доступний для багатьох міст по всьому світу. Він надає розклад руху автобусів, трамваїв, метро та іншого громадського транспорту, а також показує реальний час прибуття транспорту на зупинку. Додаток також надає повідомлення про зміни у руховому графіку і іншу корисну інформацію.

EasyWay є популярним додатком для прогнозування руху громадського транспорту в Україні. Він надає розклади руху автобусів, трамваїв, метро та маршруток для різних міст, включаючи Київ, Львів, Одесу, Харків та інші. EasyWay також надає можливість стежити в реальному часі за місцезнаходженням транспорту та отримувати сповіщення про зміни у руховому графіку.

Додаток Kyiv Metro спеціально призначений для прогнозування руху метро в Києві. Він надає користувачам розклад руху по станціях, інформацію про поточну дорожню ситуацію, а також можливість стежити за місцезнаходженням потягів в реальному часі.

Uklon є популярним сервісом замовлення таксі в Україні, але також надає можливість переглядати розклади громадського транспорту. У додатку Uklon можна знайти розклади автобусів, тролейбусів, трамваїв та маршруток для деяких міст України, зокрема Києва, Львова, Одеси, Харкова та Дніпра.

В результаті аналізу аналогів було встановлено, що аналоги добре виконують функції спостерігання за транспортом, але не мають можливості передбачування.

1.2 Аналіз вимог

Аналіз вимог було виконано методом інтерв'ю.

1.2.1 Інтерв'ю з замовником

Метою цього інтерв'ю є збір початкових вимог до системи конструктору руху міського громадського транспорту.

Питання: "Які вимоги до інструментарію для вирішення задачі?"

Відповідь: "Пропоную використати конструктор для вирішення поставленої задачі."

Питання: "Які вимоги до досліджуваних даних?"

Відповідь: "Добре було б знайти та використати реальні дані руху транспорту для дослідження."

Питання: "Які вимоги до географічної позиції даних?"

Відповідь: "Пріоритетом є м. Дніпро в Україні. "

В результаті цього інтерв'ю були сформульовані наступні вимоги:

- побудова конструктора для вирішення поставленої задачі;
- використання реальних даних для руху транспорту для дослідження;
- пріоритетом географічної позиції даних є м. Дніпро в Україні.

1.2.2 Інтерв'ю з підтримкою сервісу EasyWay

Метою цього інтерв'ю є розгляд можливості отримання історичних даних руху транспорту м. Дніпро з сервісу EasyWay.

Питання: "Чи є у Вашому сервісі можливість отримання історичних даних руху транспорту у м. Дніпро за декілька років?"

Відповідь: "Ні, такої можливості немає."

Питання: "Чи є можливість отримання моментальних даних?"

Відповідь: "Так, така можливість є, ми можемо надати доступ до нашого API."

В результаті цього інтерв'ю були сформульовані наступні вимоги:

- необхідність розробки програми для накопичування даних з сервісу EasyWay.

1.2.3 Інтерв'ю з замовником

Метою цього інтерв'ю є збір вимог до системи накопичення даних.

Питання: "Які вимоги до надійності системи?"

Відповідь: "Система повинна мати можливість збирання даних цілодобово у фоновому режимі тривалий час без втручання сторонніх осіб."

В результаті цього інтерв'ю було сформульовані наступні вимоги:

- система повинна мати можливість збирання даних цілодобово у фоновому режимі тривалий час без втручання сторонніх осіб.

1.3 Передпроєктні дослідження

1.3.1 Дослідження можливих джерел даних

В результаті дослідження можливих джерел даних було виявлено наступні джерела:

- EasyWay;
- Moovit.

Було проаналізовано кожен з варіантів і було виявлено, що Moovit не має підтримки м. Дніпро. Також, EasyWay має більш прозорий інтерфейс для відстежування місць знаходження транспортних засобів.

В результаті аналізу можливих джерел даних було вибрано сервіс EasyWay.

1.3.2 Основи конструктивно-продукційного моделювання

В основу конструктивно-продукційного моделювання покладено поняття узагальненої конструктивно-продукційної структури [4]. Як результат та продовження зазначених досліджень, у роботах [5], [6], [7], [8] запропоновано розглядати засіб конструювання – узагальнений конструктор (ОК)

$$C = \langle M, \Sigma, \Lambda \rangle, \qquad (1.1)$$

де M – неоднорідний розширюваний носій, Σ – сигнатура відносин та відповідних операцій: зв'язування, підстановки, виведення над атрибутами, Λ – безліч тверджень інформаційного забезпечення конструювання (ІОК), яке включає: онтологію, мету, правила, обмеження, умови початку та завершення конструювання.

В M можна виділити підмножини: T – терміналів, N – нетерміналів (допоміжних, абстрактних елементів), з властивостями $T \cap N = \emptyset$, $\varepsilon \in T$, $\varepsilon \notin N$, де ε– порожній елемент.

Особливостями конструктивно-продукційного моделювання із застосуванням є [5], [6], [7], [8] : атрибутивність елементів і операцій, носій, що розширюється, модель виконавця у вигляді його базових алгоритмів, зв'язок операцій з алгоритмами їх виконання..

Онтологію узагальненого конструктора у неформальному вигляді викладено у [4], нижче наведено її частину необхідну для подальшого викладу.

Сигнатура Σ складається з багатьох операцій: Ξ – зв'язування, Θ – підстановки та виведення, Φ – операцій над атрибутами. Сигнатура містить також відносини підстановки «→». Отже, формально сигнатура є $\Sigma =$

$\langle \Xi, \Theta, \Phi, \{\rightarrow\}\rangle$, з властивостями: $\Xi \cap \Theta = \emptyset; \Xi \cap \Phi = \emptyset;$ $\Theta \cap \Phi = \emptyset, \varepsilon \in \Phi$. Сигнатура складається з імен операцій $\{\bigotimes_j\}$, що володіють набором атрибутів w_i, представляється як $w \bigotimes \in \Sigma$.

Операції зв'язування елементів конструктора з'єднують окремі елементи конструкції або їх частини (проміжні форми).

У класичних формальних граматиках використовується одна бінарна операція зв'язування (конкатенації) над елементами термінального та нетермінального алфавітів, проте для спеціалізованих граматик можуть використовуватись різноманітні операції зв'язування: за умовою, багатомісні, графічні елементи та ін.

Під формою ${}_{w_l}l$ з набором атрибутів w_l розуміють:

- ${}_{w_l}l = {}_{w_0}\bigotimes ({}_{w_1}m_1, {}_{w_2}m_2, \dots, {}_{w_k}m_k)$ для $\forall {}_{w_i}m_i \in M$;
- ${}_{w_l}l = {}_{w_j}m_j$, якщо $l = {}_{w_0}\bigotimes (\varepsilon, \dots, \varepsilon, {}_{w_j}m_j, \varepsilon, \dots, \varepsilon)$;
- ${}_{w_l}l = {}_{w_0}\bigotimes ({}_{w_1}l_1, {}_{w_2}l_2, \dots, {}_{w_k}l_k)$, якщо ${}_{w_1}l_1, {}_{w_2}l_2, \dots, {}_{w_k}l_k$ – форми.

Таким чином, операція зв'язування застосовується як до елементів носія, так і форм, сконструйованим з її допомогою на основі елементів носія.

Ставлення постановки – двомісне ставлення до атрибутами ${}_{w_i}l_i \rightarrow {}_{w_j}l_j$.

Нехай $s = \langle {}_{w_1}l_1 \rightarrow {}_{w_2}l_2, {}_{w_3}l_3 \rightarrow {}_{w_4}l_4, \dots, {}_{w_m}l_m \rightarrow {}_{w_{m+1}}l_{m+1}\rangle$ – послідовність відносин підстановки чи $s = \varepsilon$, і $g = \langle \oplus_1 (w_{1,1}, w_{2,1}, \dots, w_{k_1,1}), \oplus_2 (w_{1,2}, w_{2,2}, \dots, w_{k_2,2}), \dots, \bigoplus_n (w_{1,n}, w_{2,n}, \dots, w_{k_n,n})\rangle$ – послідовність операцій над атрибутами. Назвемо правилом продукції $p: \langle s, g\rangle$. Тут $\bigoplus$ довільна операція над атрибутами ($\bigoplus \in \Phi$).

Безліч правил продукцій будемо позначати $\Psi = \{\psi_i: \langle s_i, g_i\rangle\}$.

Нехай задана форма ${}_{w_l}l = \bigotimes ({}_{w_1}l_1, {}_{w_2}l_2, \dots, {}_{w_h}l_h, \dots, {}_{w_k}l_k)$ і ставлення підстановки ${}_{w_h}l_h \rightarrow {}_{w_q}l_q$ таке, що ${}_{w_h}l_h \prec {}_{w_l}l$ (ставлення $\prec$ – містить), тоді результатом ${}_{w_l^*}l^*$ тримісної операції підстановки $\Rightarrow ({}_{w_h}l_h, {}_{w_q}l_q, {}_{w_l}l)$ буде форма ${}_{w_l^*}l^* = \bigotimes ({}_{w_1}l_1, {}_{w_2}l_2, \dots, {}_{w_q}l_q, \dots, {}_{w_k}l_k)$, де $\Rightarrow \in \Theta$.

Двомісна операція часткового виведення ${}_{w_l^*}l^* =_{v_p} | \Rightarrow (\Psi, {}_{w_l}l)$ ($| \Rightarrow \in \Theta$) полягає в:

- виборі одного з доступних правил підстановки $p_r: \langle s_r, g_r \rangle$ з відносинами підстановки s_r;
- виконання на його основі операцій підстановки;
- виконання операцій над атрибутами g_rу відповідній послідовності.

Операція повного виведення або просто виведення ($|| \Rightarrow \in \Theta$) полягає в покроковому перетворенні форм, починаючи з початкового нетерміналу і закінчуючи конструкцією, що задовольняє умову закінчення висновку, що передбачає циклічне виконання операцій часткового виведення. Операція двомісна ${}_{\Delta, w_l^*}l^* = || \Rightarrow (\Psi, {}_{w_l}l)$, де ${}_{w_l}l \in U$

Результуючі конструкції операцій повного виведення належать $\Omega(C_L)$.

Для формування конструкцій виконується ряд уточнюючих перетворень:

- спеціалізація визначає предметну область: семантичну природу носія, кінцеву множину операцій та їх семантику, атрибутику операцій, порядок їх виконання та обмеження на правила підстановки $C_S \mapsto {}_SC$;
- інтерпретація полягає у зв'язуванні операцій сигнатури C_A з алгоритмами виконання деякої алгоритмічної структури, що пов'язує інформаційну модель засобів формування конструкцій та модель виконавця ${}_SC, C_{A_I} \mapsto < {}_{S,I}C, C_A >$, утворюючи конструктивну систему;
- конкретизація конструктора полягає у розширенні аксіоматики безліччю правил продукцій, завданні конкретних множин нетермінальних та термінальних символів з їх атрибутами та, за необхідності, значень атрибутів ${}_{S,I}C_K \mapsto {}_{S,I,K}C$;

- реалізація конструктора полягає у формуванні множини конструкції з елементів носія конструктора шляхом виконання алгоритмів, пов'язаних з операціями сигнатури ${}_{S,I,K}C_R \mapsto \Omega$.

У роботах [5], [6], [7], [8] уточнюючі перетворення виконуються у такій послідовності

$$C_S \mapsto {}_SC \mapsto {}_SC, C_{A_I} \mapsto < {}_{S,I}C, C_A >_K \mapsto < {}_{S,I,K}C, C_A >_R \mapsto \Omega\,. \qquad (1.2)$$

2 ПРОЄКТУВАННЯ

До цілей проекту належать наступні пункти.

- Забезпечення даними для дослідження.
- Розробка інструменту для часткової автоматизації аналізу та дослідження даних.
- Розробка інструменту для візуалізації результатів дослідження.
- Підготовка даних та інструментів для подальшого успішного відновлення конструктору руху міського громадського автотранспорту.

До функціональних вимог належать наступні пункти.

- Накопичування даних: система повинна накопичити реальні дані для дослідження.
- Вибір даних: система повинна мати можливість вибору даних зі сховища.
- Гнучке дослідження даних: система повинна мати можливість зручного дослідження даних за допомогою вбудованих та самостійно реалізованих методів.
- Візуалізація даних: система має мати можливість візуалізації результатів дослідження.

До нефункціональних вимог належать наступні пункти.

- Надійність: система має працювати без втручання сторонніх осіб цілодобово протягом тривалого часу для того, щоб не було втрачання даних.
- Розширюваність: система має мати високий рівень розширюваності.
- Простота та зручність використання.

До користувачів системи належать люди, які роблять дослідження громадського руху за допомогою даних EasyWay, або інших постачальників даних.

2.1 ЗОВНІШНЄ ПРОЕКТУВАННЯ

2.1.1 Інтерфейси системи

2.1.1.1 Інтерфейс взаємодії з API сервісу EasyWay

Для отримання від сервісу EasyWay було використано протокол Server-sent events.

Система використовує інтерфейс SSE для взаємодії зі стороннім сервісом EasyWay. EasyWay надсилає моментальних даних місць знаходження громадського транспорту.

2.1.1.1.1 Специфікації інтерфейсу

URL: URL для підключення до SSE інтерфейсу — "https://gps.easyway.info".

Параметри: для отримання даних потрібного міста потрібно використати ідентифікатор цього міста, який можна знайти у запросі на сторінці https://www.eway.in.ua.

Формат події: Кожна подія, що надходить від EasyWay, має внутрішній формат. Для того, щоб привести його до формату JSON об’єкту, було використано код зі сторінки https://www.eway.in.ua. Отриманий JSON-об'єкт містить наступні поля.

- Об’єкт позицій “positions”. Ключем в якому виступає ідентифікатор маршруту, а значенням виступає список позицій кожної одиниці транспорту. Кожна позиція одиниці транспорту має наступні поля.
    - Статус “s”. Може мати наступні значення.
        - “OK” — машина рухається за маршрутом.
        - “STOP” — машина знаходиться на зупинці.
        - “END_STOP” — машина знаходиться на кінцевій зупинці.
    - Напрямок руху “d”. Може мати наступні значення.

        - “0” — прямий маршрут.
        - “1” — зворотній маршрут.
        - “-1” — напрямок визначити неможливо.
    - “vi” — ідентифікаційний номер транспортного засобу.
    - “vn” — державний реєстраційний номер транспортного засобу.
    - “ang” — кутовий напрямок руху.
    - “idx” — ідентифікатор найближчого опорного пункту на маршруті.
    - “time” — час останньої синхронізації запису із GPS пристроєм транспорту.
    - “spd” — швидкість транспорту.
    - “pos” — GPS координати транспортного засобу.
- Час конструювання повідомлення “timestamp”. Час знаходиться у форматі цілого числа, яке зберігає час у форматі unix секунд.
- Об’єкт маршрутів “routes”. Ключем в якому виступає ідентифікатор маршруту, а значенням виступає об’єкт інформації про маршрут. У об’єкті інформації про маршрут знаходяться наступні поля.
    - Ім’я маршруту “name”. Може бути номером трамваю, тролейбусу або автобусу.
    - Тип маршруту “tr”. Може мати наступні значення.
        - “tram” — трамвай.
        - “trol” — тролейбус.
        - “marshrutka” — маршрутка.
        - “bus” — автобус.

Приклад події можна побачити нижче.


```
[
 {
  "positions": {
   "119": [
    {
```

```
  "s": "OK",
  "hc": 0,
  "wf": 0,
  "ac": 0,
  "d": 0,
  "vi": 50273,
  "vn": "1427",
  "ang": 72,
  "t": 1135,
  "idx": 185,
  "idx_app_r": 186,
  "idx_app_nb": 186,
  "time": 1640792779,
  "spd": 15,
  "spd_raw": 0,
  "spd_ai": 9,
  "pos": "48.497631:34.949228"
},
{
  "s": "OK",
  "hc": 0,
  "wf": 0,
  "ac": 0,
  "d": 1,
  "vi": 50232,
  "vn": "1381",
  "ang": 127,
  "t": 1133,
  "idx": 25,
  "idx_app_r": 474,
  "idx_app_nb": 26,
  "time": 1640792776,
  "spd": 15,
  "spd_raw": 0,
  "spd_ai": 18,
  "pos": "48.404515:35.065659"
},
{
  "s": "OK",
  "hc": 0,
  "wf": 0,
  "ac": 0,
  "d": 0,
  "vi": 51852,
  "vn": "1292",
  "ang": 105,
  "t": 1135,
  "idx": 139,
  "idx_app_r": 140,
  "idx_app_nb": 140,
  "time": 1640792785,
  "spd": 15,
  "spd_raw": 28,
  "spd_ai": 19,
  "pos": "48.467783:34.947452"
},
{
```

```
  "s": "OK",
  "hc": 0,
  "wf": 0,
  "ac": 0,
  "d": 0,
  "vi": 50304,
  "vn": "1289",
  "ang": 59,
  "t": 1135,
  "idx": 381,
  "idx_app_r": 382,
  "idx_app_nb": 382,
  "time": 1640792778,
  "spd": 15,
  "spd_raw": 0,
  "spd_ai": 16,
  "pos": "48.465224:35.005465"
},
{
  "s": "OK",
  "hc": 0,
  "wf": 0,
  "ac": 0,
  "d": 1,
  "vi": 50168,
  "vn": "1248",
  "ang": 173,
  "t": 1133,
  "idx": 253,
  "idx_app_r": 702,
  "idx_app_nb": 254,
  "time": 1640792788,
  "spd": 15,
  "spd_raw": 19,
  "spd_ai": 8,
  "pos": "48.477529:34.969544"
},
{
  "s": "OK",
  "hc": 0,
  "wf": 0,
  "ac": 0,
  "d": 1,
  "vi": 50260,
  "vn": "3061",
  "ang": 241,
  "t": 1133,
  "idx": 102,
  "idx_app_r": 551,
  "idx_app_nb": 103,
  "time": 1640792782,
  "spd": 15,
  "spd_raw": 0,
  "spd_ai": 16,
  "pos": "48.466085:35.013927"
},
{
```

```
    "s": "OK",
    "hc": 0,
    "wf": 0,
    "ac": 0,
    "d": 1,
    "vi": 50176,
    "vn": "1442",
    "ang": 285,
    "t": 1133,
    "idx": 372,
    "idx_app_r": 821,
    "idx_app_nb": 373,
    "time": 1640792774,
    "spd": 15,
    "spd_raw": 27,
    "spd_ai": 18,
    "pos": "48.477355:34.974733"
  },
  {
    "s": "OK",
    "hc": 0,
    "wf": 0,
    "ac": 0,
    "d": 1,
    "vi": 50148,
    "vn": "1337",
    "ang": 241,
    "t": 1133,
    "idx": 52,
    "idx_app_r": 501,
    "idx_app_nb": 53,
    "time": 1640792784,
    "spd": 15,
    "spd_raw": 0,
    "spd_ai": 11,
    "pos": "48.465046:35.075788"
  },
  {
    "s": "END_STOP",
    "hc": 0,
    "wf": 0,
    "ac": 0,
    "d": -2,
    "vi": 51571,
    "vn": "AE 0749 AB ",
    "ang": 323,
    "t": 3098,
    "idx": 2104,
    "idx_app_r": 2105,
    "idx_app_nb": 2105,
    "time": 1640792758,
    "spd": 25,
    "spd_raw": 0,
    "spd_ai": 10,
    "pos": "47.970992:33.422190"
  }
],
```

```
    "1033": [
     {
      "s": "OK",
      "hc": 1,
      "wf": 0,
      "ac": 0,
      "d": 1,
      "vi": 50311,
      "vn": "2587",
      "ang": 107,
      "t": 3111,
      "idx": 138,
      "idx_app_r": 465,
      "idx_app_nb": 139,
      "time": 1640792778,
      "spd": 16,
      "spd_raw": 21,
      "spd_ai": 14,
      "pos": "48.452806:35.033635"
     }
    ]
   },
   "timestamp": "2021-12-29T15:46:52.000Z",
   "routes": {
    "119": {
     "name": "5",
     "tr": "tram"
    },
    "137": {
     "name": "7",
     "tr": "trol"
    },
    "147": {
     "name": "121",
     "tr": "bus"
    },
    "161": {
     "name": "22",
     "tr": "marshrutka"
    }
   }
  }
]
```


Лістинг 2.1 – Приклад даних від постачальника

Обробка подій: коли система отримує подію, вона записує отримані дані до бази даних.

Обробка відключення: у разі втрати з'єднання з SSE інтерфейсом, система автоматично спробує відновити з'єднання.

Частота подій: дані приходять кожні 5-10 секунд.

2.1.1.2 Інтерфейс взаємодії з базою даних MongoDB

Створення документів: Ви можете створювати нові документи, використовуючи метод insert(). Документи представляють собою об'єкти JSON і можуть містити різні типи даних, включаючи рядки, числа, масиви, вкладені об'єкти і бінарні дані.

Читання документів: Ви можете запитувати документи за допомогою методу find(), вказавши критерії запиту у вигляді об'єкта JSON. Можливе також використання операторів запиту для більш складних запитів.

Оновлення документів: Для оновлення документів використовується метод update(). Ви можете вказати критерії запиту для вибору документів, які потрібно оновити, і задати нові значення для конкретних полів цих документів.

Видалення документів: Видалення документів відбувається за допомогою методу remove(). Ви можете вказати критерії запиту для вибору документів, які потрібно видалити.

Індексація: MongoDB підтримує індексацію для покращення швидкості запитів. Ви можете створити індекси на одному або декількох полях документа.

Агрегація: MongoDB має потужний механізм агрегації, який дозволяє об'єднувати документи та обробляти результати в багатофазних операціях.

2.1.1.3 Інтерфейси програмування застосунків (API)

HTTP (Hypertext Transfer Protocol) API системи дозволяє взаємодіяти з її різними компонентами та службами. Це RESTful API, що означає, що воно використовує HTTP методи GET, POST, PUT, DELETE та інші для здійснення CRUD (Створення, Читання, Оновлення, Видалення) операцій.

Ось приклади декількох основних HTTP запитів та відповідей:

- GET /historical-data/{id}: Цей запит повертає деталі користувача з вказаним ідентифікатором. Відповідь може бути JSON об'єктом, що містить інформацію про користувача, включаючи його ім'я, електронну пошту та інші деталі.

- POST /historical-data: Цей запит створює нового користувача. Він приймає JSON об'єкт у тілі запиту, що містить деталі нового користувача. Відповідь може бути JSON об'єктом, що містить ідентифікатор новоствореного користувача.
- PUT /historical-data/{id}: Цей запит оновлює деталі користувача з вказаним ідентифікатором. Він приймає JSON об'єкт у тілі запиту, що містить нові деталі користувача. Відповідь може бути JSON об'єктом, що підтверджує успішне оновлення.
- DELETE /historical-data/{id}: Цей запит видаляє користувача з вказаним ідентифікатором. Відповідь може бути JSON об'єктом, що підтверджує успішне видалення.

Усі запити та відповіді мають заголовки, які вказують тип контенту (зазвичай application/json), статус відповіді (наприклад, 200 для успішної відповіді, 404 для "не знайдено" та 500 для внутрішньої помилки сервера) та іншу важливу інформацію.

Також важливо вказати, що взаємодія з деякими ендпоінтами може вимагати аутентифікації та авторизації, які зазвичай виконуються за допомогою токенів, що передаються в заголовку Authorization запиту.

2.1.1.4 Інтерфейс взаємодії з PowerBI

Взаємодія з PowerBI виконується за наступними способами [11].

- Імпорт даних: Це найпростіший спосіб завантаження даних. Ви можете імпортувати дані з різних джерел, таких як Excel, CSV, XML, JSON, SQL Server, Oracle, MySQL, SharePoint, Salesforce, і багато інших. Дані, імпортовані таким способом, зберігаються в пам'яті та можуть бути повністю оброблені в Power BI.
- Пряме з'єднання (DirectQuery): DirectQuery - це функція, яка дозволяє Power BI взаємодіяти безпосередньо з джерелом даних для отримання і оновлення даних. Він не завантажує дані в пам'ять, а натомість формує SQL-запити для отримання даних на

льоту. Це корисно для великих наборів даних, які перевищують обмеження пам'яті.

2.1.1.5 Інтерфейси для тестування і відлагодження

Для тестування та відлагодження наша система надає інтерфейс логування, який записує події та помилки в системні логи. Це допомагає виявити та усунути проблеми.

2.1.2 Протоколи та стандарти

- JSON (JavaScript Object Notation) - легкий формат обміну даними, що є простим для людей для читання та написання, а також простим для машин для генерування та аналізу. JSON стандарт визначений в RFC 8259.
- Server-Sent Events (SSE) - це стандарт, що дозволяє веб-серверу пушити дані до клієнта. Він визначений в розділі 9 специфікації HTML5. SSE використовує HTTP з'єднання для передачі повідомлень в форматі текстових подій [12].
- HTTP REST: REST (Representational State Transfer) це архітектурний стиль, що використовується в веб-службах. Він не має конкретного стандарту, але базується на стандартних HTTP методах, таких як GET, POST, PUT, DELETE і т.д. Він описаний в дисертації Роя Філдінга [13].
- MongoDB - це розподілена база даних, яка використовує формат подібний до JSON для зберігання даних. Хоча MongoDB не є стандартом, вона має власний протокол для інтеракції з серверами баз даних. MongoDB також використовує JavaScript для своїх функцій запитів та агрегації.
- CSV (Comma-Separated Values): CSV - це простий формат файлу, що використовується для зберігання табличних даних, таких як електронна таблиця або реляційна база даних. Кожен рядок у файлі CSV відповідає

запису даних, а коми використовуються для відокремлення значень полів. RFC 4180 визначає формат CSV.

2.1.3 Безпека

У системі використовується безпека за допомогою API ключів

API ключі є унікальними ідентифікаторами, які використовуються для визначення суб'єкта або користувача, який робить запит до API. Це дозволяє серверу перевірити, чи має суб'єкт доступ до вказаного ресурсу або служби.

Декілька аспектів безпеки, пов'язаних з використанням API ключів [15]:

Створення API ключів: API ключі повинні бути унікальними та складними для вгадування. Вони зазвичай генеруються автоматично системою та можуть включати велику комбінацію букв, цифр та спеціальних символів.

Зберігання API ключів: API ключі повинні зберігатися в безпечному місці і не повинні бути доступні невповноваженим користувачам. Не рекомендується вбудовувати API ключі безпосередньо в код, особливо якщо він відкритий для загального доступу.

Використання API ключів: При використанні API ключів важливо використовувати захищені протоколи передачі даних, такі як HTTPS, для запобігання перехоплення ключа під час передачі.

Обмеження використання API ключів: API ключі можуть бути налаштовані на доступ до певних ресурсів або операцій, що обмежує можливість зловмисного використання ключа, якщо він стане відомим.

2.1.4 Масштабування та продуктивність

Наразі, система має лише можливість вертикального масштабування у рамках одного серверу. Але у перспективі є можливість підключення механізму Sharding MongoDB.

Sharding, або горизонтальне масштабування, в MongoDB - це процес розподілу даних по кількох серверах. MongoDB використовує шардування для

підтримки розгортання даних великого обсягу і для забезпечення високого рівня виконання операцій.

Шардування даних в MongoDB відбувається на рівні колекцій, і документи в шардованих колекціях розподіляються по шардам (серверам). Ключ шардування, який вибирається при створенні шардованої колекції, визначає, як документи будуть розподілятися. З правильно налаштованим ключем шардування, дані можуть бути ефективно збалансовані та запити можуть бути ефективно маршрутизовані [16] .

2.2 ВНУТРІШНЄ ПРОЕКТУВАННЯ

Система складається за наступних частин.

- Завантажувач даних
- Система аналізу даних
- Програма візуалізації даних

Поток даних у системі виглядає наступним чином.

1. Завантажувач даних підключається до API сервісу EasyWay та отримує моментальні знімки місця знаходження машин.
2. Завантажувач даних накопичує знімки до бази даних.
3. Система аналізу даних отримує набір даних про місця знахождення машин у часі для подальшого аналізу.
4. Система аналізу даних робить статистичний аналіз даних, відфільтровує помилкові дані, робить розрахунки та зменшує об'єм даних для подальшої візуалізації.

Програма візуалізації даних отримує та візуалізує відфільтровані дані з розрахунками.

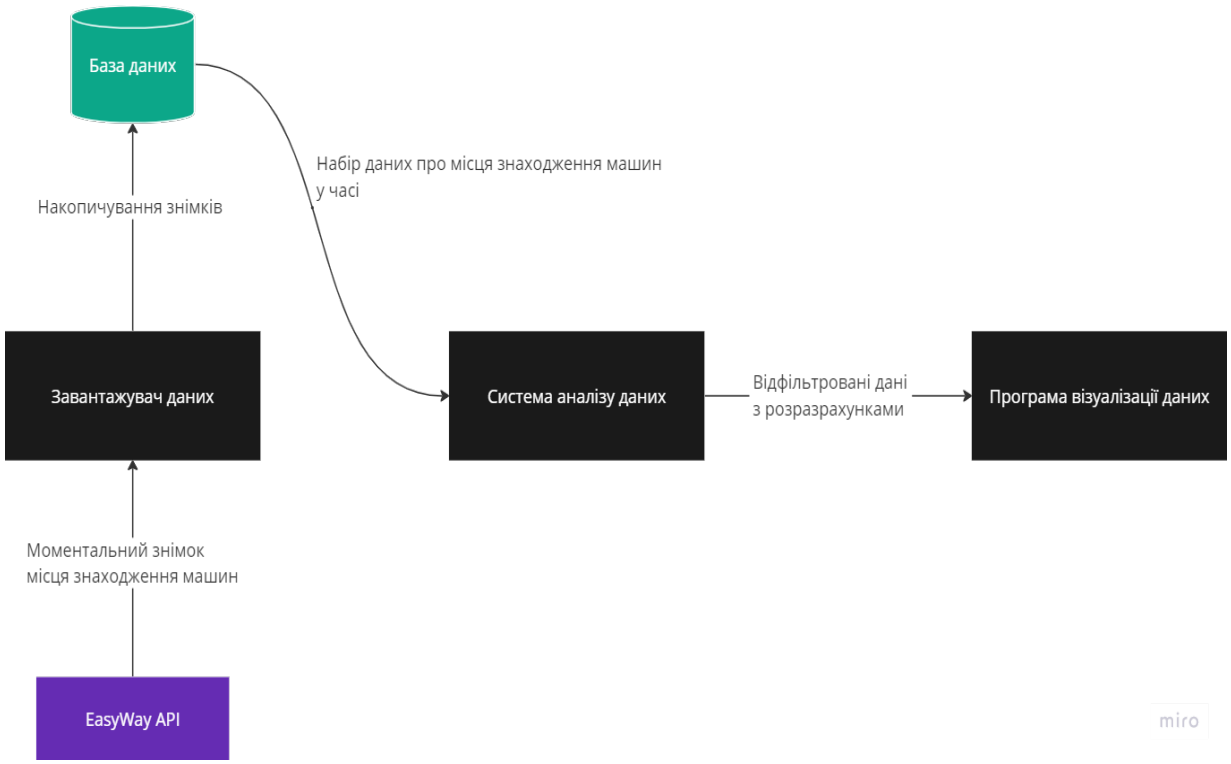

Рисунок 2.1 – Діаграма взаємодії компонентів системи

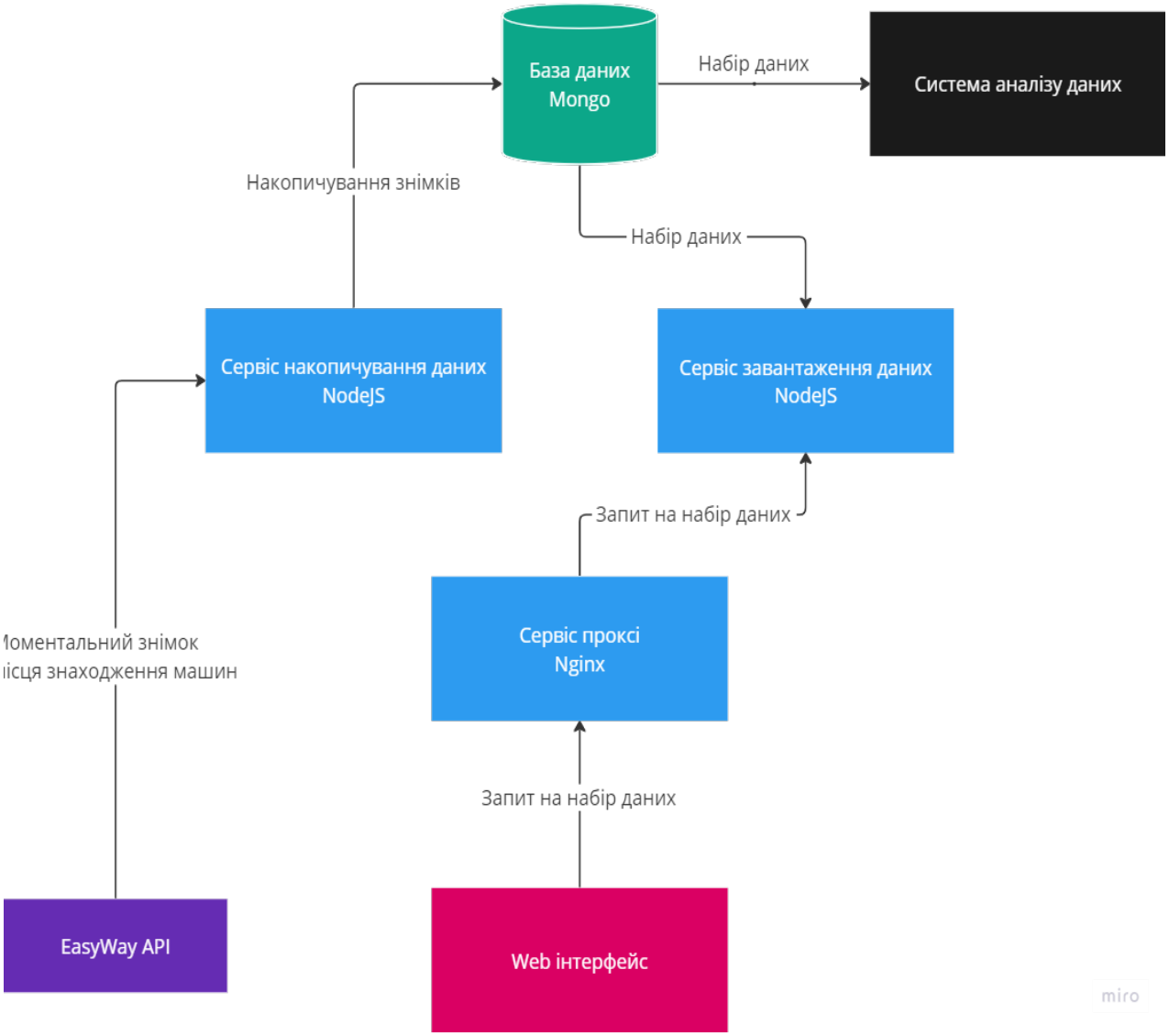

Рисунок 2.2 – Діаграма взаємодії компонентів завантажувача даних

2.2.1 Завантажувач даних

Завантажувач даних має наступні відповідальності.

- Накопичення моментальних знімків у базі даних.
- Обробка HTTP запитів на набори даних.

2.2.1.1 Структура і аналіз бази даних

2.2.1.1.1 Аналіз бази даних

Для вибору бази даних було вирішено зробити аналіз існуючих рішень.

Спочатку, зроблено аналіз переваг NoSQL та SQL [14] . Результати можна побачити у таблиці знизу.

Таблиця 2.1 – Порівняльні характеристики SQL та NoSQL баз даних

Характеристики	**NoSQL бази даних**	**SQL бази даних**
Структура даних	Гнучка, недокладна	Сувора, жорстка
Схема даних	Схема може бути змінена під час роботи	Статична схема, потребує попереднього визначення
Масштабованість	Легко масштабується горизонтально	Менш гнучка масштабованість, зазвичай вертикально
Запити	Прості та швидкі	Складніші, оптимізовані для складних операцій
Даний обсяг	Підтримує великі обсяги даних	Підтримує даний обсяг
Підтримка зв'язків	Обмежена підтримка	Сильна підтримка зв'язків
Спільна робота	Гнучка асинхронна реплікація та шарування	Синхронна реплікація та шарування
Дані збереженість	Гарантує високий рівень надійності даних	Гарантує високий рівень надійності даних
Транзакції	Менша підтримка транзакцій	Сильна підтримка транзакцій
Приклади рішень	MongoDB, Cassandra, Redis	MSSQL, MySQL, PostgreSQL, Oracle

Для нашої структури є важним, щоб усі дані були записані, навіть якщо їх структура не відповідає початковим правилам. Також дуже важливими є пункти про обсяг даних та масштабованість, бо об’єм даних безперервно зростає.

В результаті аналізу було вибрано NoSQL бази даних.

Далі, було зроблено аналіз існуючих рішень NoSQL баз даних.

- MongoDB – документоорієнтована база даних, яка працює з форматом даних BSON, схожим на JSON. MongoDB добре підходить для додатків, які мають великі обсяги даних та потребують гнучкості схеми, швидкості та масштабованості.

MongoDB підтримує шардування для горизонтального масштабування.

- Cassandra – розподілена система ключ-значення, розроблена для обробки великих обсягів даних на багатьох комодитних серверах. Cassandra надає високу доступність без єдиного точки збою і відмінно підходить для додатків, які не можуть терпіти простої.
- Redis – високошвидкісне сховище даних у пам'яті, яке використовується як база даних, кеш та брокер повідомлень. Redis підтримує різні типи даних, такі як рядки, множини, списки, хеши, геопросторові індекси з радіусним запитом та публікацію/підписку на повідомлення.
- Apache HBase – розподілена, масштабована база даних, зорієнтована на великі таблиці, яка забезпечує швидкий доступ до даних на великій кількості вузлів. HBase часто використовується з Apache Hadoop і Apache Phoenix.
- Amazon DynamoDB – повністю керована NoSQL база даних, що надає швидкі та прогнозовані часи відповіді з будь-якою масштабованістю. Ідеально підходить для великих веб-масштабних додатків через свої гнучкі моделі даних та низьку латентність.
- Google Cloud Datastore – гнучке, безсхемне NoSQL для веб, мобільних та IoT додатків.

В результаті аналізу було вибрано MongoDB, бо нам важливі наступні критерії.

- Максимальна гнучкість формату.
- Масштабованість у перспективі.

2.2.1.1.2 Структура бази даних

До бази даних записуються JSON-об’єкти, які приходять під сервісу EasyWay.

У базі даних присутня одна колекція “positions_history”, яка має наступну структуру.

Таблиця 2.2 – Опис структури документа колекції “positions_history”

Шлях у документі	**Тип даних**
_id	ObjectId
positions	Object
positions.x	Array
positions.x.s	String
positions.x.hc	Int32
positions.x.wf	Int32
positions.x.ac	Int32
positions.x.d	Int32
positions.x.vi	Int32
positions.x.vn	String
positions.x.ang	Int32
positions.x.t	Int32
positions.x.idx	Int32
positions.x.idx_app_r	Int32
positions.x.idx_app_nb	Int32
positions.x.time	Int32
positions.x.spd	Int32
positions.x.pos	String
positions.x.spd_raw	Int32

positions.x.spd_ai	Int32
positions.x.spd_test	Int32
timestamp	Date
routes	Object
routes.x	Object
routes.x.name	String
routes.x.tr	String

Приклад документа можна побачити нижче.


```
{
        "_id": ObjectId("61cc830acdbccd7069be2d25"),
        "positions": {
                "119": [
                        {
                                "s": "OK",
                                "hc": 0,
                                "wf": 0,
                                "ac": 0,
                                "d": 0,
                                "vi": 50273,
                                "vn": "1427",
                                "ang": 72,
                                "t": 1135,
                                "idx": 185,
                                "idx_app_r": 186,
                                "idx_app_nb": 186,
                                "time": 1640792779,
                                "spd": 15,
                                "spd_raw": 0,
                                "spd_ai": 9,
                                "pos": "48.497631:34.949228"
                        },
                        {
                                "s": "OK",
                                "hc": 0,
                                "wf": 0,
                                "ac": 0,
                                "d": 1,
                                "vi": 50232,
                                "vn": "1381",
                                "ang": 127,
                                "t": 1133,
                                "idx": 25,
                                "idx_app_r": 474,
                                "idx_app_nb": 26,
                                "time": 1640792776,
                                "spd": 15,
                                "spd_raw": 0,
                                "spd_ai": 18,
                                "pos": "48.404515:35.065659"
```

```
                }
            ],
            "123": [
                {
                    "s": "OK",
                    "hc": 0,
                    "wf": 0,
                    "ac": 0,
                    "d": 0,
                    "vi": 51745,
                    "vn": "2233",
                    "ang": 108,
                    "t": 1139,
                    "idx": 353,
                    "idx_app_r": 354,
                    "idx_app_nb": 354,
                    "time": 1640792777,
                    "spd": 15,
                    "spd_raw": 0,
                    "spd_ai": 15,
                    "pos": "48.501043:34.909227"
                },
                {
                    "s": "OK",
                    "hc": 0,
                    "wf": 0,
                    "ac": 0,
                    "d": 1,
                    "vi": 51751,
                    "vn": "1501",
                    "ang": 98,
                    "t": 1138,
                    "idx": 228,
                    "idx_app_r": 726,
                    "idx_app_nb": 229,
                    "time": 1640792769,
                    "spd": 15,
                    "spd_raw": 19,
                    "spd_ai": 11,
                    "pos": "48.438233:34.926422"
                }
            ]
        },
        "timestamp": ISODate("2021-12-29T15:46:52.000Z"),
        "routes": {
            "119": {
                "name": "5",
                "tr": "tram"
            },
            "123": {
                "name": "19",
                "tr": "tram"
            }
        }
}
```

Лістинг 2.2 – Приклад документа колекції “positions_history”

2.2.2 Система аналізу даних

Система аналізу даних базується на Python Jupyter Notebooks, SQL Server та Excel, представляє собою взаємодію між цими трьома основними компонентами.

2.2.2.1 SQL Server

SQL Server – основний компонент для зберігання та управління даними. Це система управління реляційними базами даних (RDBMS), яка використовується для створення, обслуговування та управління великими обсягами структурованих даних. В нашому випадку, SQL Server служить як основне джерело даних для подальшого аналізу [17].

2.2.2.2 Python Jupyter Notebooks

Python Jupyter Notebooks – інтерактивне середовище для написання коду, створення візуалізацій та документування результатів. В даному контексті, Jupyter Notebooks використовується для отримання даних з SQL Server, їх аналізу та створення візуалізацій. Python має великий набір бібліотек для аналізу даних, таких як pandas для обробки даних, Matplotlib та Seaborn для візуалізації даних, що робить його могутнім інструментом для аналізу даних [2], [3].

2.2.2.3 Excel

Excel – популярний інструмент для обробки та візуалізації даних. В контексті нашої системи, Excel використовується як кінцева точка відображення для даних, що були аналізовані та оброблені в Jupyter Notebooks. Це дозволяє представити дані в структурованій, зручній для читання формі, що може бути використана для подальшого аналізу або презентації.

Спілкування між цими компонентами відбувається за допомогою SQL запитів до SQL Server для отримання даних, Python коду в Jupyter Notebooks для аналізу цих даних та формату CSV або Excel для експорту результатів до Excel.

2.2.3 Система візуалізації даних

Для візуалізації даних було вирішено розробити програму на основі Power BI.

2.2.3.1 Обґрунтування вибору Power BI від Microsoft у якості платформи для аналізу і візуалізації даних

У квітні 2023 Gartner, Inc., світовий лідер у галузі IT досліду та консалтингу, опублікувала свій черговий “магічний квадрант” для платформ аналітики та бізнес-аналітики (Magic Quadrant for Analytics and Business Intelligence Platforms) [9], [10].

Рисунок 2.3 – Магічний квадрант для платформ аналітики та бізнес-аналітики від Gartner, Inc.

У цій публікації Gartner зазначив, що Microsoft є лідером цього аналізу ринка. Його основна ABI платформа, Power BI, має величезне ринкове охоплення та динаміку завдяки інтеграції Microsoft 365, Azure та Teams, гнучкому ціноутворенню, функціональності, що є вище середнього, і амбітній дорожній карті продукту.

У 2022 році Microsoft додала можливість відстеження показників, яка дає змогу командам узгоджувати свої цілі та ключові пріоритети у спільному візуальному досвіді. Нова Premium Gen 2 версія підвищує співвідношення

ціна/продуктивність і вартість завдяки можливостям автоматичного масштабування.

Нарешті, Microsoft додала вітрини даних із кодом низького рівня (low-code data marts) для легкого доступу до самообслуговуваних керованих рішень реляційної аналітики.

2.2.3.1.1 Сильні сторони

Узгодженість із Microsoft 365, Teams і Azure Synapse: включення Power BI у Microsoft 365 E5 забезпечило величезний канал для поширення платформи. Оскільки багато клієнтів звертаються до Teams для віддаленої співпраці, можливість доступу до Power BI, а тепер і до «Цілей» в одному інтерфейсі Teams є переконливою інтеграцією для бізнес-користувачів. Узгодження Power BI і Azure Synapse стосується кількох персонажів D&A та випадків використання.

Комбінація ціна/вартість: служба Power BI тепер пропонує індивідуальну пропозицію для невеликих організацій із 300 або менше співробітниками. Великі організації все ще можуть обрати варіант із розрахунком на ємність, який, як правило, є економічно ефективнішим із більшою кількістю користувачів. Microsoft не використовує стратегію перехресних продажів для збільшення доходу на клієнта, як це роблять більшість постачальників платформи ABI.

Потужний портфель і амбіції щодо продукту: Microsoft має чітке бачення перехресного використання Power BI, Power Apps і Power Automate для підвищення цінності бізнесу. Power Apps можна вбудовувати в інформаційні панелі Power BI або мати доступ до наборів даних Power BI. Потоки Power Automate можна створювати для виконання різних дій на основі даних. Сервіси на основі штучного інтелекту, як-от аналітика тексту, настроїв і зображень, доступні в Power BI Premium.

2.2.3.1.2 Слабкі сторони

Управління створенням контенту і публікації: Gartner зазначив, що отримує все більше запитів від клієнтів Power BI, яким важко керувати

процесом створення та публікації аналітичного контенту. Клієнти висловлюють занепокоєння щодо наявності багатьох способів виконання більшості завдань, таких як моделювання даних або просування контенту. Наприклад, завдання моделювання даних можна виконувати за допомогою наборів даних, вітрин даних, потоків даних і Dataverse. Завдяки низькій вартості та легкому налаштуванню, застосування Power BI має тенденцію швидко поширюватися, як слідство, важко застосовувати стандартні практики керування (standard governance practices).

Обмежена headless відкрита архітектура: хоча більшість клієнтів, що використовують Power BI цінують тісну інтеграцію архітектури Microsoft, зростає попит на більшу сумісність між конкуруючими платформами BI. Зокрема, як аналітичний каталог і сховище показників, багато клієнтів Power BI хотіли б бачити більш відкриту headless інтеграцію з продуктами, що конкурують з Microsoft.

Azure як єдиний варіант розгортання: корпорація Майкрософт не надає клієнтам гнучкості у виборі хмарної пропозиції IaaS. У той час як підключення до даних дозволяє багатохмарні та гібридні хмарні сценарії, його служба Power BI працює лише в Azure. Однак клієнти, які використовують Azure, можуть скористатися можливостями глобального охоплення та мультигеографії, які пропонує хмарна платформа Microsoft.

Підсумовуючи, Power BI — це комплексне рішення для аналітики даних і бізнес-аналітики, яке, пропонує підготовку даних, візуальне виявлення даних, інтерактивні інформаційні панелі та розширену аналітику. Воно доступне як варіант SaaS, що працює в хмарі Azure, або як локальний параметр на сервері звітів Power BI». Щотижневі оновлення хмарної служби Power BI: більше розширеної аналітики у формі досвіду, наповненого штучним інтелектом, включаючи розумні наративи (NLG) і можливості виявлення аномалій для готових візуальних елементів.

Відносно UX & UI: Power BI задумувався як корпоративний інструмент, призначений для використання бізнес-користувачами. Очевидно,

що його основними користувачами є аналітики даних і консультанти BI. Бізнес-концепція платформи робить її одним із інструментів бізнес-аналізу з найкращою зручністю використання та інтерфейсом користувача.

З точки зору економіки: має неперевершене поєднання ціни та потужності: Power BI є однією з причин нещодавнього різкого зниження цін на платформи ABI. Важко конкурувати з технічними можливостями і можливостями, які пропонують як безкоштовна, так і платна версії, які, порівняно з іншими платформами, є дуже економічними.

3 РОЗРОБКА ПРОГРАМИ

3.1 Аналіз даних

3.1.1 Тестування статистичних гіпотез

Статистична гіпотеза – це припущення щодо параметра сукупності.

Перевірка гіпотези – це формальний статистичний тест, який використовується, щоб відхилити або не відхилити статистичну гіпотезу.

Перевірка статистичної гіпотези — це метод статистичного висновку, який використовується для визначення того, чи достатньо наявні дані підтверджують певну гіпотезу. Перевірка гіпотез дозволяє нам робити ймовірнісні твердження щодо параметрів сукупності.

Процес тестування складається з таких етапів:

1. Існує початкова дослідницька гіпотеза, істинність якої невідома.
2. Першим кроком є формулювання відповідної нульової H0 та альтернативної гіпотез H1. Це важливо, оскільки неправильне формулювання гіпотез зіпсує решту процесу.
3. Другим кроком є врахування статистичних припущень, зроблених щодо вибірки під час виконання тесту. Наприклад, припущення про статистичну незалежність або про форму розподілу спостережень. Це також важливо, оскільки недійсні припущення означатимуть, що результати тесту є недійсними.
4. Вирішіть, який тест підходить, і визначить відповідну тестову статистику, T (наприклад f-Test Фішера, t-Test Стьюдента, інше).
5. Отримайте розподіл тестової статистики за нульовою гіпотезою з припущень. У стандартних випадках це буде добре відомий варіант. Наприклад, тестова статистика може відповідати t-розподілу Стьюдента з відомими ступенями свободи або нормальному розподілу з відомими середнім і дисперсією. Якщо розподіл тестової статистики повністю фіксується нульовою гіпотезою, гіпотезу зветься простою, інакше її називають складеною.

6. Виберіть рівень значущості Альфа (α), поріг ймовірності, нижче якого нульову гіпотезу буде відхилено. Зазвичай використовуються значення 5%(0.05) або 1% (0.01).
7. Розподіл тестової статистики за нульовою гіпотезою поділяє можливі значення T на ті, для яких нульову гіпотезу відхилено — так звану критичну область — і ті, для яких вона не відхилена. Імовірність того, що T відбудеться в критичній області за нульовою гіпотезою, дорівнює α. У випадку складної нульової гіпотези максимальна ймовірність дорівнює α.
8. Обчисліть на основі спостережень спостережуване значення t_{obs} тестової статистики T.
9. Вирішіть або відхилити нульову гіпотезу на користь альтернативи, або не відхиляти її. Правило прийняття рішення полягає в тому, щоб відхилити нульову гіпотезу H0, якщо спостережуване значення t_{obs} знаходиться в критичній області, і не відхилити нульову гіпотезу в іншому випадку.

   Альтернативне формулювання Правила прийняття рішення:

   - Обчисліть на основі спостережень спостережуване значення t_{obs} тестової статистики T.
   - Обчисліть p-значення. Це ймовірність, згідно з нульовою гіпотезою, вибірки тестової статистики, принаймні такого ж значення (не менш екстремального), як було спостережено під час експерименту.
   - Відхиляйте нульову гіпотезу на користь альтернативної гіпотези тоді і тільки тоді, коли p-значення менше (або дорівнює) порогу (α) рівня значущості (вибраної ймовірності), наприклад 0,05 або 0,01 (див. вище) [18].

3.1.2 t-Test Стьюдента

Тест Стьюдента (також: t-Test, T-критерій) - це висновувальна статистика (inferential statistic), яка використовується для визначення того, чи існує значна різниця між середніми значеннями двох груп і як вони пов’язані.

T-тест із двома вибірками завжди використовує таку нульову гіпотезу:

H0: $\mu_1 = \mu_2$ (два середні сукупності рівні)

Альтернативна гіпотеза може бути двобічною, лівосторонньою або правосторонньою:

H1 (двосторонній): $\mu_1 \neq \mu_2$ (два середні сукупності не рівні)

H1 (лівосторонній): $\mu_1 < \mu_2$ (середнє значення сукупності 1 менше, ніж середнє значення сукупності 2)

H1 (правий бік): $\mu_1 > \mu_2$ (середнє значення сукупності 1 більше, ніж середнє значення сукупності 2)

Статистика тесту обчислюється як:

$t = (\bar{x}1 - \bar{x}2) / sp(\sqrt{1/n1 + 1/n2})$, де $\bar{x}1$ і $\bar{x}2$ — вибіркові середні значення, n_1 і n_2 — розміри вибірки, і де S_p обчислюється як:

$S_p = \sqrt{(n_1-1)s_1^2 + (n_2-1)s_2^2 / (n_1+n_2-2)}$

де s_1^2 і s_2^2 - дисперсії вибірки.

Інтерпретація результату тесту: якщо p-значення, яке відповідає тестовій статистиці t менше за вибраний рівень значущості (α) ми можемо відхилити нульову гіпотезу. Це означає нерівність середніх значень, і навпаки, якщо p-значення більше (α), ми не можемо відхилити нульову гіпотезу, тобто середні – рівні.

3.1.3 Тест Фішера

f-Test Фішера (також: F-критерій, тест Фішера) у статистиці - це процедура перевірки гіпотез, яка розглядає дві дисперсії з двох вибірок. F-критерій використовується, коли необхідно суттєво оцінити різницю між двома дисперсіями, тобто коли визначають, чи можна взяти дві вибірки як репрезентативні для нормальної сукупності з однаковою дисперсією.

Нульова та альтернативна гіпотези для перевірки такі:

H0: $\sigma_1^2 = \sigma_2^2$ (дисперсії сукупності рівні)

H1: $\sigma_1^2 \neq \sigma_2^2$ (дисперсії сукупності не однакові)

Статистика тесту F обчислюється як s_1^2 / s_2^2.

Інтерпретація результату тесту: якщо p-значення тестової статистики (f-Test) менше певного рівня значущості (α), то нульову гіпотезу відхиляють. Тобто, результатом такого тесту є нерівність дисперсій у двох вибірках.

Якщо p-значення тестової статистики (f-Test) більше певного рівня значущості (α), то ми не можемо відхилити нульову гіпотезу. Дисперсії (варіабельність даних) оцінюються як рівні.

3.1.4 Довірчі інтервали

Довірчі інтервали використовуються для знаходження діапазону значень величини, що оцінюється. Коли нам потрібно отримати одне число як оцінку параметра сукупності, ми використовуємо точкову оцінку. Тим не менш, через помилку вибірки, точкова оцінка не точно дорівнює параметру сукупності при будь-якому розмірі даної вибірки. Часто, замість точкової оцінки, більш корисним підходом буде знайти діапазон значень, в рамках якого, як ми очікуємо, може бути значення параметра з заданим рівнем ймовірності. Цей підхід називається інтервальної оцінкою параметра (англ. Interval estimate of parameter), а довірчий інтервал виконує роль цього діапазону значень.

3.1.4.1 Визначення довірчого інтервалу.

Довірчий інтервал (англ. 'confidence interval') являє собою діапазон, для якого можна стверджувати, із заданою ймовірністю **1 - α** , яка називається ступенем довіри (або ступенем впевненості, англ. 'degree of confidence'), що цей інтервал міститиме оцінюваний параметр.

Цей інтервал часто згадується як 100 * (1- α) % довірчий інтервал для параметра, наприклад, 95%-довірчий інтервал середнього значення сукупності.

Кінцеві значення довірчого інтервалу називаються нижньою та верхньою довірчими межами (або довірчими межами або граничною похибкою, англ. 'lower/upper confidence limits').

3.1.4.2 Інтерпретація довірчих інтервалів:

Ми інтерпретуємо 95% довірчий інтервал для середнього значення сукупності наступним чином. При вибірці, що повторюється, 95% таких довірчих інтервалів будуть, в кінцевому рахунку, включати в себе середнє значення сукупності.

Наприклад, припустимо, що ми робимо вибірку із сукупності 1000 разів, і на підставі кожної вибірки ми побудуємо 95% довірчий інтервал, використовуючи обчислене вибіркове середнє. Через випадковий характер вибірок ці довірчі інтервали відрізняються один від одного, але ми очікуємо, що 95% (або 950) цих інтервалів включають невідоме значення середнього за сукупністю.

На практиці ми зазвичай не робимо такі вибірки, що повторюються. Тому в практичній інтерпретації, ми стверджуємо, що ми 95% впевнені в тому, що один 95% довірчий інтервал містить середнє за сукупністю. Ми маємо право зробити цю заяву, тому що ми знаємо, що 95% усіх можливих довірчих інтервалів, побудованих аналогічним чином, матимуть середнє за сукупністю [22], [23].

3.1.4.3 Побудова довірчих інтервалів.

Довірчий інтервал для параметра має наступну структуру:

Точкова оцінка ± Фактор надійності × Стандартна помилка, де

- Точкова оцінка - це точкова оцінка параметра (значення вибіркової статистики).
- Фактор надійності (англ. 'reliability factor') - коефіцієнт Z , заснований на передбачуваному розподілі точкової оцінки та ступеня довіри **(1- α)** для довірчого інтервалу.
- Стандартна помилка – це стандартна помилка вибіркової статистики, значення якої отримано за допомогою точкової оцінки.

Величину (Фактор надійності × Стандартна помилка) іноді називають точністю оцінки (англ. 'precision of estimator'). Великі значення цієї величини мають на увазі нижчу точність оцінки параметра сукупності.

Довірчий інтервал зі ступенем довіри $100 * (1- \alpha) \%$ для середнього за сукупністю μ у разі вибірки з нормального розподілу з відомою дисперсією σ^2 задається формулою: Довірчий Інтервал (Confidence Interval) = $\bar{x} \pm z * \frac{\sigma}{\sqrt{n}}$,

Фактори надійності Z для найчастіше використовуваних довірчих інтервалів наведені нижче у таблиці 3.1.

Таблиця 3.1 – Фактори надійності Z залежно від ступенів довіри довірчих інтервалів

Довірчій інтервал	Фактор надійності, Z
90%	1.65
95%	1.96
99%	2.58

3.1.5 Практичне застосування статистичних тестів у роботі і результати

3.1.5.1 Тест №1: Оцінка репрезентативності зробленої вибірки даних трекінгу маршрутного транспорту.

Якщо варіабельність (дисперсія) та середні значення цільового показника (середньої швидкості) у вибірки і у повному масиві даних статистично рівні - вибірка репрезентативна і ми можемо подальший аналіз і візуалізацію проводити на ній. Для оцінки варіабельності застосовуємо тест Фішера, для оцінки рівності середніх – тест Стьюдента. Для проведення тестів f-Test Фішера і t-Test Стьюдента було застосовано пакет статистичного аналізу для Microsoft Excel Analysis ToolPak [24].

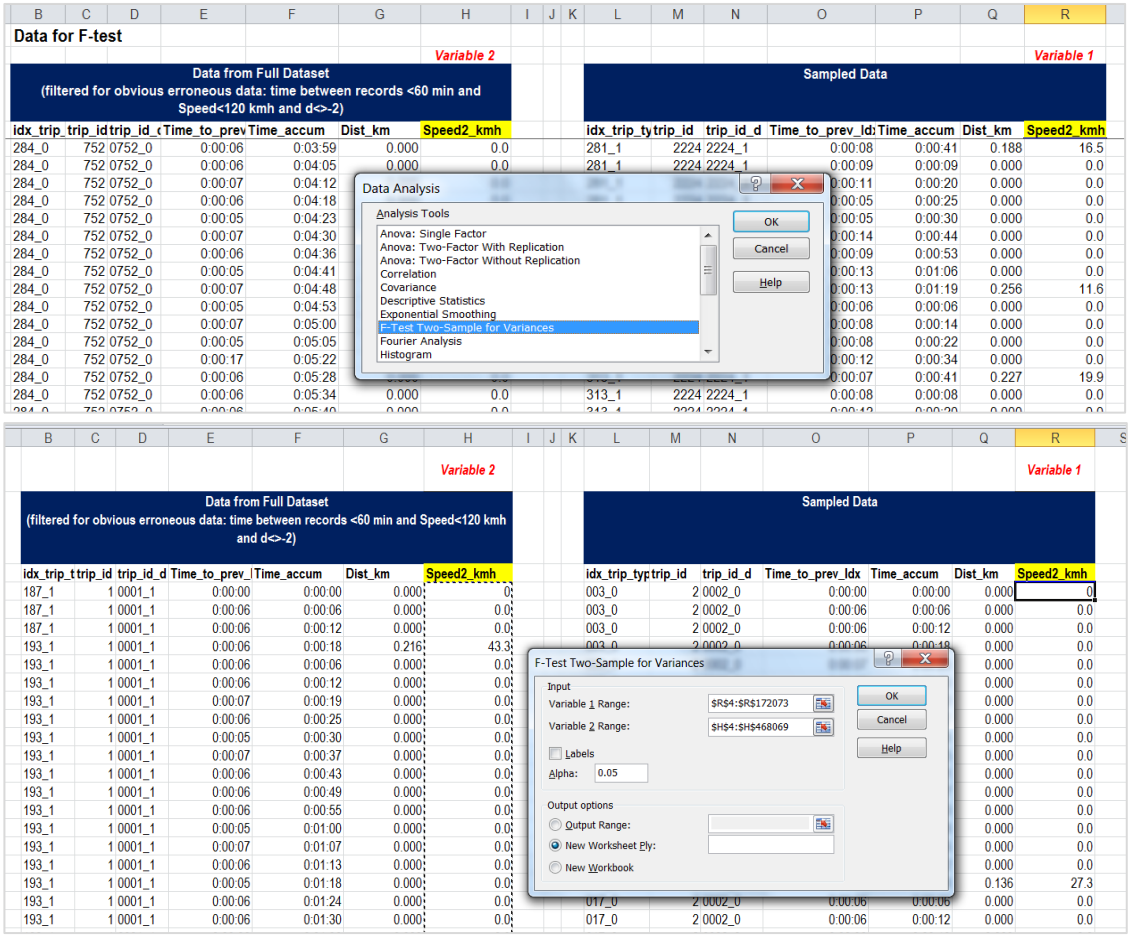

Рисунок 3.1 – Скріншоти реалізації f-Test Фішера за допомогою Microsoft Excel Analysis ToolPak

Формулювання гіпотез для f-Test Фішера: $H0: \sigma_1^2 = \sigma_2^2$; $H1: \sigma_1^2 \neq \sigma_2^2$

Resul of the Calculation		
F-Test Two-Sample for Variances		"Speed2_kmh"
	Variable 1	*Variable 2*
Mean	6.563338984	6.088560395
Variance	174.3628745	173.5470717
Observations	172070	468066
df	172069	468065
F	1.004700758	
P(F<=f) one-tail	0.119538347	
F Critical one-tail	1.006573792	

Рисунок 3.2 – Результат f-Test Фішера, Тест №1

Інтерпретація результату тестування критерієм Фішера (F-Test Two-Sample for Variances): P значення F > Alfa (0,05) --- > ми не можемо відхилити нульову гіпотезу. Це означає рівну дисперсію $\sigma_1^2 = \sigma_2^2$ у наших двох наборах даних.

Формулювання гіпотез для теста Стьюдента: H0: Mean1= Mean2; H1: Mean1 ≠ Mean2

Calculated Average Speed for Idx		
t-Test: Two-Sample Assuming Equal Variances		
	Variable 1	*Variable 2*
Mean	21.63128002	21.49999009
Variance	69.77518543	70.84228778
Observations	806	815
Pooled Variance	70.31170261	
Hypothesized Mean Difference	0	
df	1619	
t Stat	0.315189991	
P(T<=t) one-tail	0.376329012	
t Critical one-tail	1.645795348	
P(T<=t) two-tail	0.752658023	
t Critical two-tail	1.961430331	

Рисунок 3.3 – Результат t-Test Стьюдента, Тест №1

Інтерпретація результату тестування критерієм Стьюдента (t-Test: Two-Sample Assuming Equal Variances): P(T<=t) two-tail значення > Alfa (0,05) --- > ми не можемо відхилити нульову гіпотезу. Це означає рівні середні $Mean_1 = Mean_2$ у наших двох наборах даних.

Загальний висновок за результатами f-Test Фішера і t-Test Стьюдента: наша вибірка – репрезентативна.

3.1.5.2 Тест №2: Оцінка даних двох різних автомобілів, що курсують на маршруті, що аналізується.

Мета: зробити висновок, чи є статистичні різниці у даних двох окремих автомобілів (і водіїв), чи ці різниці статистично не значущі і не треба розділяти ці дані.

Variable 1	Variable 2
AE5645AC	AE5739MH
Speed2_kmh	Speed2_kmh
0	0
0	0
0	0
0	8.252056663
0	0
0	0
0	0
0	28.5816938
0	0
0	0
0	28.80147971
0	0
0	0
0	24.125911
0	0
0	0
27.27050698	0
0	7.881163579
0	0
0	0
13.84626287	0
0	0
0	0
22.69941728	0
0	0
0	0
16.12580445	0
0	0
0	0
0	0
0	5.1673398
0	0.080555278
0	0.040277637
0	26.801538
0	0
0	0

CONCLUSION: Key statistical characteristics of the two subsets of our data (1 vehicle and 2 vehicle) are statistically same. So we could treat the both vehicles as one without disremination among them.

$H_0: \sigma_1^2 = \sigma_2^2$

$H_1: \sigma_1^2 \neq \sigma_2^2$

F-Test Two-Sample for Variances

	Variable 1	*Variable 2*
Mean	6.551494516	6.570655864
Variance	175.3304215	173.7666782
Observations	65706	106364
df	65705	106363
F	1.008999098	
P(F<=f) one-tail	**0.100681833**	
F Critical one-tail	1.011600156	

P(F<=f) one-tail > Alfa (0.05) ----> Can not reject H0, so Variances are equal.

H_0: **Mean$_1$= Mean$_2$**

H_1: **Mean$_1$ ≠ Mean$_2$**

t-Test: Two-Sample Assuming Unequal Variances

	Variable 1	*Variable 2*
Mean	6.551494516	6.570655864
Variance	175.3304215	173.7666782
Observations	65706	106364
Hypothesized Mean	0	
df	138677	
t Stat	-0.292136277	
P(T<=t) one-tail	0.385091434	
t Critical one-tail	1.644864615	
P(T<=t) two-tail	0.770182868	
t Critical two-tail	1.959981091	

P(T<=t) two-tail > Alfa (0.05) ----> Can not reject H0, so Means are equal.

Рисунок 3.4 – Результати f-Test Фішера та t-Test Стьюдента, Тест №2

Як видно з результатів на ілюстрації, і дисперсія (варіабельність) і середні швидкості двох автомобілів на маршруті – статистично рівні. Таким

чином, для цілей аналізу середньої швидкості по маршруту не потрібно аналізувати окремо, треба аналізувати їх дані як єдине ціле.

3.1.5.3 Тест №3:

Оцінка даних пересування по маршруту на двох його частинах “прямий напрямок від однієї кінцевої зупинки до другої кінцевої” і, відповідно, “зворотній напрямок”. Проведення такого аналізу важливе, оскільки ці частини проходять різними вулицями і мають суттєво різну довжину.

Variable 2 D = 0 Speed2_kmh	Variable 1 D = 1 Speed2_kmh
0	0
0	0
0	0
0	41.77827
0	0
0	0
0	40.4765462
0	0
0	0
0	0
0	30.5817778
0	0
0	0
0	32.3539153
0	0
0	0
27.27050698	28.2640889
0	0
0	0
0	0
13.84626287	32.7832222
0	0
0	0
22.69941728	55.6660377
0	0
0	0
16.12580445	60.3088799
0	0
0	0
0	0
0	41.8076743
0	0
0	0
0	59.9563166
0	0

CONCLUSION: Key statistical characteristics of the two subsets of our data (D=0 and D=1) are clearly statistically different. It is logical, as they describe two different sub-routs with different roads, length, etc..

$H_0: \sigma_1^2 = \sigma_2^2$

$H_1: \sigma_1^2 \neq \sigma_2^2$

F-Test Two-Sample for Variances

	Variable 1	*Variable 2*
Mean	7.494263536	6.081280917
Variance	199.9898935	160.413008
Observations	58704	113366
df	58703	113365
F	1.246718679	
P(F<=f) one-tail	**2.7999E-211**	
F Critical one-tail	1.011885583	

P(F<=f) one-tail << Alfa (0.05) ----> Rreject H0, so Variances are NOT equal.

H_0: **Mean$_1$= Mean$_2$**

H_1: **Mean$_1$ ≠ Mean$_2$**

t-Test: Two-Sample Assuming Unequal Variances

	Variable 1	*Variable 2*
Mean	7.494263536	6.081280917
Variance	199.9898935	160.413008
Observations	58704	113366
Hypothesized Mean	0	
df	107951	
t Stat	20.34859352	
P(T<=t) one-tail	3.55659E-92	
t Critical one-tail	1.644867742	
P(T<=t) two-tail	**7.11319E-92**	
t Critical two-tail	1.95998596	

P(T<=t) two-tail << Alfa (0.05) ----> Reject H0, so Means are NOT equal.

Рисунок 3.5. Результати f-Test Фішера та t-Test Стьюдента, Тест №3

Як видно на ілюстрації, і дисперсія (варіабельність) і середні швидкості двох автомобілів на маршруті – статистично суттєво розрізняються.

Це відповідає реальності (частини проходять різними вулицями і мають суттєво різну довжину) і свідчить, що не можна не розрізняти напрямок при подальшому аналізі. Це було б суттєвою помилкою. Детальні дані, розрахунки та висновки цих тестів додаються у файлі Stat Analysis.xlsx.

3.1.5.4 Побудова довірчих інтервалів для середніх значень швидкості

У роботі проведено побудову довірчих інтервалів для середніх значень швидкості для кожної з дистанцій між двома послідовними IDX точками маршруту у розрізі напрямків руху (D=0 та D=1). Розмір масиву: 810 розрахованих середніх значень. Середня кількість записів трекінгу для розрахунку однієї середньої: 80.7.

Розрахунки проведено в Microsoft Excel із застосуванням пакету Microsoft Excel Analysis ToolPak та вбудованої статистичної функції

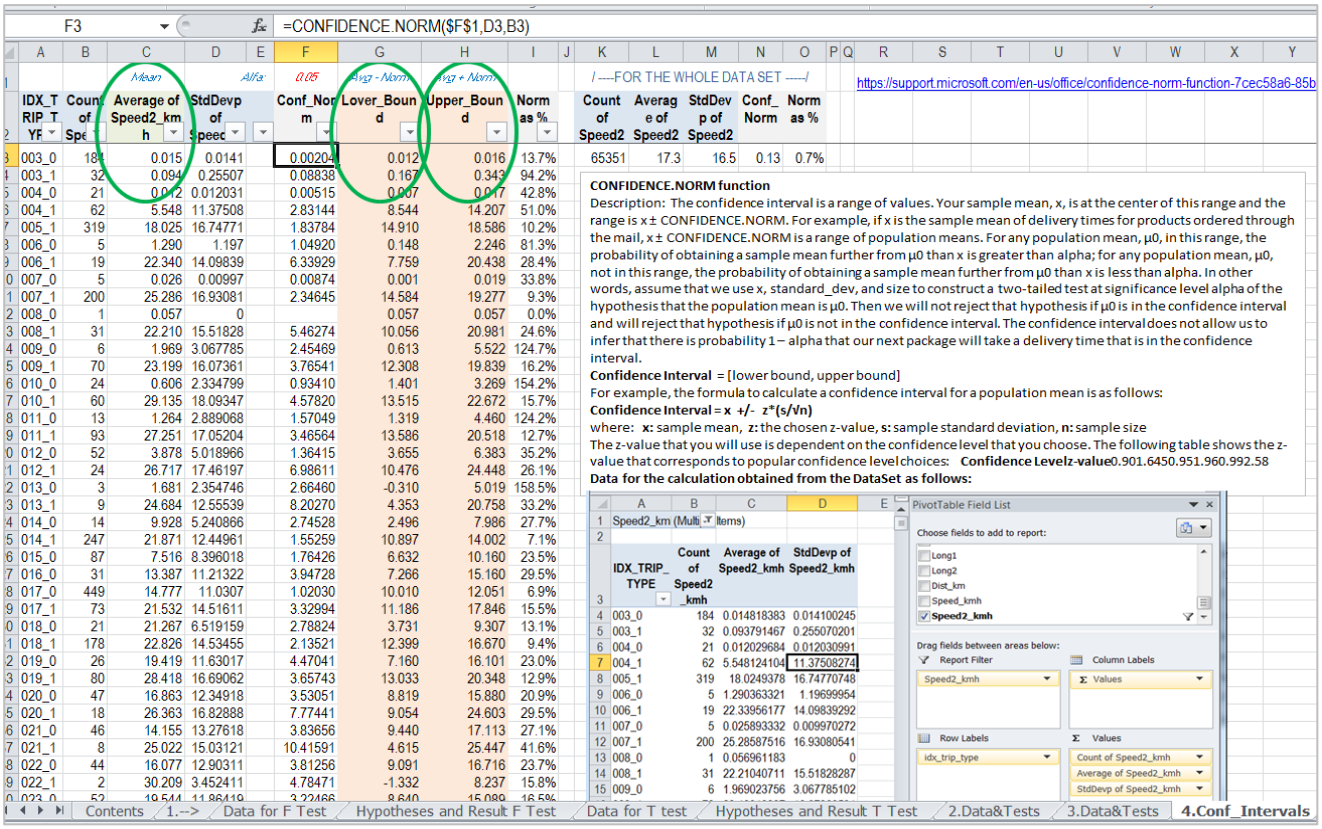

Рисунок 3.6 – Скріншот побудови довірчих інтервалів у Microsoft Excel

CONFIDENCE.NORM function.

Розраховані у розрізі IDX та D довірчі інтервали дають можливість знаходження діапазону значень величини середньої швидкості на окремих ланках маршруту із ступенем довіри (рівнем вірогідності) 95%.

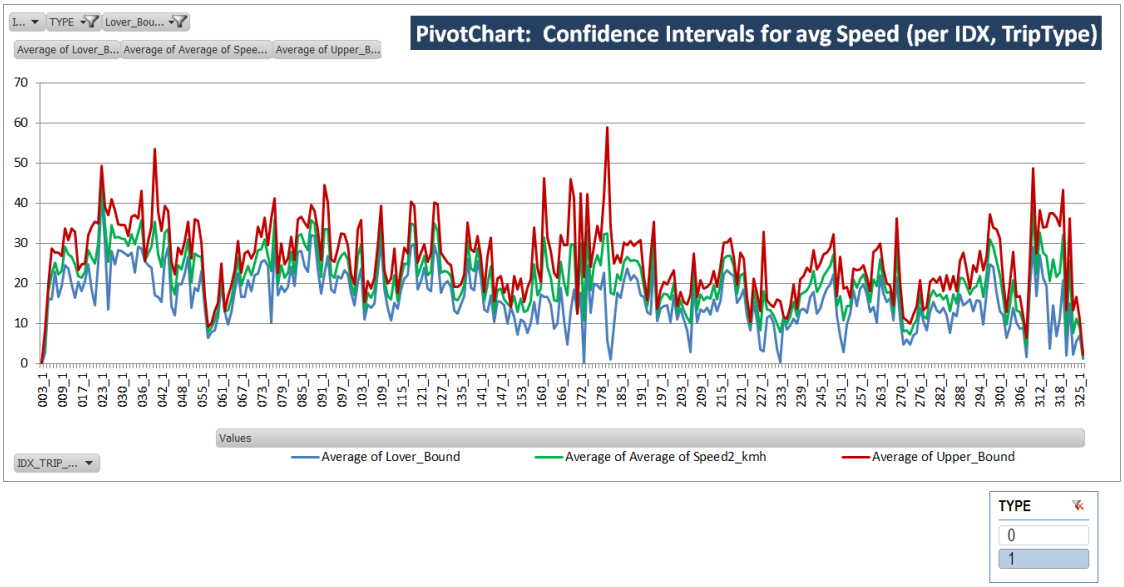

Рисунок 3.7 – Скріншот дашборду для аналізу розрахованих довірчих інтервалів побудованого на базі Зведеної Діаграми (Microsoft Excel PivotChart)

Отриманий розбіг інтервалів є логічним (низький розбіг на ланках зі значною швидкістю і кількістю спостережень і навпаки, значний розбіг там де швидкість і кількість спостережень малі) і цілком задовольняє цілям побудови інструменту для аналізу і прогнозування. Детальні дані та розрахунок додаються у файлі Stat Analysis.xls.

3.2 Розробка системи накопичування даних

3.2.1 Вибір технологій

3.2.1.1 Вибір мови та платформи для розробки

Для розробки програми було обрано платформу Node.js та мову JavaScript.

Під час вибору було розглянуто наступні сильні сторони JavaScript.

- JavaScript може використовуватися як на клієнтській, так і на серверній стороні. Це спрощує розробку та підтримку, оскільки вам потрібно вивчити лише одну мову.
- JavaScript вбудований в браузерах, які підтримують SSE, що дозволяє серверам асинхронно надсилати оновлення клієнту через одне HTTP з'єднання.
- JavaScript використовує подієво-орієнтовану, неблокуючу модель вводу-виводу, що робить його відмінно підходящим для додатків, що працюють з великою кількістю відкритих з'єднань, таких як SSE.
- MongoDB має драйвери та бібліотеки для JavaScript, які полегшують роботу з цією базою даних.
- JavaScript має одну з найбільших спільнот розробників у світі. Це означає, що ви можете знайти велику кількість бібліотек, інструментів, навчальних матеріалів та відповідей на питання про роботу з JavaScript.
- JavaScript використовує JSON для обміну даними, який є одним з найбільш популярних форматів обміну даними і добре підтримується більшістю сучасних баз даних.

Під час вибору було розглянуто наступні сильні сторони Node.js.

- Node.js використовує асинхронний та неблокуючий ввід-вивід, що дозволяє обробляти багато запитів одночасно без блокування основного потоку.
- Модульність: Node.js використовує систему пакетів NPM, яка є однією з найбільших екосистем відкритого програмного забезпечення. Це дозволяє розробникам легко використовувати та ділитися кодом.

- Масштабованість: Node.js добре підходить для масштабованих додатків, завдяки його подіє-орієнтованій та асинхронній природі.

3.2.1.2 Вибір середовища розробки

Для середовища розробки було обрано наступні IDE:

- Jetbrains WebStorm для розробки JavaScript програми;
- Nosqlbooster для управління базою даних MongoDB.

3.2.2 Етапи розробки

Процес розробки поділений на наступні етапи.

1. Розробка модуля для отримання та запису до бази даних.
2. Розробка модуля для вибору даних.
3. Розробка скриптів для розгортання програми.

3.2.3 Процес розробки

3.2.3.1 Розробка модуля для отримання та запису до бази даних

Для отримання даних реалізовано підключення до сервісу EasyWay за допомогою бібліотеки клієнту Server-Sent Events – eventsource.

Для запису до бази даних було використано бібліотеку mongodb для підключення до бази даних.

Під час отримання даних було використано код зі сторінки EasyWay для того, щоб отримати JSON-об’єкт з тексту повідомлення.

3.2.3.2 Розробка модуля вибору даних

Для вибору даних було розроблено HTTP-сервер. Його представлено у вигляді одного файлу.

Імпортуємо залежності за допомогою наступного лістингу.

```
const express = require('express')
const cors = require('cors')
const moment = require('moment')
```

Лістинг 3.1 – Імпорт залежностей

Створюємо новий додаток Express за допомогою наступного коду.

```
const app = express()
const port = 3000

app.use(cors())
```

Лістинг 3.2 – Створення нового додатку фреймворку Express

Задаємо кореневий маршрут за допомогою наступного коду.

```
app.get('/', async (req, res) => {
   res.setHeader('Access-Control-Allow-Origin', '*');
   res.setHeader("Access-Control-Allow-Methods", "*");
   res.setHeader('Access-Control-Allow-Headers', 'origin, content-type, accept');

   const from = moment(req.query.from);
   const to = moment(req.query.to);
   const docs = client.db('admin').collection('positions_history')
      .find({
         timestamp: { $gte: new Date(from.valueOf()), $lte: new Date(to.valueOf()) }
      })
      .sort({ timestamp: 1 })
      .project({ "positions.523": 1, "routes.523": 1, timestamp: 1 });
   const array = await docs.toArray();
   res.send(JSON.stringify(array));
})
```

Лістинг 3.3 – Створення кореневого маршруту.

Додаємо слухача на заданий порт за допомогою наступного коду.

```
app.listen(port, () => {
   console.log(`App is listening at http://localhost:${port}`)
})
```

Лістинг 3.4 – Додавання слухача на заданий порт

3.2.3.3 Розробка скриптів для розгортання програми

Для розгортання програми було обрано Docker контейнери. Для того, щоб їх запустити використано Docker Compose.

Спочатку, було розроблено Dockerfile для Node.js програми, його можна побачити нижче.

```
FROM node:12

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied

COPY package*.json ./

RUN npm install

# Bundle app source
COPY . .

CMD [ "node", "index.js" ]
```

Лістинг 3.5 – Dockerfile для модуля отримання та запису до бази даних

Потім, було розроблено файл “docker-compose.yml” для автоматизування компіляції та запуску контейнерів модуля для отримання даних, бази даних та проксі-сервера. Його можна побачити нижче.

```
version: '3'

services:
  mongo:
    image: mongo
    restart: on-failure
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: ${MONGO_PASSWORD}
      MONGO_INITDB_DATABASE: project
    ports:
      - "27018:27017"
    volumes:
      - ./.docker/storage:/data/db

  scrapper:
    build: .
    restart: on-failure
    ports:
      - "3000:3000"
    depends_on:
      - mongo

  nginx:
    image: nginx
    volumes:
      - ./static:/usr/share/nginx/html
    ports:
      - "8081:80"

volumes:
  redis-data:
```

Лістинг 3.6 – Файл docker-compose.yml

3.2.4 Впровадження та розгортання

Програму було розгорнуто на хмарній платформі Hetzner.

3.2.4.1 Специфікації серверу

Таблиця специфікацій серверу наведена нижче.

Таблиця 3.2 – Таблиця специфікацій серверу

Характеристика	**Значення**
Тип серверу	CPX21
Кількість віртуальних ядер	3
Оперативна пам’ять	4GB
Диск	SSD 80 GB

3.2.4.2 Моніторинг

Додано моніторинг засобами Hetzner. Моніторинг відправляє електронні листи при виникненні наступних ситуацій:

- загрузка процесору більше ніж 60%;
- використання оперативної пам’яті більше ніж 60%;
- використання дискового простору більше ніж 60%;
- сервер не відповідає на запит перевірки здоров’я.

3.2.4.3 Безперервна доставка оновлень

Безперервну доставку оновлень, або CI/CD, було додано за допомогою Bitbucket Pipelines. Спочатку, робиться перевірка на те, що контейнер модуля отримання даних компілюється без помилок. Потім, на сервері завантажуються останні оновлення, та запускається команда для компілювання контейнерів та їх заміни. Команди наведені нижче.

```
cd /app
git pull
docker-compose up --build -d
```

Лістинг 3.7 – Команди, які автоматично виконуються на сервері під час оновлення програми.

3.3 Розробка системи аналізу даних

3.3.1 Вибір технологій

3.3.1.1 Вибір мови та платформи для розробки

Для розробки системи було обрано Python Jupyter Notebook, Microsoft Excel та Microsoft SQL Server.

3.3.1.1.1 Пояснення вибору та використання Python Jupyter Notebook

Jupyter Notebook – це інструмент, який надзвичайно популярний серед дослідників у сфері науки про дані, машинного навчання і штучного інтелекту. Він базується на мові Python і має наступні сильні сторони, які можна використати у даній системі [2].

- Jupyter Notebook дозволяє виконувати код по частинах, що полегшує відлагодження та здійснення ітеративного аналізу.
- Jupyter Notebook підтримує багато бібліотек для візуалізації даних, таких як Matplotlib, Seaborn, Plotly тощо. Це дозволяє вам генерувати графіки прямо в блокноті.
- Jupyter Notebook дозволяє використовувати Markdown і LaTeX для форматування тексту та створення математичних формул. Це робить Jupyter Notebook хорошим інструментом для наукового документування.
- Jupyter Notebook підтримується великою спільнотою і має багато плагінів, які можуть розширити його функціональність.

У даній системі Python Jupyter Notebooks використовується для аналізу, перетворення та візуалізації даних.

3.3.1.1.2 Пояснення вибору та використання Microsoft Excel

Microsoft Excel – це потужний інструмент для обробки і аналізу даних, який широко використовується в бізнесі, освіті, науці і технологіях. Він має наступні сильні сторони, які можна використати у даній системі.

- Excel дозволяє працювати з різними типами даних, включаючи числові, текстові та дати.
- Excel має велику кількість вбудованих функцій і формул, які дозволяють виконувати розрахунки, аналізувати дані, обробляти текст, проводити статистичний аналіз тощо.
- Excel надзвичайно корисний для обробки та аналізу табличних даних, оскільки це його основна функція.
- Excel має велику кількість розширень і плагінів, що дозволяють розширити його функціональність.

У даній системі Microsoft Excel використовується для перегляду та аналізу даних [25].

3.3.1.2 Вибір середовища розробки

Для середовища розробки було обрано наступні IDE:

- Jetbrains WebStorm для розробки JavaScript програми;
- Nosqlbooster для управління базою даних MongoDB.

3.3.2 Етапи розробки

Процес розробки поділений на наступні етапи.

1. Загрузка даних з системи накопичування.
2. Первинний аналіз координат.
3. Аналіз поля idx.
4. Аналіз координат.
5. Аналіз швидкості транспортних засобів.

3.3.3 Процес розробки

3.3.3.1 Загрузка даних з системи накопичування

Для завантаження даних с системи накопичування можна скористатися наступними способами.

- Завантажити csv-експорт з бази даних та прочитати csv-файл у блокноті Jupyter.
- Завантажити дані напряму з бази даних MongoDB за допомогою бібліотеки pymongo.
- Завантажити дані за допомогою протоколу HTTP і модуля вибору даних.

3.3.3.2 Первинний аналіз координат

Для початку, було вирішено проаналізувати координати місць знаходжень транспортних засобів. Для цього, використано сервіс Google Maps.

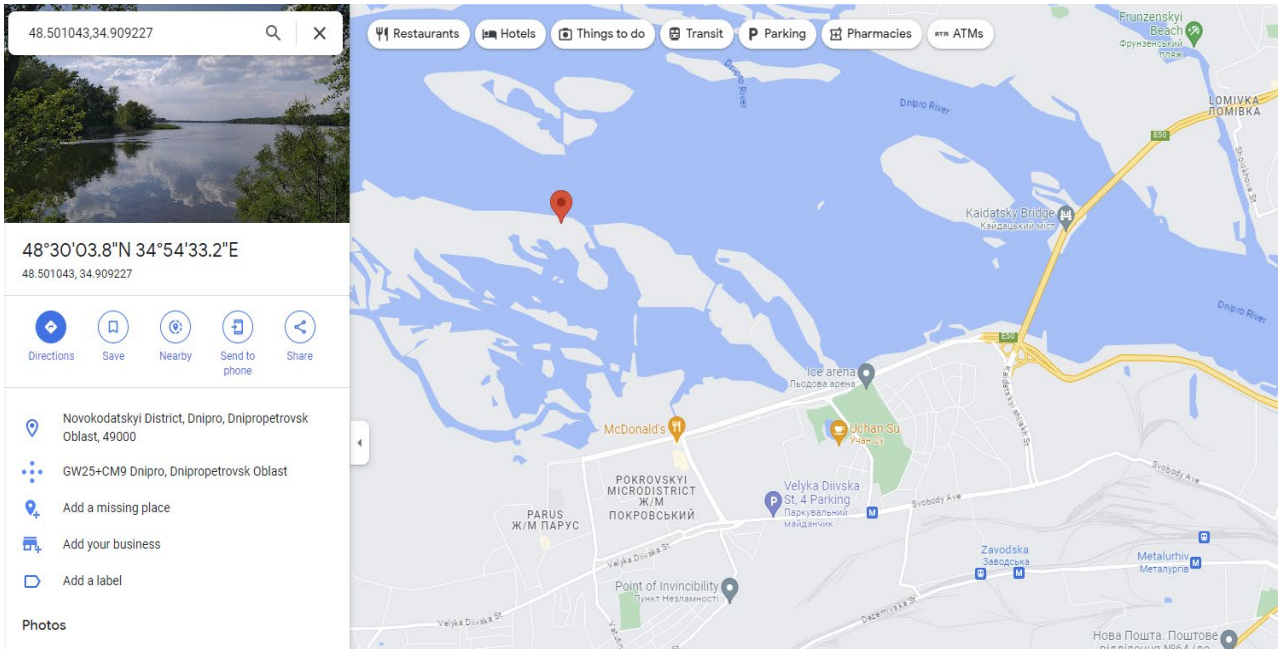

Рисунок 3.8 – Приклад координати, яка не підпадає під маршрут

В результаті аналізу було виявлено, що координати не відповідають реальному маршруту. Приклад такої координати можна побачити на скріншоті вище. Було висунуто гіпотезу, що ці координати проходять через процес перетворення перед відправкою через SSE.

Для вирішення цієї проблеми було вирішено проаналізувати, як ці координати перетворюються на сайті EasyWay.

В результаті аналізу коду сторінки, було виявлено, що перед використанням к координатам застосовується спеціальна функція, вона була

переписана з мови програмування Javascript на Python. В результаті отримано наступний код.

```
def convertToLatLng(c):
    b=c+""
    a=b.split(".")
    if 1 < len(a):
        c=a[0]
        a=a[1]
        e=len(a)
        if 3 < e:
            b=""+a[0]
            for d in range(2, e):
                b+=a[d]
            b+=a[1]
            b= c + "." + b
        c = b
    return c
```

Лістинг 3.8 – Функція для перетворення координат

Після застосування функції зазначеної вище до отриманих координат, проблема із координатам вирішилась.

3.3.3.3 Аналіз поля idx

Далі, було вирішено зробити аналіз поля idx. Це поле відображає поточну позицію транспортного засобу на маршруті.

За одну поїздку було вирішено приймати такий набір послідовних точок, в в кожній з яких idx менше або рівняється idx у попередній точці. Виключенням є перша точка у поїздці.

Для виявлення аномалій було розраховано кумулятивний час на кожну точку у поїздці та побудовано наступний графік.

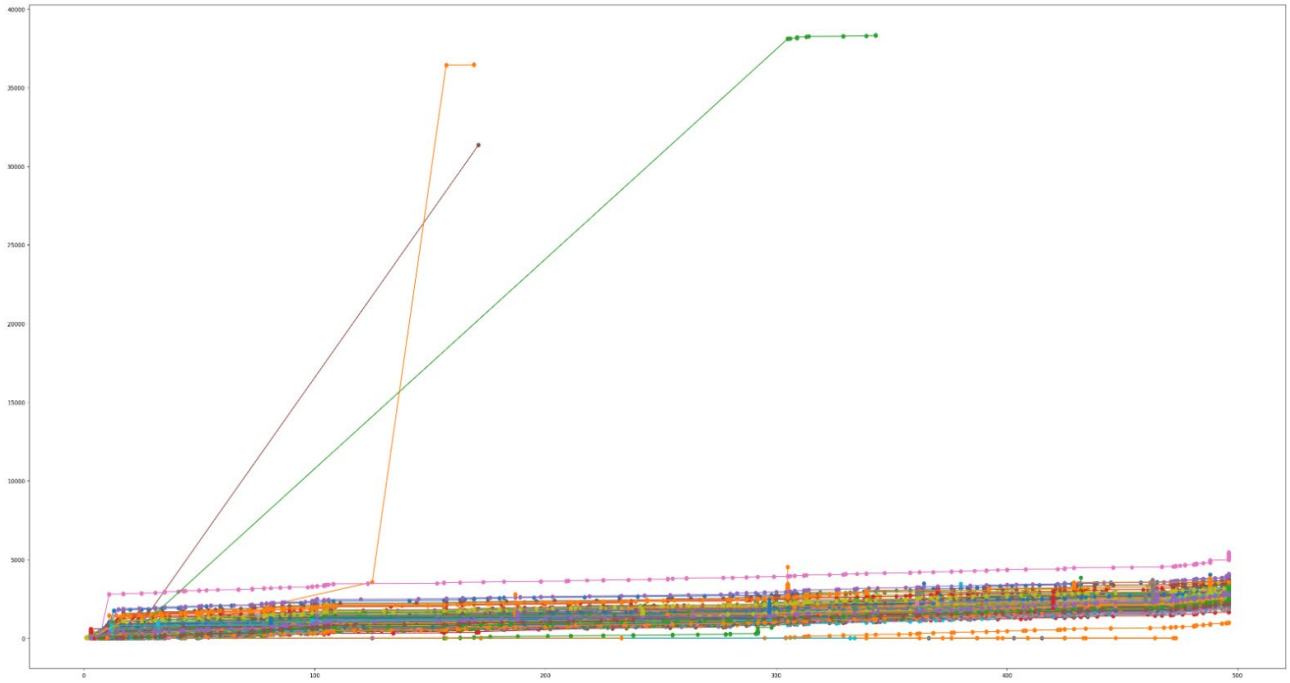

Рисунок 3.9 – Графік залежності кумулятивного часу від idx у поїздках

На цьому графіку по осі X маємо idx, а по осі Y маємо час у секундах. Також на графіку можна побачити, що маємо аномальні поїдки, де у машини, або не працював GPS маячок, або машина закінчила маршрут не у кінці і розпочала вже з середини.

В результаті аналізу графіку було вирішено зробити правила вибору поїздки строгішими.

До правил вибору поїдки додано, що кумулятивний час не має перевищувати 20000 секунд.

В результаті застосування нових правил отримано наступний графік.

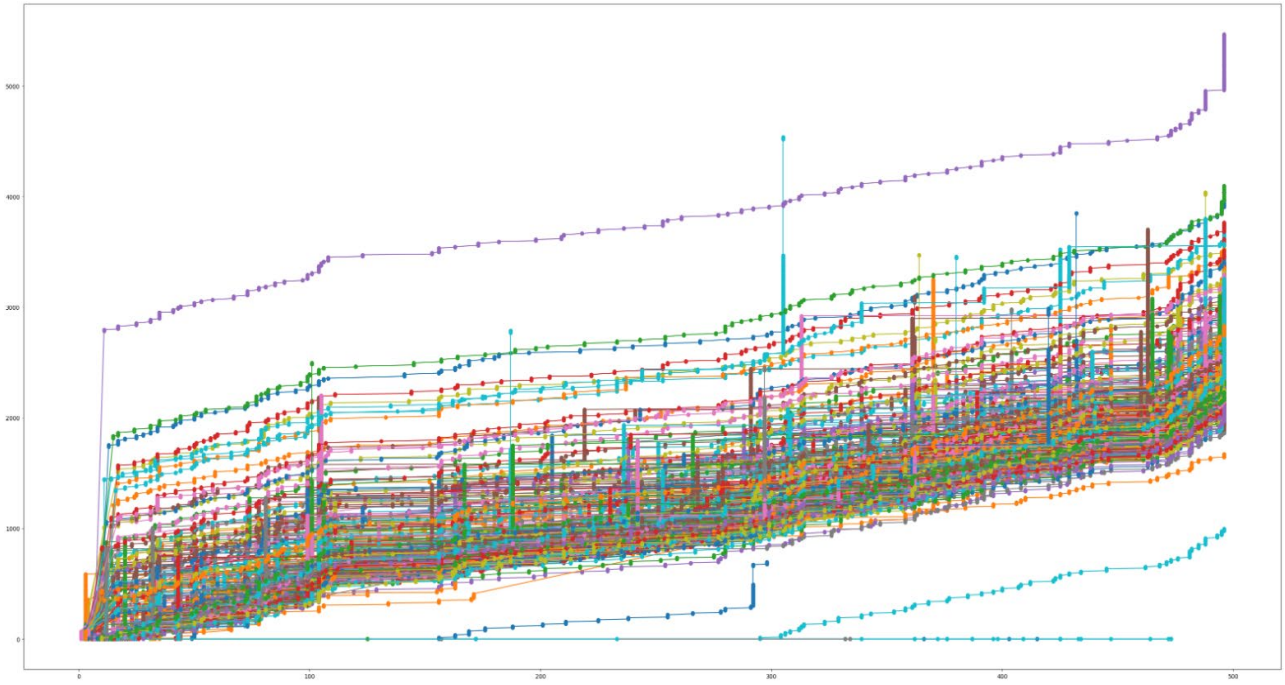

Рисунок 3.10 – Графік після застосування правила 20000 секунд

На цьому графіку бачимо, що попередніх аномалій вже немає, але ще є поїздки, які закінчуються не в кінці та поїздки, які мають пропуски у даних.

В результаті аналізу графіку було вирішено використати бібліотеку для створення графіків plotly.

Перевагою plotly перед стандартною matplotlib є те, що у нас є можливість навести курсор на лінію на графіку та отримати інформацію про цю поїздку, також вагомою перевагою є можливість масштабування графіку.

Після використання бібліотеки plotly, отримано наступний графік.

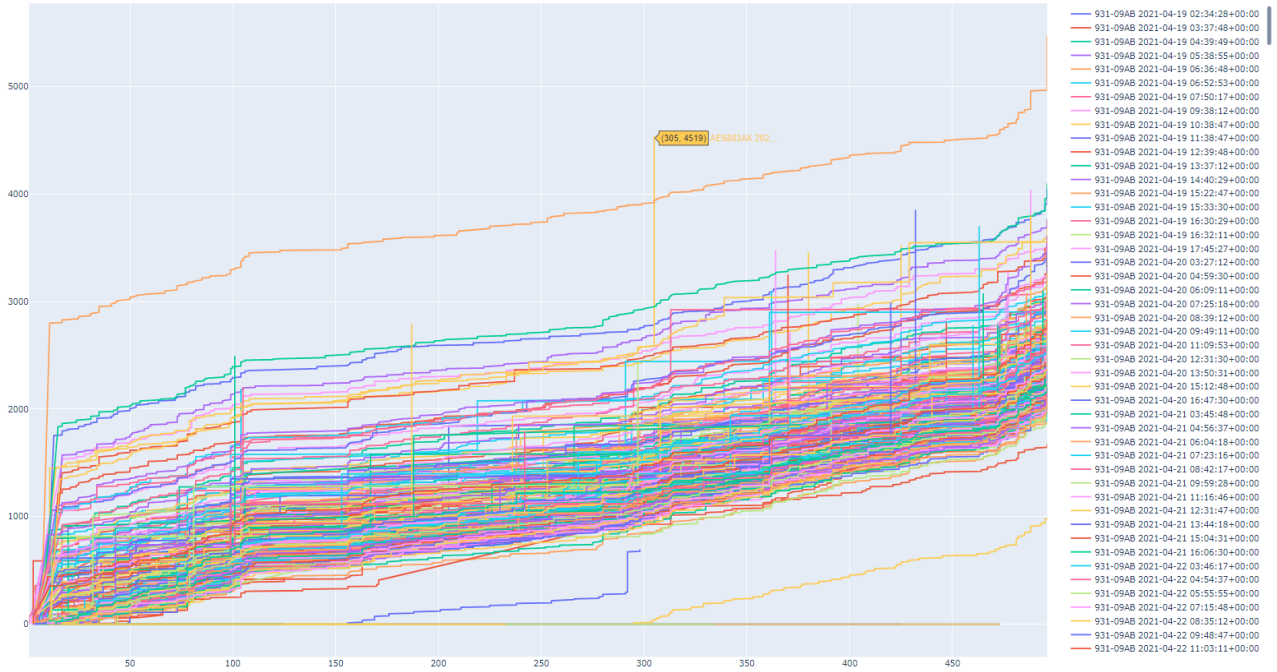

Рисунок 3.11 – Графік після застосування бібліотеки plotly

Далі, було вирішено відфільтрувати поїздки, які не починаються з початку та не закінчуються у кінці маршруту. Також було вирішено використовувати мінути на осі Y.

В результаті фільтрації отримано наступний графік.

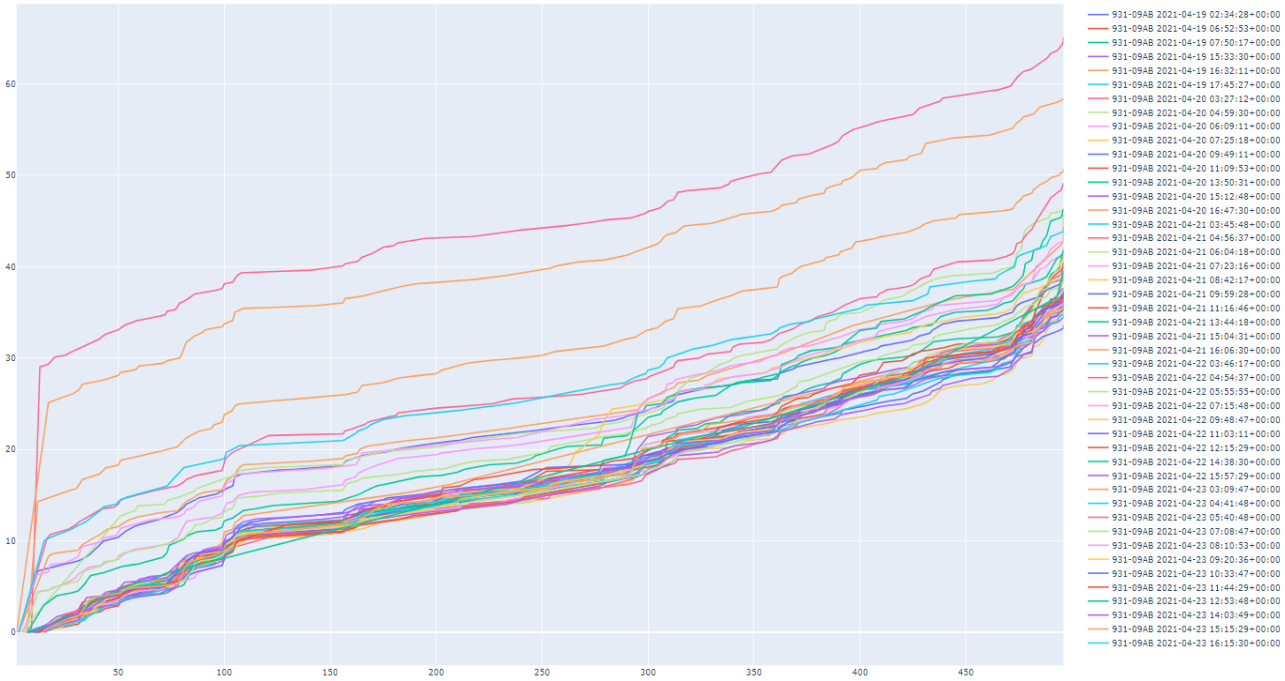

Рисунок 3.12 – Графік після застосування фільтрації на початок та кінець поїдки

На графіку бачимо, що попередні аномалії зникли.

Далі, було вирішено застосувати інтерполяцію даних, щоб для всіх координат idx на графіку у поїздці був розрахований час. Для цього було використано лінійну інтерполяцію вбудовану у pandas.

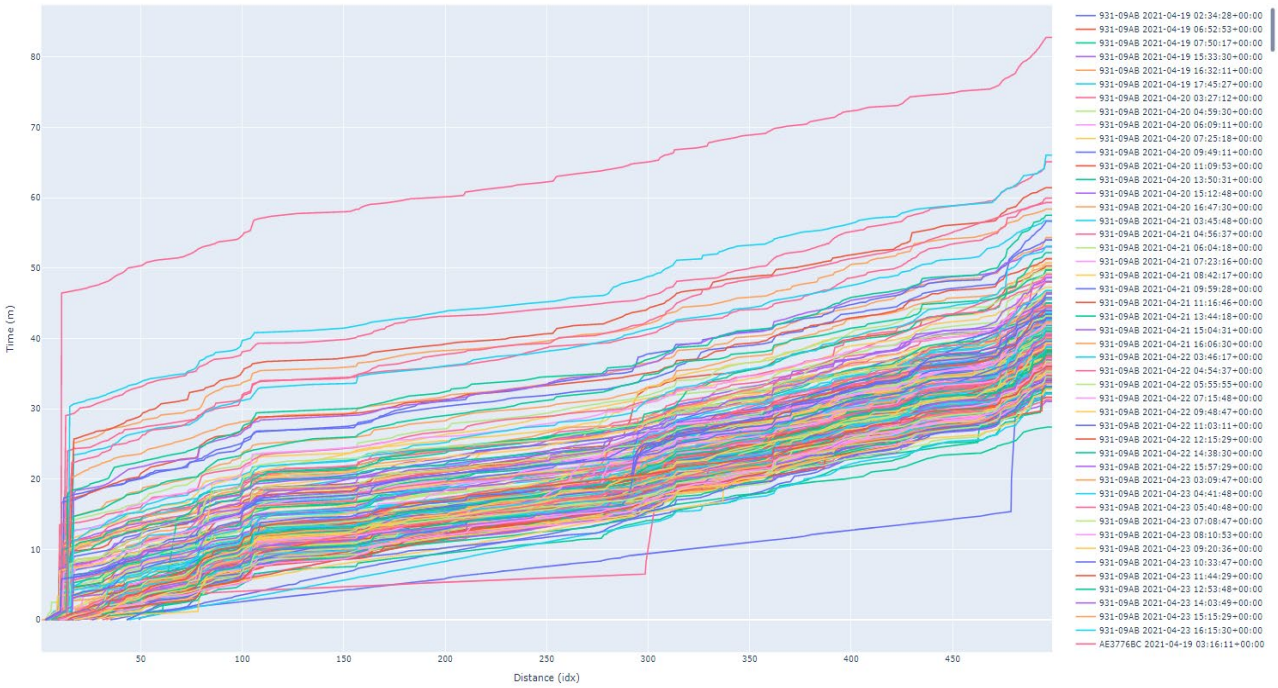

Рисунок 3.13 – Графік після інтерполяції

В результаті аналізу поля idx отримано правила розмежування поїздок.

3.3.3.4 Аналіз координат

Далі було вирішено зробити аналіз координат. Для візуалізації результатів, було використано бібліотеку ipyleaflet на пару з сервісом OpenStreetMap.

Для початку, було вирішено побудувати 119 маршрут на карті. Результат можна побачити нижче.

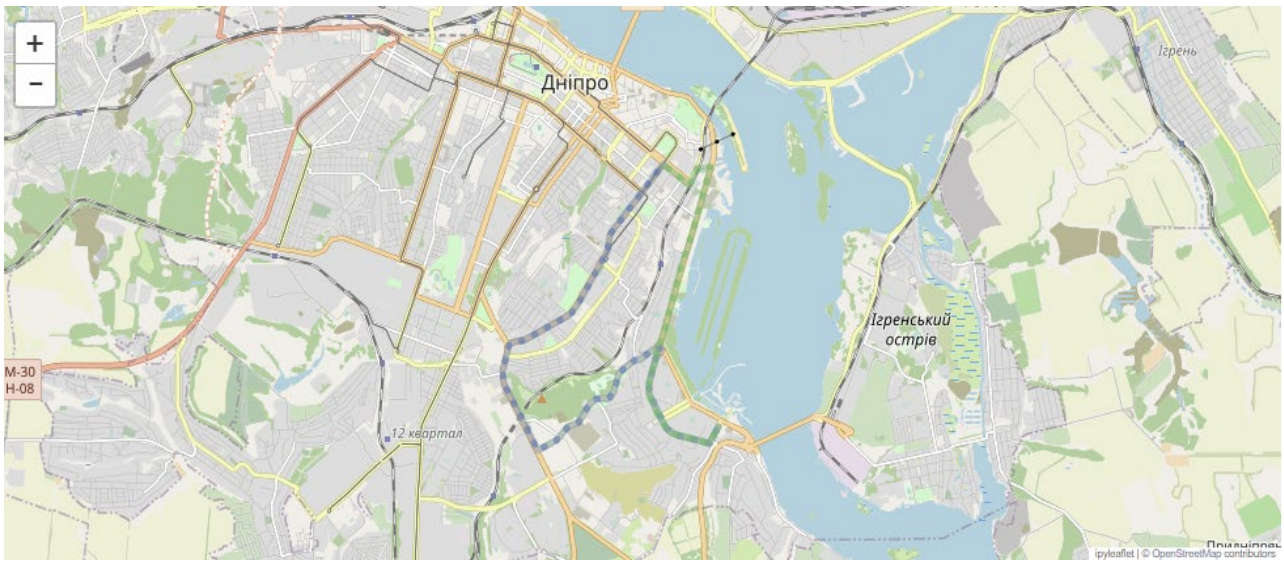

Рисунок 3.14 – Карта маршруту 119

На цій карті сірим кольором позначено прямий маршрут, а зеленим – зворотній.

Далі було вирішено побудувати всі точки поїздок прямим маршрутом для того, щоб досконально перевірити, чи збігаються вони з маршрутом взятим під EasyWay.

Рисунок 3.15 – Карта координат точок всіх обраних поїздок прямого маршруту

В результаті перевірки було підтверджено, що всі точки збігаються з маршрутом.

3.3.3.5 Аналіз швидкості транспортних засобів

Швидкість, яка передається від EasyWay не відповідає дійсності та має одне і те саме значення у всьому обсязі даних. Тому, було вирішено розрахувати свою швидкість.

Для цього було відфільтровано дані, де поле pos не змінилося. Для цього було використано наступний код, де current – таблиця вибраної машини.

```
current = current[(current.shift()['pos'] != current['pos']) |
current.index.isin([0])]
```

Лістинг 3.9 – Код фільтрації дублікатів позицій однієї машини

В результаті фільтрації отримано наступну таблицю.

	index	s	hc	wf	ac	d	vi	vn	ang	t	...	idx_app_nb	time	spd	pos	timestamp	posx	posy	t
90	90	OK	0	0	0	0	9786	AE5645AC	322	1720	...	4	1619835596	20	48.401120:35.027934	2021-05-01 02:20:09	48.411200	35.079342	
491	547	OK	0	0	0	0	9786	AE5645AC	324	1720	...	16	1619838011	20	48.460998:35.057844	2021-05-01 03:00:29	48.409986	35.078445	
494	550	OK	0	0	0	0	9786	AE5645AC	322	1720	...	18	1619838032	20	48.490889:35.007759	2021-05-01 03:00:47	48.408899	35.077590	
498	558	OK	0	0	0	0	9786	AE5645AC	285	1720	...	26	1619838053	20	48.420858:35.087637	2021-05-01 03:01:12	48.408582	35.076378	
501	564	OK	0	0	0	0	9786	AE5645AC	261	1720	...	32	1619838073	20	48.420880:35.027505	2021-05-01 03:01:28	48.408802	35.075052	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
234141	2667131	OK	0	0	0	0	12158	AE5645AC	130	1720	...	494	1622800231	20	48.475158:35.026182	2021-06-04 09:50:49	48.451587	35.061822	
234144	2667167	OK	0	0	0	0	12158	AE5645AC	131	1720	...	497	1622800248	20	48.415270:35.096308	2021-06-04 09:51:09	48.452701	35.063089	
234146	2667191	OK	0	0	0	0	12158	AE5645AC	131	1720	...	499	1622800268	20	48.405408:35.086467	2021-06-04 09:51:30	48.454080	35.064678	
234149	2667227	OK	0	0	0	0	12158	AE5645AC	113	1720	...	500	1622800294	20	48.435422:35.046501	2021-06-04 09:51:48	48.454223	35.065014	
234153	2667275	OK	0	0	0	0	12158	AE5645AC	63	1720	...	500	1622800309	20	48.415415:35.056515	2021-06-04 09:52:12	48.454151	35.065155	

32776 rows × 22 columns

Рисунок 3.16 – Таблиця результатів фільтрації дублікатів позицій

Далі, для розрахунку швидкості потрібно розрахувати відстань до попередньої точки для кожної точки поїздки.

Для цього було налаштовано локальний сервіс Open Route Service. Він дозволяє знайти найкоротший шлях між двома точками з прив’язкою до доріг. Для налаштування використано карту України OpenStreetMap.

Потім, було використано наступний код для розрахунку відстані за допомогою попередньо встановленого та запущеного серверу Open Route Service.

```
client = openrouteservice.Client(base_url='http://192.168.50.201:8080/ors')

res = client.directions(((previous['posy'], previous['posx']), (this['posy'],
this['posx'])))
distance = res['routes'][0]['summary']['distance'] if 'distance' in
res['routes'][0]['summary'] else -1
if len(res['routes']) > 1:
    print('multiple routes detected', len(res['routes']))
current_with_dist['distance'].iloc[i] = distance
```

Лістинг 3.10 – Код розрахунку відстані за допомогою локального серверу Open Route Service

В результаті розрахунків отримано наступну таблицю з новим полем distance, в якому записано відстань у метрах.

	index	s	hc	wf	ac	d	vi	vn	ang	t	...	idx_app_nb	time	spd	pos	timestamp	posx	posy	time_to_previous	idx_to_previous	distance
90	90	OK	0	0	0	0	9786	AE5645AC	322	1720	...	4	1619835596	20	48.401120:35.027934	2021-05-01 02:20:09	48.411200	35.079342	NaN	NaN	0.0
491	547	OK	0	0	0	0	9786	AE5645AC	324	1720	...	16	1619838011	20	48.460998:35.057844	2021-05-01 03:00:29	48.409986	35.078445	2415.0	12.0	150.3
494	550	OK	0	0	0	0	9786	AE5645AC	322	1720	...	18	1619838032	20	48.490889:35.007759	2021-05-01 03:00:47	48.408899	35.077590	21.0	2.0	136.3
498	558	OK	0	0	0	0	9786	AE5645AC	285	1720	...	26	1619838053	20	48.420858:35.087637	2021-05-01 03:01:12	48.408582	35.076378	21.0	8.0	105.2
501	564	OK	0	0	0	0	9786	AE5645AC	261	1720	...	32	1619838073	20	48.420880:35.027505	2021-05-01 03:01:28	48.408802	35.075052	20.0	6.0	100.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
234141	2667131	OK	0	0	0	0	12158	AE5645AC	130	1720	...	494	1622800231	20	48.475158:35.026182	2021-06-04 09:50:49	48.451587	35.061822	40.0	2.0	42.8
234144	2667167	OK	0	0	0	0	12158	AE5645AC	131	1720	...	497	1622800248	20	48.415270:35.096308	2021-06-04 09:51:09	48.452701	35.063089	17.0	3.0	155.2
234146	2667191	OK	0	0	0	0	12158	AE5645AC	131	1720	...	499	1622800268	20	48.405408:35.086467	2021-06-04 09:51:30	48.454080	35.064678	20.0	2.0	193.0
234149	2667227	OK	0	0	0	0	12158	AE5645AC	113	1720	...	500	1622800294	20	48.435422:35.046501	2021-06-04 09:51:48	48.454223	35.065014	26.0	1.0	39.3
234153	2667275	OK	0	0	0	0	12158	AE5645AC	63	1720	...	500	1622800309	20	48.415415:35.056515	2021-06-04 09:52:12	48.454151	35.065155	15.0	0.0	13.1

32776 rows × 22 columns

Рисунок 3.17 – Таблиця після розрахунків відстані

Далі розраховано швидкість за допомогою наступного коду.

```
dist_df.loc[dist_df['idx_to_previous'] >= 0, 'speed'] = dist_df['distance'] /
dist_df['time_to_previous']
```

Лістинг 3.11 – Код розрахунку швидкості у метрах на секунду

Також, було розраховано швидкість у кілометрах на годину за допомогою наступного коду.

```
dist_df['speed_kmh'] = dist_df['speed'] * 3.6
```

Лістинг 3.12 – Код розрахунку швидкості у кілометрах на годину

За допомогою наступного коду було виявлено аномалії у швидкостях.

```
dist_df_anomalies = dist_df.copy()
dist_df_anomalies.reset_index(inplace=True)
```

```
dist_df_anomalies[dist_df_anomalies['speed_kmh'] > 100]
```

Лістинг 3.13 – Код знаходження аномалій у швидкостях

В результаті отримаємо наступні записи з аномаліями.

	level_0	index	s	hc	wf	ac	d	vi	vn	ang	...	spd	pos	timestamp	posx	posy	time_to_previous	idx_to_previous	distance	speed	speed_kmh
118	1348	4707	OK	0	0	0	0	9786	AE5645AC	296	...	20	48.462295:35.066047	2021-05-01 04:27:30	48.422956	35.060476	20.0	6.0	631.6	31.580000	113.688000
439	3545	17802	OK	0	0	0	0	9786	AE5645AC	139	...	20	48.494008:35.085012	2021-05-01 08:12:29	48.440089	35.050128	9.0	5.0	304.7	33.855556	121.880000
556	4719	25592	OK	0	0	0	0	9786	AE5645AC	221	...	20	48.451368:35.086334	2021-05-01 10:13:11	48.413685	35.063348	21.0	3.0	1463.9	69.709524	250.954286
824	6910	40540	OK	0	0	0	0	9786	AE5645AC	296	...	20	48.422260:35.035970	2021-05-01 13:59:11	48.422602	35.059703	20.0	7.0	701.0	35.050000	126.180000
1369	11593	72228	OK	0	0	0	0	9786	AE5645AC	149	...	20	48.402316:35.056366	2021-05-02 06:38:29	48.423160	35.063665	19.0	6.0	547.4	28.810526	103.717895
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
31071	223500	2547825	OK	0	0	0	0	12158	AE5645AC	138	...	20	48.402316:35.056367	2021-06-02 17:58:29	48.423160	35.063675	12.0	4.0	478.1	39.841667	143.430000
31949	229337	2611646	OK	0	0	0	0	12158	AE5645AC	225	...	20	48.411261:35.076421	2021-06-03 14:07:48	48.412611	35.064217	20.0	1.0	1404.6	70.230000	252.828000
32034	229554	2614307	OK	0	0	0	0	12158	AE5645AC	247	...	20	48.405409:35.006527	2021-06-03 14:41:38	48.454090	35.065270	22.0	0.0	1218.2	55.372727	199.341818
32235	230933	2631673	OK	0	0	0	0	12158	AE5645AC	295	...	20	48.442296:35.026049	2021-06-04 02:26:46	48.422964	35.060492	15.0	10.0	639.9	42.660000	153.576000
32501	232388	2646022	OK	0	0	0	0	12158	AE5645AC	250	...	20	48.405409:35.006527	2021-06-04 06:19:37	48.454090	35.065270	43.0	0.0	1219.3	28.355814	102.080930

113 rows × 25 columns

Рисунок 3.18 – Таблиця записів аномалій швидкості

Далі було розглянуто одну з аномалій. Для цього побудовано на карті точку, в якій виявлено аномалію, попередню та наступну точки до неї. Результат можна побачити нижче.

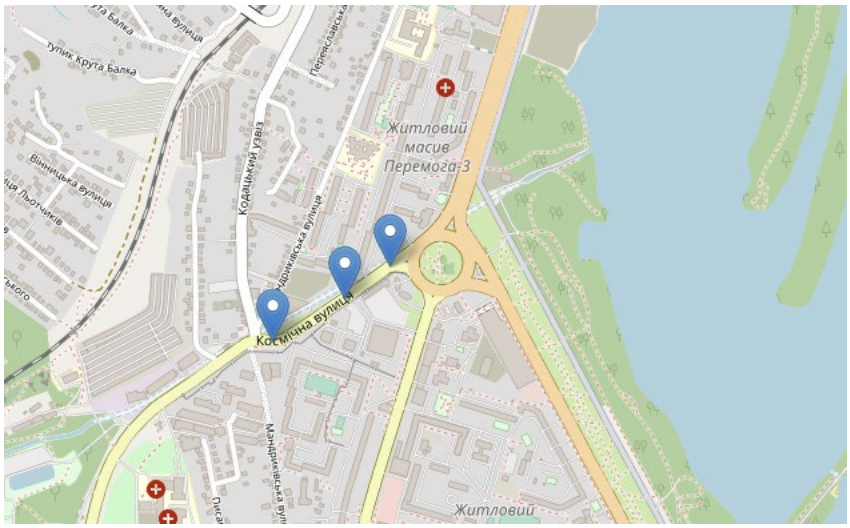

Рисунок 3.19 – Карта вибраної точки з аномалією, попередньої та наступної до неї точок

На карті вище маємо три точки. Справа – попередня, зліва – наступна та по центру – вибрана точка. В результаті аналізу аномалії виявлено, що аномалії трапляються, коли одна з точок знаходиться близько до зустрічного смуги.

В результаті аналізу аномалій було вирішено відфільтрувати поїздки, в яких є аномальні швидкості.

3.4 Розробка системи візуалізації даних

3.4.1 Вибір технологій

3.4.1.1 Вибір мови та платформи для розробки

Для розробки системи було обрано платформу Power BI.

У даній системі Power BI використовується для фільтрації та візуалізації відфільтрованих даних.

3.4.2 Етапи розробки

Процес розробки поділений на наступні етапи.

1. Загрузка даних з системи аналізу даних.
2. Розробка інтерфейсу візуалізації.

3.4.3 Процес розробки

3.4.3.1 Завантаження даних з системи аналізу даних

З системи аналізу дані можна завантажити наступними способами.

- Використати експорт у xlsx, або csv, та завантажити цей файл у Power BI.
- Використати SQL Server, як постачальника даних для Power BI.

3.4.3.2 Розробка інтерфейсу візуалізації

Інтерфейс візуалізації розроблена за допомогою платформи Power BI.

Для малювання карти обрано сервіс Mapbox. Для того, щоб відобразити на 3D карті середню швидкість транспортних засобів в межах одного значення idx, було розроблено свій tileset. Для того, щоб його створити, було розроблено наступний код.

```
geojson = {
    "type": "FeatureCollection",
    "features": []
}

for i, row in df.sort_values(by='idx').iterrows():
```

```
    geojson['features'].append({
      "type": "Feature",
      "properties": {
        "idx": int(row['idx'])
      },
      "geometry": {
        "coordinates": [
            [
                [row['idx_posy_median'] - 0.00003, row['idx_posx_median'] -
0.00003],
                [row['idx_posy_median'] - 0.00003, row['idx_posx_median'] +
0.00003],
                [row['idx_posy_median'] + 0.00003, row['idx_posx_median'] +
0.00003],
                [row['idx_posy_median'] + 0.00003, row['idx_posx_median'] -
0.00003],
                [row['idx_posy_median'] - 0.00003, row['idx_posx_median'] -
0.00003],
            ]
        ],
        "type": "Polygon"
      },
    })
```

Лістинг 3.14 – Код створення інформації для Mapbox tileset

Після цього, дані було записано до JSON-файлу, та завантажено до сервісу Mapbox. Далі, цей tileset можна використовувати при відображенні різних характеристик транспортних засобів з прив’язкою до карти.

Далі отримуємо наступну сторінку у Power BI.

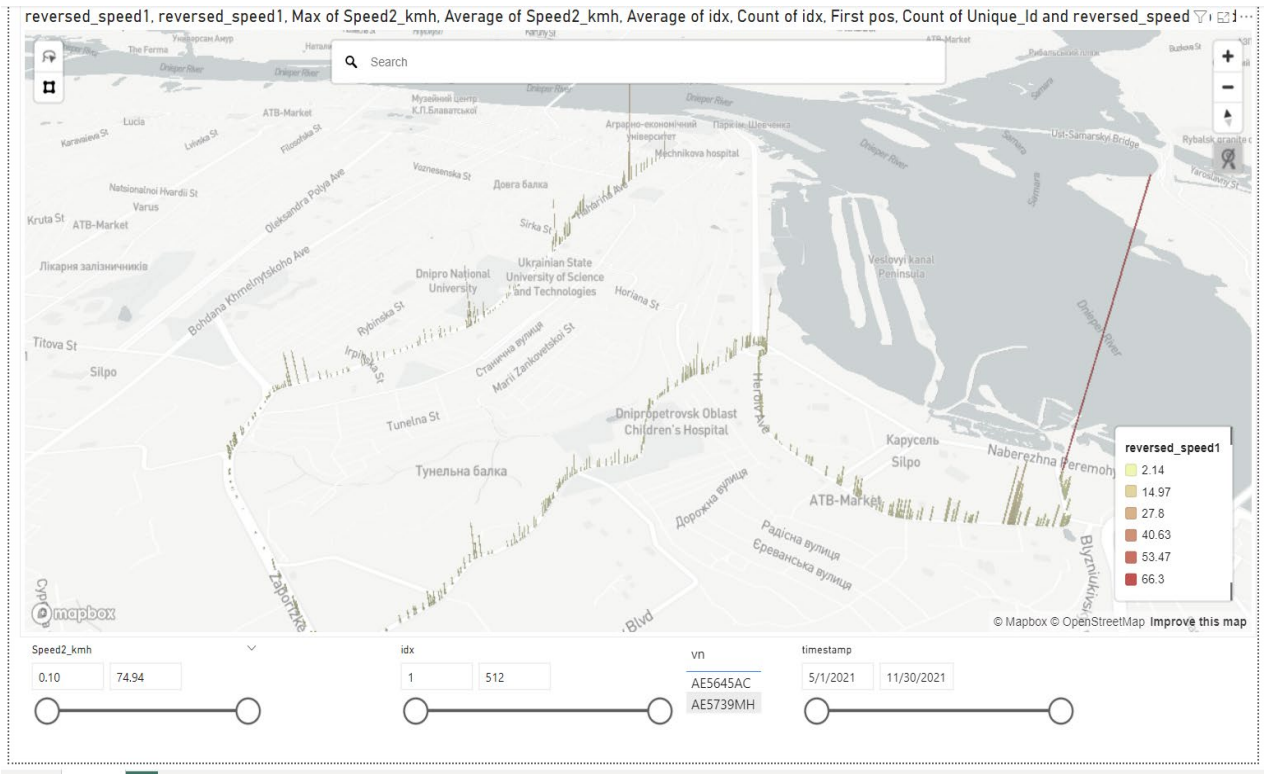

Рисунок 3.20 – Скріншот третьої сторінки у Power BI

На цій сторінці можна побачити елементи фільтрації вхідних даних для карти. А також, саму карту. На карті відображено значення $1/average(speed)$, тобто середня зворотна швидкість транспортних засобів на ділянці. Чим вище стовпчик, тим менша середня швидкість на цій ділянці.

Також, на іншій сторінці у Power BI було відображено середню швидкість на вбудованій у Power BI 2D карті. Приклад роботи сторінки можна побачити нижче.

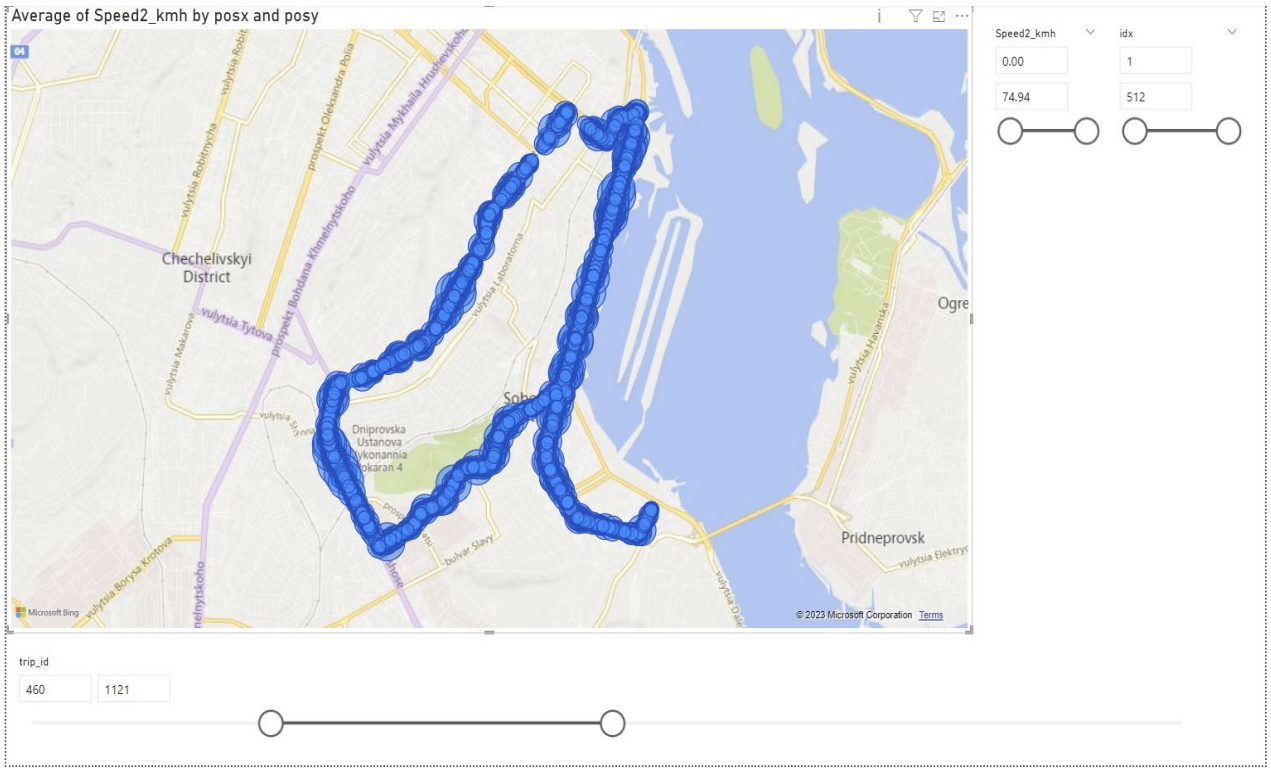

Рисунок 3.21 – Скріншот першої сторінки у Power BI

Також, розроблено сторінку, на якій зазначені кордони ділянок. Її можна побачити нижче.

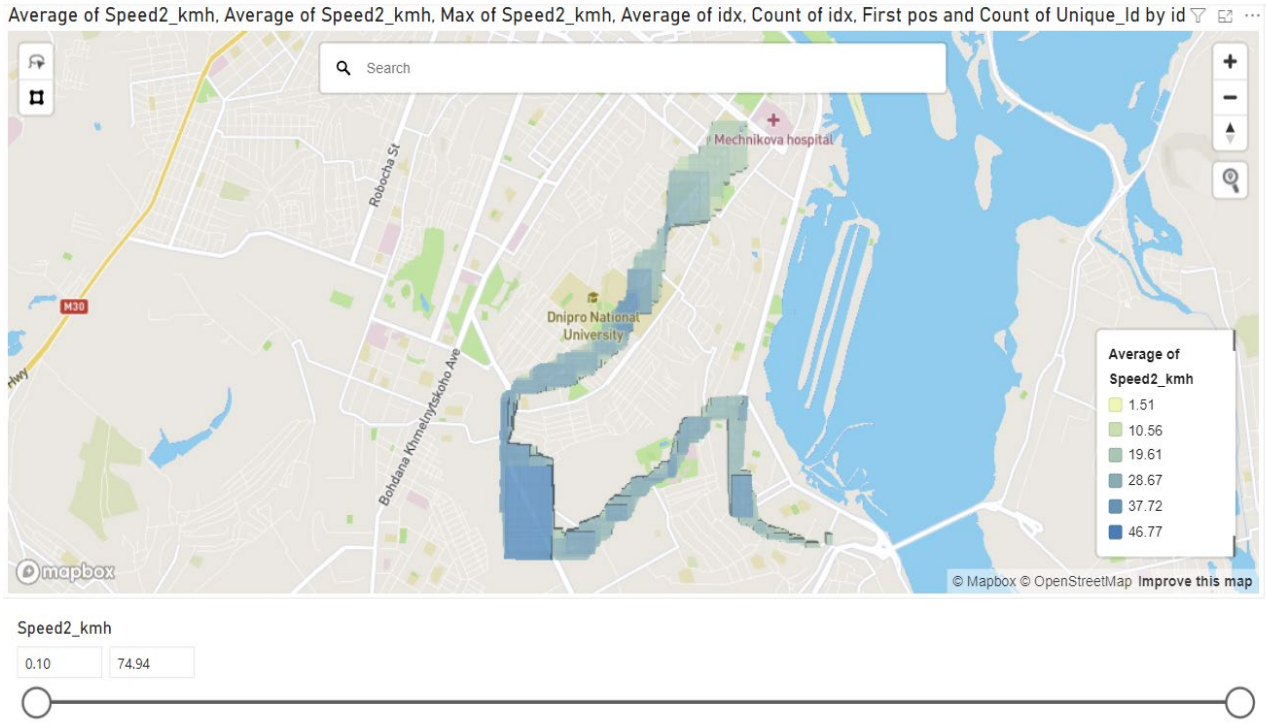

Рисунок 3.22 – Скріншот другої сторінки у Power BI

3.5 Розробка конструктивно-продукційної моделі

3.5.1 Спеціалізація конструктора

Розробимо спеціалізацію конструктора – конструктор руху засобів громадського транспорту.

$$C =< M, \Sigma, \Lambda > \quad {}_{s} \mapsto \; C_{TR} = \langle M_{TR}, \Sigma_{TR}, \Lambda_{TR} \rangle, \tag{3.1}$$

де M_{GF} – включає множину терміналів T (всіх можливих авто на карті), нетерміналів $N = \{A\}$, правил підстановки [26].

Позначимо авто z з атрибутами $\mathrm{v} \downarrow \mathrm{z}$ – швидкість авто, $\mathrm{pos} \downarrow \mathrm{z}$ – позиція авто.

Введемо операції над атрибутами:

- завантаження швидкості авто $\mathrm{v} \downarrow \mathrm{z}$ з бази даних $\circ\,(i, DB)$, де i – номер авто, DB – база даних SQL Server;
- завантаження позиції авто $\mathrm{pos} \downarrow \mathrm{z}$ з бази даних $*\,(i, DB)$;
- відображення авто на карті $+(\mathrm{MAP}, \mathrm{pos} \downarrow \mathrm{z}, \mathrm{v} \downarrow \mathrm{z})$.

3.5.2 Конкретизація конструктору

Інформаційне забезпечення конструювання включає вище приведені визначення, а також наступні положення:

- вводяться поняття «дійсне число», «координати»;
- мета конструювання – відображення швидкості та місця положення авто на карті;
- правила підстановки:

  $\{\langle\langle \mathrm{A} \to \mathrm{zA}\rangle, \langle \circ\,(i, DB), *\,(i, DB), +(\mathrm{MAP}, \mathrm{pos} \downarrow \mathrm{z}, \mathrm{v} \downarrow \mathrm{z})\rangle\rangle,$

  $\langle\langle \mathrm{A} \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle\}$
- обмеження – правило $\langle\langle \mathrm{A} \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle$ виконується, якщо перше правило не прийнятне;

- початкові умови: t – час початку поїздки, vn – реєстраційний номер авто;
- умови завершення – виконання правила $\langle\langle \mathrm{A} \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle$.

3.5.3 Інтерпретація конструктору

Алгоритми внутрішнього виконавця:

- виконання операції композиції алгоритмів $A_1^0|_{A_i,A_j}^{A_i \cdot A_j}$ ($A|_X^Y$ – алгоритм над даними з вхідної множини X зі значеннями з множини Y, A^0 – утворюючий алгоритм), , – послідовне виконання алгоритму після алгоритму;
- алгоритм завантаження швидкості авто з бази даних $A_3|_{i,DB}^{\mathrm{v}\downarrow\mathrm{z}}$;
- алгоритм завантаження позиції авто з бази даних $A_4\ |_{i,DB}^{\mathrm{pos}\downarrow\mathrm{z}}$;
- алгоритм відображення позиції та швидкості авто на карті $A_5\ |_{MAP,\mathrm{pos}\downarrow\mathrm{z},\,\mathrm{v}\downarrow\mathrm{z}}^{MAP}$;
-

Поставимо у відповідність кожній операції з Σ_{TR} алгоритм внутрішнього виконавця.

$$\{(A_1^0|_{A_i,A_j}^{A_i \cdot A_j} \hookleftarrow \cdot), (A_2^0|_{Z_1,Z_2,A_i}^{:A_i} \hookleftarrow :), (A_3|_{i,DB}^{\mathrm{v}\downarrow\mathrm{z}} \hookleftarrow \circ); (A_6|_{l_h,l_q,l_i}^{l_j} \hookleftarrow \Rightarrow);$$

$$(A_7|_{l_i,\Psi}^{l_j} \hookleftarrow | \Rightarrow); (A_8|_{l_i,\Psi}^{l_j} \hookleftarrow || \Rightarrow)\}$$

3.5.4 Реалізація конструктору

Приклад результатів реалізації конструктору наведено на рис. 3.20

4 ТЕСТУВАННЯ ТА ВІДЛАГОДЖЕННЯ

4.1 Тестування системи аналізу даних

Тестування чорної скриньки та білої скриньки – це два основних методи тестування програмного забезпечення, які мають власні особливості.

У методі тестування чорної скриньки фокус ставиться на функціональність системи без урахування внутрішньої структури або робочого механізму. Тестери використовують вхідні дані та перевіряють вихідні, щоб переконатися, що система працює відповідно до вимог.

До переваг тестування чорної скриньки відносять наступні пункти.

- Не потребує знання програмування та внутрішньої структури системи.
- Ефективне для великих систем коду.
- Можна швидко виявити невідповідності в системі.

До недоліків тестування чорної скриньки відносять наступні пункти.

- Можливість пропустити важливі внутрішні помилки.
- Тестування може бути поверховим та не всі аспекти системи перевіряються.

У методі тестування білої скриньки перевіряється внутрішня структура системи. Тестери мають доступ до вихідного коду та використовують його для розробки тестових сценаріїв та випробувань. Це може включати перевірку окремих функцій, модулів або процедур коду.

До переваг тестування білої скриньки відносять наступні пункти.

- Виявляє помилки внутрішньої логіки та структури коду.
- Забезпечує глибоке тестування, включаючи шляхи коду, що рідко використовуються.
- Можливість раннього виявлення та виправлення помилок.

До недоліків тестування білої скриньки відносять наступні пункти.

- Знання програмування та внутрішньої структури коду є обов'язковим.

- Часоємкий та ресурсоємкий процес.
- Може бути складно провести для великих систем коду.

Через те, що розроблена система має велику кількість елементів вирішено обрати метод тестування білої скриньки і протестувати критичні функції з перетворення даних.

4.1.1 Тестування функції видалення дублікатів

У наступному лістингу зазначено код функції.

```
def filter_out_duplicates(current):
    """Filter out duplicates in DataFrame based on field 'pos'"""
    return current[(current.shift()['pos'] != current['pos']) |
current.index.isin([0])]
```

Лістинг 4.1 – Код функції видалення дублікатів

Функція з видалення дублікатів в pandas DataFrame використовує методи shift() і isin() для видалення дублікатів. Ми проводимо тестування цієї функції білою скринькою, що означає, що ми вивчаємо код зсередини та перевіряємо, що він виконується належним чином.

4.1.1.1 Вхідні дані для тестових випадків

Вхідні дані для тестових випадків представлено у вигляді коду у наступному лістингу.

```
# Тестовий випадок 1
data1 = {
    'pos': ['51.5:0.12', '51.5:0.12', '48.85:2.35', '48.85:2.35', '40.71:-74.01',
'52.37:4.89', '52.37:4.89']
}

# Тестовий випадок 2
data2 = {
    'pos': ['51.5:0.12', '51.5:0.12', '51.5:0.12', '51.5:0.12', '51.5:0.12',
'51.5:0.12', '51.5:0.12']
}

# Тестовий випадок 3
data3 = {
    'pos': ['51.5:0.12', '48.85:2.35', '40.71:-74.01', '52.37:4.89', '35.68:139.7',
'55.75:37.6', '39.91:116.4']
}
```

```
# Тестовий випадок 4
data4 = {
    'pos': ['39.91:116.4', '55.75:37.6', '35.68:139.7', '35.68:139.7',
'52.37:4.89', '40.71:-74.01', '48.85:2.35']
}

# Тестовий випадок 5
data5 = {
    'pos': []
}
```

Лістинг 4.2 – Вхідні дані для тестових випадків

4.1.1.2 Тестові випадки

4.1.1.2.1 Тестовий випадок 1

Виконуємо функцію використовуючи data1, очікуваний результат DataFrame без дублікатів наведено у наступному лістингу.

```
          pos
0   51.5:0.12
2  48.85:2.35
4 40.71:-74.01
5  52.37:4.89
```

Лістинг 4.3 – Очікуваний результат для тестового випадку 1

Оскільки фактичний результат співпадає з очікуваним, тест пройдено успішно.

4.1.1.2.2 Тестовий випадок 2

Виконуємо функцію використовуючи data2, очікуваний результат – DataFrame без дублікатів наведено у наступному лістингу.

```
         pos
0  51.5:0.12
```

Лістинг 4.4 – Очікуваний результат для тестового випадку 2

Фактичний результат наведено у лістингу нижче.

```
         pos
0  51.5:0.12
```

Лістинг 4.5 – Фактичний результат для тестового випадку 2

Оскільки фактичний результат співпадає з очікуваним, тест пройдено успішно.

4.1.1.2.3 Тестовий випадок 3

Виконуємо функцію використовуючи data3, очікуваний результат – DataFrame без дублікатів наведено у наступному лістингу.

```
          pos
0   51.5:0.12
1  48.85:2.35
2 40.71:-74.01
3  52.37:4.89
4  35.68:139.7
5   55.75:37.6
6  39.91:116.4
```

Лістинг 4.6 – Очікуваний результат для тестового випадку 3

Фактичний результат наведено у лістингу нижче.

```
          pos
0   51.5:0.12
1  48.85:2.35
2 40.71:-74.01
3  52.37:4.89
4  35.68:139.7
5   55.75:37.6
6  39.91:116.4
```

Лістинг 4.7 – Фактичний результат для тестового випадку 3

Оскільки фактичний результат співпадає з очікуваним, тест пройдено успішно.

4.1.1.2.4 Тестовий випадок 4

Виконуємо функцію використовуючи data4, очікуваний результат – оригінальний DataFrame, оскільки в ньому не було дублікатів, наведено у наступному лістингу.

```
          pos
0  39.91:116.4
1   55.75:37.6
2  35.68:139.7
4  52.37:4.89
5 40.71:-74.01
6  48.85:2.35
```

Лістинг 4.8 – Очікуваний результат для тестового випадку 4

Фактичний результат наведено у лістингу нижче.

```
          pos
```

```
0  39.91:116.4
1   55.75:37.6
2  35.68:139.7
4  52.37:4.89
5 40.71:-74.01
6  48.85:2.35
```

Лістинг 4.9 – Фактичний результат для тестового випадку 4

Оскільки фактичний результат співпадає з очікуваним, тест пройдено успішно.

4.1.1.2.5 Тестовий випадок 5

Виконуємо функцію використовуючи data5, очікуваний результат – DataFrame порожній DataFrame, оскільки вхідні дані були порожніми.

Фактичним результат є порожній DataFrame.

Оскільки фактичний результат співпадає з очікуваним, тест пройдено успішно.

4.1.1.3 Висновок

Тестування білою скринькою допомагає нам перевірити правильність внутрішньої логіки функції видалення дублікатів. Так як всі тести пройдені успішно, можна вважати, що функція працює коректно.

4.1.2 Тестування функції розрахунку швидкості

У наступному лістингу зазначено код функції.

```
def calculate_speed_based_on_distance(dist_df):
    """Calculate fields 'speed' and 'speed_kmh'"""
    dist_df.loc[dist_df['idx_to_previous'] >= 0, 'speed'] = dist_df['distance'] /
dist_df['time_to_previous']
    dist_df['speed_kmh'] = dist_df['speed'] * 3.6
```

Лістинг 4.10 – Код функції розрахунку швидкості

Проводимо тестування білою скринькою функції calculate_speed_based_on_distance(), яка розраховує швидкість на основі відстані та часу, збережених в DataFrame. Вона також конвертує швидкість в км/год.

Функція calculate_speed_based_on_distance() вимагає на вхід pandas DataFrame з такими стовпцями: idx_to_previous, distance та time_to_previous.

4.1.2.1 Тестові випадки

4.1.2.1.1 Тестовий випадок 1

Перевіряємо розрахунки при нормальних умовах. Виконуємо функцію calculate_speed_based_on_distance() на df1, де df1 містить дані, які наведено у лістингу нижче.

```
df1 = pd.DataFrame({
    'idx_to_previous': [1, 2, 3],
    'distance': [1000, 2000, 3000],  # distance in meters
    'time_to_previous': [100, 200, 300]  # time in seconds
})
```

Лістинг 4.11 – Вхідні дані для тестового випадку 1

Вихідними даними є DataFrame з доданими стовпцями speed та speed_kmh. Його наведено у лістингу нижче.

```
   idx_to_previous  distance  time_to_previous  speed  speed_kmh
0                1      1000               100   10.0       36.0
1                2      2000               200   10.0       36.0
2                3      3000               300   10.0       36.0
```

Лістинг 4.12 – Вихідні дані для тестового випадку 1

4.1.2.1.2 Тестовий випадок 2

Перевіряємо обробку нульового часу. Виконуємо функцію calculate_speed_based_on_distance() на df2, де df2 містить дані, які наведено у лістингу нижче.

```
df2 = pd.DataFrame({
    'idx_to_previous': [1, 2, 3],
    'distance': [1000, 2000, 3000],
    'time_to_previous': [0, 200, 300]  # time in seconds
})
```

Лістинг 4.13 – Вхідні дані для тестового випадку 2

Очікуваним результатом є DataFrame з доданими стовпцями speed та speed_kmh. Значення швидкості для першого рядка має бути NaN, оскільки час є нульовим. Очікуваний DataFrame наведено у лістингу нижче.

```
   idx_to_previous  distance  time_to_previous  speed  speed_kmh
0                1      1000                 0    NaN        NaN
1                2      2000               200   10.0       36.0
2                3      3000               300   10.0       36.0
```

Лістинг 4.14 – Очікуваний результат для тестового випадку 2

Фактичний результат наведено у лістингу нижче.

```
   idx_to_previous  distance  time_to_previous  speed  speed_kmh
0                1      1000                 0    NaN        NaN
1                2      2000               200   10.0       36.0
2                3      3000               300   10.0       36.0
```

Лістинг 4.15 – Фактичний результат для тестового випадку 2

4.1.2.2 Висновок

Тестування білою скринькою допомагає нам перевірити правильність внутрішньої логіки функції видалення дублікатів. Так як всі тести пройдені успішно, можна вважати, що функція працює коректно.

ВИСНОВКИ

В ході виконання дипломної роботи розроблено систему для накопичення, аналізу та візуалізації даних руху громадського автотранспорту. Даний проект включає широкий спектр функціональності.

Проект розбито на три підсистеми. А саме, система накопичення, система аналізу та система візуалізації даних.

Для розробки системи накопичення обрано мову програмування Javascript разом з платформою Node.js. Для розробки системи аналізу обрано мову програмування Python разом з Jupyter блокнотами та Microsoft Excel. Для розробки системи візуалізації даних обрано платформу Power BI.

Систему було успішно протестовано методом білої скриньки.

Систему накопичення даних було викладено на сервер хмарного провайдера Hetzner, де було накопичено два роки даних для дослідження. Потім, було досліджено ці дані, зроблено потрібні перетворення, та візуалізовано їх.

Для перевірки даних на коректність зроблено статистичний аналіз за допомогою t-test Стьюдента та тесту Фішера. Окрім того, побудовано довірчі інтервали.

В результаті розробки системи та проведення дослідження, отримано вибірку даних для успішного відновлення конструктору та розроблено базову кострукційно-продукційну модель.

Далі планується доопрацювати та ускладнити конструкційно-продукційну модель.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) [Електронний ресурс]: навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. – Київ : КПІ ім. Ігоря Сікорського, 2020. – 192с. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/33651/1/PIS_KL.pdf

2. Gupta Prateek, Practical Data Science with Jupyter, Second Edition / Prateek Gupta. – New Delhi : BPB Publications, 2021. – 654 pages, ISBN: 978-93-89898-064.

3. "Python Scripts vs. Jupyter Notebooks: Pros and Cons", LearnPython, accessed 8 March, 2023, https://learnpython.com/blog/python-scripts-vs-jupyter-notebooks/

4. Shynkarenko, V.I., Ilman, V.M. Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part I. Generalized Formal Constructive-Synthesizing Structure. [Virtual Resource] / Viktor Shynkarenko // Cybernetics and Systems Analysis. – Springer. – 2014. – Vol. 50, No 5, p. 665 – 662. – Access Mode : DOI : 10.1007/s10559-014-9655-z, https://link.springer.com/article/10.1007/s10559-014-9655-z, Part II. Refining Transformations. – Springer. – 2014. – Vol. 50, No 6, 2014. p. 829 – 841, ISSN: 1060-0396 (Print) 1573-8337 (Online), DOI: 10.1007/s10559-014-9674-9, https://link.springer.com/article/10.1007/s10559-014-9674-9

5. Shynkarenko, V.I., Vasetska, T.M. Modeling the Adaptation of Compression Algorithms by Means of Constructive-Synthesizing Structures. [Virtual Resource] / Viktor Shynkarenko // Cybernetics and Systems Analysis. – Springer. – 2015. – Vol. 51, No 6. p. 849-861, DOI: 10.1007/s10559-015-9778-x, https://link.springer.com/article/10.1007/s10559-015-9778-x

6. Шинкаренко, В. И., Васецкая, Т. Н. Моделирование процесса ранжирования альтернатив методом анализа иерархий средствами

конструктивно-продукционных структур. [Текст] / В.И. Шинкаренко // Математические машины и системы. № 1. – 2016 – С. 39-47, ISSN 1028-9763

7. Шинкаренко, В. И., Забула, Г. В. Конструктивная модель адаптации структур данных в оперативной памяти. ЧАСТЬ I. Конструирование текстов программ. [Текст] / В.И. Шинкаренко // Наука и прогресс транспорта. № 1 (61), 2016. С. 109-121.; ЧАСТЬ II. Конструкторы сценариев и процессов адаптации. № 2 (62), 2016. С. 88-97, ISSN 2307-6666

8. Шинкаренко, В.И., Куропятник, Е.С. Конструктивно-продукционная модель графового представления текста. [Текст] / В.И. Шинкаренко // Проблемы программирования. № 2-3, 2016. С. 63-72, ISSN 1727- 4907, http://dspace.nbuv.gov.ua/bitstream/handle/123456789/126391/07-Shinkarenko.pdf?sequence=1

9. "2022 Gartner Magic Quadrant for Analytics and Business Intelligence Platforms", Gartner, Inc., accessed May 26, 2023, https://info.microsoft.com/ww-landing-2022-gartner-mq-report-on-bi-and-analytics-platforms.html.

10. "Analytics and Business Intelligence Platforms Reviews 2023 | Gartner Peer Insights", Gartner, Inc., accessed May 26, 2023, https://www.gartner.com/reviews/market/analytics-business-intelligence-platforms .

11. "Power BI documentation", Microsoft Inc., https://learn.microsoft.com/en-au/power-bi/ .

12. "HTML Living Standard | 9.2 Server-sent events", World Wide Web Consortium and WHATWG, https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events .

13. Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Dissertation [Virtual Resource] / Roy Thomas Fielding // University of California, Irvine. – 2020. Access Mode : URL : https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. – Date of Access: 12 April 2023.

14. “SQL vs. NoSQL: full comparison of features, differences, and more”, TestGorilla, https://www.testgorilla.com/blog/sql-vs-nosql/ .

15. “Why and when to use API keys | Security of API keys ”, Google Inc., https://cloud.google.com/endpoints/docs/openapi/when-why-api-key#security_of_api_keys .

16. “MongoDB Manual | Sharding”, MongoDB, Inc., https://www.mongodb.com/docs/manual/sharding/.

17. “SQL Server technical documentation”, Microsoft Inc., https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16.

18. “Introduction to Hypothesis Testing”, Statology.org, accessed May 5, 2023, https://www.statology.org/hypothesis-testing/.

19. “An Explanation of P-Values and Statistical Significance”, Statology.org, accessed May 5, 2023, https://www.statology.org/p-values-statistical-significance/.

20. “An Introduction to t Tests | Definitions, Formula and Examples”, Scribbr, https://www.scribbr.com/statistics/t-test/ .

21. “F-Test for Equal Variances Calculator”, Statology.org, accessed May 6, 2023, https://www.statology.org/f-test-for-equal-variances-calculator/.

22. Urdan, Timothy C., Statistics in Plain English, 5th Edition / Timothy C. Urdan. – Published by Routledge, 2022. – 322 pages. ISBN 9780367342838 .

23. “Understanding The Fundamentals Of Confidence Interval In Statistics”, Simplilearn, accessed May 6, 2023, https://www.simplilearn.com/tutorials/data-analytics-tutorial/confidence-intervals-in-statistics .

24. “Use the Analysis ToolPak to perform complex data analysis”, Microsoft Inc., accessed May 6, 2023, https://support.microsoft.com/en-us/office/use-the-analysis-toolpak-to-perform-complex-data-analysis-6c67ccf0-f4a9-487c-8dec-bdb5a2cefab6 .

25. “Descriptive statistics using Excel and Stata”, Princeton University, https://www.princeton.edu/~otorres/Excel/excelstata.htm.

26. Шинкаренко, В. И., Литвиненко, К. В., Чигирь, Р. Конструктивное соответствие мультисимвольных и линейных геометрических фракталов. [Текст] / В.И. Шинкаренко // International Journal «Information Technologies & Knowledge». Vol. 13. No 1. – 2019 – P. 76–99, ISSN: 1313-0455, http://eadnurt.diit.edu.ua/jspui/handle/123456789/11818.

ДОДАТОК А

Технічне завдання

ЗАТВЕРДЖЕНО
1116130.01303-01-ЛЗ

ВІДНОВЛЕННЯ КОНСТРУКТОРУ РУХУ МІСЬКОГО ГРОМАДСЬКОГО АВТОТРАНСПОРТУ

Технічне завдання

1116130.01303-01
Листів 12

2023

ЗМІСТ

1 ВВЕДЕННЯ

ВІДНОВЛЕННЯ КОНСТРУКТОРУ РУХУ МІСЬКОГО ГРОМАДСЬКОГО АВТОТРАНСПОРТУ. Проблема нерегулярності руху громадського транспорту стає виразною під час подорожей, коли людям важко визначити час очікування наступного маршрутного засобу. Це особливо помітно під час дорожніх заторів, які можуть статися внаслідок ремонту доріг, святкувань та інших подій.

Сервіс EasyWay використовується для прогнозування часу прибуття громадського транспорту. Основні переваги цього сервісу включають можливість відслідковування руху громадського автотранспорту у реальному часі з точністю до хвилини, а також здатність відслідковувати рух тролейбусів та трамваїв.

Ціль даної роботи полягає у збиранні даних для їх подальшого аналізу за допомогою сучасних інструментів та методик.

Практичне застосування: ця робота допоможе людям більш точно планувати свої поїздки на громадському транспорті.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 07.12.22 №1209ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи – “Відновлення конструктору руху міського громадського автотранспорту”. Керівник – Шинкаренко В.І.

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

3.1 Функціональне призначення

Система призначена для виконання ряду ключових функцій, що включають накопичення реальних даних для подальшого дослідження, вибір даних зі сховища, гнучке дослідження даних за допомогою вбудованих та самостійно реалізованих методів, а також візуалізацію результатів дослідження.

3.2 Експлуатаційне призначення

Система призначена для допомоги дослідникам та аналітикам у зручному зборі, обробці та аналізі даних. Вона сприяє ефективній візуалізації результатів дослідження, що забезпечує краще розуміння вибірки даних та дозволяє виявляти залежності та тренди. Під час експлуатації, система має гарантувати гнучкість та зручність користування, надаючи користувачу змогу обрати потрібні дані та використовувати різноманітні методи для їх дослідження.

4 ВИМОГИ ДО ПРОГРАМИ АБО ПРОГРАМНОГО ПРОДУКТУ

4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- Зберігання даних: система має бути спроможна збирати та зберігати дані для подальшого аналізу.
- Вибір даних: система повинна надавати можливість вибирати конкретні дані з загального сховища даних.
- Адаптивний аналіз даних: система має забезпечувати зручність дослідження даних, використовуючи вбудовані або користувацькі методи.
- Графічне представлення даних: система повинна включати можливість візуального відображення результатів аналізу даних.

4.2 Вимоги до надійності

Цілодобова доступність: Система має бути доступна 24 години на добу, 7 днів на тиждень. Це означає, що мають бути вжиті всі необхідні заходи для забезпечення її безперебійної роботи, уникнення простоїв або відновлення роботи якомога швидше у випадку виникнення проблем.

Стабільність: Система повинна працювати стабільно і без помилок протягом тривалого періоду часу. Вона повинна бути здатна обробляти великі об'єми даних і користувацькі запити без зниження продуктивності або якості обслуговування.

Відмовостійкість: В системі має бути реалізована механіка відновлення після збоїв. Це включає в себе резервне копіювання даних, резервні системи, автоматизовані процедури відновлення та регулярне тестування цих процедур.

Безпека: Цілодобова робота системи не повинна створювати додаткових ризиків для безпеки даних. Всі дані та транзакції повинні бути захищені від несанкціонованого доступу, витоку інформації та інших загроз безпеки.

4.3 Умови експлуатації

Операційна система Windows 10 або 11, Ubuntu 22.04.

Система потребує стабільного Інтернет-з’єднання для передачі даних.

Також система потребує встановленого Python версії не менше 3.11.

4.4 Вимоги до складу і параметрів технічних засобів

Система потребує як мінімум чотирьохядерний процесор з тактовою частотою 2.2 ГГц, 4 ГБ оперативної пам’яті, 10 ГБ пам’яті жорсткого диску.

4.5 Вимоги до інформаційної і програмної сумісності

Операційна система Windows/Linux.

Мови програмування: JavaScript, Python, SQL.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації повинні входити:

- специфікація;
- текст програми.

Програмна документація повинна відповідати вимогам ДСТУ [1].

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки програмного продукту приведені в табл. 1.

Таблиця А.1 – Стадії та етапи розробки програмного продукту

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вступ	12.09.22 – 26.10.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	27.10.22 – 04.03.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	05.03.23 – 31.04.23	
4	Постановка задачі, технічне завдання	01.05.23 – 07.05.23	30%
5	Узгодження та затвердження ТЗ	08.05.23 – 15.05.23	
6	Розробка інструментальних засобів дослідження	16.05.23 – 21.05.23	
7	Виконання досліджень	29.05.23 – 01.06.23	60%
8	Оформлення тез доповідей	21.03.2023 – 24.03.2023	
9	Оформлення статті у фаховий журнал	02.06.23 – 06.06.23	
10	Оформлення пояснювальної записки	07.06.23 – 11.06.23	
11	Розробка демонстраційних матеріалів	12.06.23 – 18.06.23	100%
12	Подання	22.06.23	

	кваліфікаційної роботи до кафедри		
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	27.06.23	

7 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль здійснює керівник розробки Шинкаренко В.І

Прийом здійснює уповноважена комісія.

8 БІБЛІОГРАФІЧНИЙ СПИСОК

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

ДОДАТОК Б

Специфікація

ЗАТВЕРДЖЕНО
1116130.01303-01-ЛЗ

ВІДНОВЛЕННЯ КОНСТРУКТОРУ РУХУ МІСЬКОГО ГРОМАДСЬКОГО АВТОТРАНСПОРТУ

Специфікації

1116130.01303-01

Листів 2

2023

1 Специфікації

Таблиця 1.1 – Специфікації.

Позначення	Найменування	Примітка
	Документація	
1116130.01303-01-ЛЗ	Лист затвердження	
1116130.01303-01-ЛЗ	Лист затвердження	
1116130.01303-01 12 01	Текст програми	
1116130.01303-01 13 01-ЛЗ	Лист затвердження	
1116130.01303-01 13 01	Опис програми	
1116130.01303-01 І3 01-ЛЗ	Лист затвердження	
1116130.01303-01 І3 01	Керівництво користувача	

ДОДАТОК В

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
_____Анатолій РАДКЕВИЧ

ВІДНОВЛЕННЯ КОНСТРУКТОРУ РУХУ МІСЬКОГО ГРОМАДСЬКОГО АВТОТРАНСПОРТУ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01303-01 -ЛЗ

Завідувач кафедри КІТ
Вадим ГОРЯЧКІН

Керівник розробки
Віктор ШИНКАРЕНКО

Виконавець
Михайло ВЄТЛУЖСЬКИХ

Нормоконтролер
Світлана ВОЛКОВА

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ

ВІДНОВЛЕННЯ КОНСТРУКТОРУ РУХУ МІСЬКОГО ГРОМАДСЬКОГО АВТОТРАНСПОРТУ

Специфікації
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01303-01 -ЛЗ

Завідувач кафедри КІТ
Вадим ГОРЯЧКІН

Керівник розробки
Віктор ШИНКАРЕНКО

Виконавець
Михайло ВЄТЛУЖСЬКИХ

Нормоконтролер
Світлана ВОЛКОВА

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ

ВІДНОВЛЕННЯ КОНСТРУКТОРУ РУХУ МІСЬКОГО ГРОМАДСЬКОГО АВТОТРАНСПОРТУ

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01303-01 -ЛЗ

Завідувач кафедри КІТ
Вадим ГОРЯЧКІН

Керівник розробки
Віктор ШИНКАРЕНКО

Виконавець
Михайло ВЄТЛУЖСЬКИХ

Нормоконтролер
Світлана ВОЛКОВА

2023

ДОДАТОК Г

Текст програми

ЗАТВЕРДЖЕНО
1116130.01303-01-ЛЗ

ВІДНОВЛЕННЯ КОНСТРУКТОРУ РУХУ МІСЬКОГО ГРОМАДСЬКОГО АВТОТРАНСПОРТУ

Текст програми

1116130.01303-01

Листів 74

2023

convert_parquet_to_excel.ipynb:

```
#%%

import pandas

#%%

df = pandas.read_parquet('../data/test_chart_1_plotly_filtered_fixed_interpolated_osm_with_new_data.parquet')

df

#%%

df.to_excel('test_chart_1_plotly_filtered_fixed_interpolated_osm_with_new_data.xlsx')
```

create_idx_geojson.ipynb

```
#%%

import pandas

#%%

df = pandas.read_csv('C:\\Users\\misha\\OneDrive\\Документы\\Book2.csv')

df

#%%

geojson = {
   "type": "FeatureCollection",
   "features": []
}
for i, row in df.sort_values(by='idx').iterrows():
   # print(row)
   geojson['features'].append({
     "type": "Feature",
     "properties": {
       "idx": int(row['idx'])
     },
     "geometry": {
       "coordinates": [
         [
             [row['idx_posy_median'] - 0.00003, row['idx_posx_median'] - 0.00003],
             [row['idx_posy_median'] - 0.00003, row['idx_posx_median'] + 0.00003],
             [row['idx_posy_median'] + 0.00003, row['idx_posx_median'] + 0.00003],
             [row['idx_posy_median'] + 0.00003, row['idx_posx_median'] - 0.00003],
             [row['idx_posy_median'] - 0.00003, row['idx_posx_median'] - 0.00003],
             # [row['idx_posy_min'], row['idx_posx_min']],
             # [row['idx_posy_min'], row['idx_posx_max']],
             # [row['idx_posy_max'], row['idx_posx_max']],
             # [row['idx_posy_max'], row['idx_posx_min']],
             # [row['idx_posy_min'], row['idx_posx_min']],
         ]
       ],
       "type": "Polygon"
     },
   })
# geojson

#%%

import json

with open('../data/geojson_idx.json', 'w') as f:
   json.dump(geojson, f)
```

distance_and_idx.ipynb

```
#%%

import pandas
import matplotlib.pyplot as plt
import numpy

#%%

df = pandas.read_csv('../data/export_2021-03-31T15_56_29.611Z.csv')

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%%

df

#%%

vns = df['vn'].unique()[:1]

#%%

plt.figure(figsize=(40,20))
for vn in vns:
   current = df.loc[df['vn'] == vn].copy()
   current['time_from_stop'] = numpy.zeros(len(current))
   last_stop_time = current['timestamp'].iloc[0]
   j = 0
   for i in range(1, len(current)):
       if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:
           plt.plot(current['time_from_stop'].iloc[j:i], current['idx'].iloc[j:i], 'o-', label=vn + ' ' + str(last_stop_time))
           j = i
```

```
            last_stop_time = current['timestamp'].iloc[i]
            # print(last_stop_time)
        current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()
    # df1 = pandas.concat([df1, current[['time', vn]]])
plt.legend()
plt.savefig('../data/distance_and_idx.png')
plt.show()
```

download_data.py.ipynb:

```
#%%
mongo_host = '168.119.191.39:27018'
mongo_user = 'root'
mongo_password = '******'
#%%
from urllib.parse import quote_plus
import pymongo
uri = "mongodb://%s:%s@%s" % (
    quote_plus(mongo_user), quote_plus(mongo_password),
mongo_host)
client = pymongo.MongoClient(uri)
#%%
client.list_database_names()
#%%
db = client.get_database('admin')
db.list_collection_names()
#%%
db.get_collection('positions_history').find({
    'timestamp': {'$and': [
        {'$gte': '2021-05-01'},
        {'$lte': '2021-12-01'},
    ]}}).sort('timestamp', pymongo.ASCENDING).limit(5)
#%%
print(uri)
#%%
import datetime
datetime.datetime.fromtimestamp(1640792779)
```

geopy_test.ipynb:

```
#%%
from geopy.distance import distance, geodesic, great_circle,
ELLIPSOIDS
#%%
print('distance', distance((48.423098, 35.060568),
(48.42368892920818, 35.06196274759801)).meters)
#%%
print('great_circle', great_circle((48.423098, 35.060568),
(48.42368892920818, 35.06196274759801)).meters)
#%%
print('geodesic', geodesic((48.423098, 35.060568),
(48.42368892920818, 35.06196274759801)).meters)
#%%
for ellipsoid in ELLIPSOIDS:
    print('geodesic', ellipsoid, geodesic((48.423098,
35.060568), (48.42368892920818, 35.06196274759801),
ellipsoid=ellipsoid).meters)
```

ipyleaflet_test.ipynb:

```
#%%
from ipyleaflet import Map, Polyline
line = Polyline(
    locations=[
        (48.423098, 35.060568),
        (48.42368892920818, 35.06196274759801)
    ],
    color="green" ,
    fill=False,

)
m = Map(center = (48.423098, 35.060568), zoom = 17)
m.add_layer(line)
m
#%%
```

test_chart.ipynb

```
#%%
import pandas
import matplotlib.pyplot as plt
import numpy
pandas.options.mode.chained_assignment = None  #
default='warn'
#%%
df = pandas.read_csv('../data/export_2021-05-
12T15_51_16.094Z.csv')
df = df.loc[df['d'] == 0]
df['timestamp'] = pandas.to_datetime(df['timestamp'])
```

```
#%%

df

#%%

vns = df['vn'].unique()

#%%

df['vn'].unique()

#%%

plt.figure(figsize=(40,20))

for vn in vns:

    print(vn)

    current = df.loc[df['vn'] == vn].copy()

    current.iloc[1:].loc[current['idx'].shift(-1) == current['idx'],
'idx'] = numpy.nan

    current['idx'] = current['idx'].interpolate(method='linear')

    current['time_from_stop'] = numpy.zeros(len(current))

    start = 0

    if current['idx'].iloc[start] >= 50:

        for i in range(1, len(current)):

            if current['idx'].iloc[i] < current['idx'].iloc[i - 1] and
current['idx'].iloc[i] < 50:

                start = i

                break

    last_stop_time = current['timestamp'].iloc[start]

    j = 0

    cut = False

    for i in range(start + 1, len(current)):

        # if not cut and current['idx'].iloc[i] -
current['idx'].iloc[j] > 50:

        #     print(i)

        #     cut = True

        #     last_stop_time = current['timestamp'].iloc[i]

        #     j = i

        #     continue

        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:

            plt.plot(current['idx'].iloc[j:i],
current['time_from_stop'].iloc[j:i], 'o-', label=vn + ' ' +
str(last_stop_time))

            if current['idx'].iloc[i] >= 50:

                for k in range(i + 1, len(current)):

                    if current['idx'].iloc[k] < current['idx'].iloc[k - 1]
and current['idx'].iloc[k] < 50:

                        i = k

                        break

            j = i

            cut = False

            last_stop_time = current['timestamp'].iloc[i]

            # print(last_stop_time)

        current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()

    # df1 = pandas.concat([df1, current[['time', vn]]])

# plt.legend()

plt.savefig('../data/test_chart.png')

plt.show()

test_chart_1.ipynb

#%%

import pandas

import matplotlib.pyplot as plt

import numpy

pandas.options.mode.chained_assignment = None  #
default='warn'

#%%

df = pandas.read_csv('../data/export_2021-05-
12T15_51_16.094Z.csv')

df = df.loc[df['d'] == 0]

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%%

df

#%%

vns = df['vn'].unique()

#%%

df['vn'].unique()

#%%

plt.figure(figsize=(40,20))

for vn in vns:

    print(vn)

    current = df.loc[df['vn'] == vn].copy()

    current.iloc[1:].loc[current['idx'].shift(-1) == current['idx'],
'idx'] = numpy.nan

    current['idx'] = current['idx'].interpolate(method='linear')

    current['time_from_stop'] = numpy.zeros(len(current))

    start = 0

    if current['idx'].iloc[start] >= 50:

        for i in range(1, len(current)):
```

```
        if current['idx'].iloc[i] < current['idx'].iloc[i - 1] and
current['idx'].iloc[i] < 50:
            start = i
            break
    last_stop_time = current['timestamp'].iloc[start]
    j = 0
    cut = False
    for i in range(start + 1, len(current)):
        # if not cut and current['idx'].iloc[i] -
current['idx'].iloc[j] > 50:
        #     print(i)
        #     cut = True
        #     last_stop_time = current['timestamp'].iloc[i]
        #     j = i
        #     continue
        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:
            if current['time_from_stop'].iloc[i - 1] < 20000:
                plt.plot(current['idx'].iloc[j:i],
current['time_from_stop'].iloc[j:i], 'o-', label=vn + ' ' +
str(last_stop_time))
            if current['idx'].iloc[i] >= 50:
                for k in range(i + 1, len(current)):
                    if current['idx'].iloc[k] < current['idx'].iloc[k - 1]
and current['idx'].iloc[k] < 50:
                        i = k
                        break
            j = i
            cut = False
            last_stop_time = current['timestamp'].iloc[i]
            # print(last_stop_time)
        current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()
    # df1 = pandas.concat([df1, current[['time', vn]]])
# plt.legend()
plt.savefig('../data/test_chart1.png')
plt.show()
```

test_chart_1_and_dispersion.ipynb:

```
#%%
import pandas
import numpy
import plotly.graph_objects as go
pandas.options.mode.chained_assignment = None  #
default='warn'
#%%
df = pandas.read_csv('../data/export_2021-05-
12T15_51_16.094Z.csv')
df = df.loc[df['d'] == 0]
df['timestamp'] = pandas.to_datetime(df['timestamp'])
#%%
df
#%%
vns = df['vn'].unique()[:1]
#%%
df['vn'].unique()
#%%
fig = go.Figure(layout=go.Layout(
    autosize=False,
    width=1920,
    height=1080,))
for vn in vns:
    print(vn)
    current = df.loc[df['vn'] == vn].copy()
    current.iloc[1:].loc[current['idx'].shift(-1) == current['idx'],
'idx'] = numpy.nan
    current['idx'] = current['idx'].interpolate(method='linear')
    current['time_from_stop'] = numpy.zeros(len(current))
    start = 0
    if current['idx'].iloc[start] >= 50:
        for i in range(1, len(current)):
            if current['idx'].iloc[i] < current['idx'].iloc[i - 1] and
current['idx'].iloc[i] < 50:
                start = i
                break
    last_stop_time = current['timestamp'].iloc[start]
    j = 0
    cut = False
    for i in range(start + 1, len(current)):
        # if not cut and current['idx'].iloc[i] -
current['idx'].iloc[j] > 50:
        #     print(i)
        #     cut = True
        #     last_stop_time = current['timestamp'].iloc[i]
        #     j = i
```

```
        #     continue

        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:

            if current['time_from_stop'].iloc[i - 1] < 20000 and
current['idx'].iloc[i - 1] >= 490:

                fig.add_trace(go.Scatter(x=current['idx'].iloc[j:i],
y=current['time_from_stop'].iloc[j:i],

                    mode='lines', # lines+markers

                    name=vn + ' ' + str(last_stop_time)))

            if current['idx'].iloc[i] >= 50:

                for k in range(i + 1, len(current)):

                    if current['idx'].iloc[k] < current['idx'].iloc[k - 1]
and current['idx'].iloc[k] < 50:

                        i = k

                        break

            j = i

            cut = False

            last_stop_time = current['timestamp'].iloc[i]

            # print(last_stop_time)

        current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()

    # df1 = pandas.concat([df1, current[['time', vn]]])

# plt.legend()

fig.show()

#%%

test_chart_1_export_to_excel.ipynb:

#%%

import pandas

import numpy

import plotly.graph_objects as go

pandas.options.mode.chained_assignment = None  #
default='warn'

#%%

df = pandas.read_csv('../data/export_2021-05-
12T15_51_16.094Z.csv')

df = df.loc[df['d'] == 0]

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%%

df

#%%

vns = df['vn'].unique()

#%%

df['vn'].unique()

#%%

trips = []

for vn in vns:

    print(vn)

    current = df.loc[df['vn'] == vn].copy()

    current.iloc[1:].loc[current['idx'].shift(-1) == current['idx'],
'idx'] = numpy.nan

    current['idx'] = current['idx'].interpolate(method='linear')

    current['time_from_stop'] = numpy.zeros(len(current))

    start = 0

    if current['idx'].iloc[start] >= 50:

        for i in range(1, len(current)):

            if current['idx'].iloc[i] < current['idx'].iloc[i - 1] and
current['idx'].iloc[i] < 50:

                start = i

                break

    last_stop_time = current['timestamp'].iloc[start]

    j = 0

    cut = False

    for i in range(start + 1, len(current)):

        # if not cut and current['idx'].iloc[i] -
current['idx'].iloc[j] > 50:

        #     print(i)

        #     cut = True

        #     last_stop_time = current['timestamp'].iloc[i]

        #     j = i

        #     continue

        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:

            if current['time_from_stop'].iloc[i - 1] < 20000:

                current_trip = pandas.DataFrame()

                current_trip['x'] = current['idx'].iloc[j:i]

                current_trip[vn + ' ' + str(last_stop_time)] =
current['time_from_stop'].iloc[j:i]

                trips.append(current_trip)

            if current['idx'].iloc[i] >= 50:

                for k in range(i + 1, len(current)):

                    if current['idx'].iloc[k] < current['idx'].iloc[k - 1]
and current['idx'].iloc[k] < 50:

                        i = k

                        break

            j = i

            cut = False
```

```
            last_stop_time = current['timestamp'].iloc[i]

            # print(last_stop_time)

        current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()

    # df1 = pandas.concat([df1, current[['time', vn]]])

# plt.legend()

#%%

for trip in trips:

    trip.set_index('x', inplace=True)

#%%

output = pandas.concat(trips)

#%%

output.sort_index(inplace=True)

output = output.interpolate(method='linear')

#%%

output

#%%

output.to_csv('excel_export_3.csv')
```

test_chart_1_plotly.ipynb:

```
#%%

import pandas

import numpy

import plotly.graph_objects as go

pandas.options.mode.chained_assignment = None  #
default='warn'

#%%

df = pandas.read_csv('../data/export_2021-05-
12T15_51_16.094Z.csv')

df = df.loc[df['d'] == 0]

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%%

df

#%%

vns = df['vn'].unique()

#%%

df['vn'].unique()

#%%

fig = go.Figure(layout=go.Layout(

    autosize=False,

    width=1920,

    height=1080,))

for vn in vns:

    print(vn)

    current = df.loc[df['vn'] == vn].copy()

    current.iloc[1:].loc[current['idx'].shift(-1) == current['idx'],
'idx'] = numpy.nan

    current['idx'] = current['idx'].interpolate(method='linear')

    current['time_from_stop'] = numpy.zeros(len(current))

    start = 0

    if current['idx'].iloc[start] >= 50:

        for i in range(1, len(current)):

            if current['idx'].iloc[i] < current['idx'].iloc[i - 1] and
current['idx'].iloc[i] < 50:

                start = i

                break

    last_stop_time = current['timestamp'].iloc[start]

    j = 0

    cut = False

    for i in range(start + 1, len(current)):

        # if not cut and current['idx'].iloc[i] -
current['idx'].iloc[j] > 50:

        #     print(i)

        #     cut = True

        #     last_stop_time = current['timestamp'].iloc[i]

        #     j = i

        #     continue

        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:

            if current['time_from_stop'].iloc[i - 1] < 20000:

                fig.add_trace(go.Scatter(x=current['idx'].iloc[j:i],
y=current['time_from_stop'].iloc[j:i],

                    mode='lines', # lines+markers

                    name=vn + ' ' + str(last_stop_time)))

            if current['idx'].iloc[i] >= 50:

                for k in range(i + 1, len(current)):

                    if current['idx'].iloc[k] < current['idx'].iloc[k - 1]
and current['idx'].iloc[k] < 50:

                        i = k

                        break

            j = i

            cut = False

            last_stop_time = current['timestamp'].iloc[i]

            # print(last_stop_time)
```

```
        current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()

    # df1 = pandas.concat([df1, current[['time', vn]]])

# plt.legend()

fig.show()

#%%

test_chart_1_plotly_filtered.ipynb:

#%%

import pandas

import numpy

import plotly.graph_objects as go

import chart_studio.plotly as py

pandas.options.mode.chained_assignment = None  #
default='warn'

#%%

df = pandas.read_csv('../data/export_2021-05-
12T15_51_16.094Z.csv')

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%%

df

#%%

vns = df['vn'].unique()

#%%

df['vn'].unique()

#%%

fig = go.Figure(layout=go.Layout(

    autosize=False,

    width=1920,

    height=1080,))

points = list(range(1, 500))

output_df = pandas.DataFrame(columns=points)

for vn in vns[:1]:

    print(vn)

    current = df.loc[df['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]

    current['time_from_stop'] = numpy.zeros(len(current))

    start = 0

    # if current['idx'].iloc[start] >= 50:

    #     for i in range(1, len(current)):

    #         if current['idx'].iloc[i] < current['idx'].iloc[i - 1] and
current['idx'].iloc[i] < 50:

    #             start = i

    #             break

    last_stop_time = current['timestamp'].iloc[start]

    j = 0

    for i in range(start + 1, len(current)):

        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:

            if current['idx'].iloc[j] < 50 and current['idx'].iloc[i -
1] >= 490:

                interpolated_idx = current[['time_from_stop',
'idx']].iloc[j:i].copy()

                interpolated_idx.loc[interpolated_idx['idx'].shift(1)
== interpolated_idx['idx']] = numpy.nan

                interpolated_idx['idx'].interpolate(method='linear',
inplace=True)

                interpolated_idx.reset_index(drop=True,
inplace=True)

                for p in points:

                    if len(interpolated_idx.loc[interpolated_idx['idx']
== p]) == 0:

                        interpolated_idx.append({'idx': p,
'time_from_stop': numpy.nan}, ignore_index=True)

                interpolated_idx.sort_values(by=['idx'],
inplace=True)


interpolated_idx['time_from_stop'].interpolate(method='linea
r', inplace=True)

                # print(interpolated_idx)

                fig.add_trace(go.Scatter(x=interpolated_idx['idx'],
y=interpolated_idx['time_from_stop'],

                    mode='lines', # lines+markers

                    name=vn + ' ' + str(last_stop_time)))

            # if current['idx'].iloc[i] >= 50:

            #     for k in range(i + 1, len(current)):

            #         if current['idx'].iloc[k] < current['idx'].iloc[k -
1] and current['idx'].iloc[k] < 50:

            #             i = k

            #             break

            j = i

            last_stop_time = current['timestamp'].iloc[i]

            # print(last_stop_time)

        current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()
/ 60

    # df1 = pandas.concat([df1, current[['time', vn]]])
```

```
# plt.legend()

fig.show()

#%%

test_chart_1_plotly_filtered_fixed_interpolated.ipynb:

#%%

import pandas

import numpy

import plotly.graph_objects as go

import chart_studio.plotly as py

pandas.options.mode.chained_assignment = None  # default='warn'

#%%

df = pandas.read_csv('../data/export_2021-05-12T15_51_16.094Z.csv')

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%% md

### Loaded data frame sample

#%%

df

#%%

vns = df['vn'].unique()

#%% md

### Unique serial numbers loaded

#%%

df['vn'].unique()

#%% md

### Interpolated chart of all trips

#%%

fig = go.Figure(layout=go.Layout(
    title="Trips",
    xaxis_title="Distance (idx)",
    yaxis_title="Time (m)",
    autosize=False,
    width=1920,
    height=1080,))

points = list(range(1, 500))

output_dfs = []

for vn in vns[:]:
    print(vn)
    current = df.loc[df['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]
    current['time_from_stop'] = numpy.zeros(len(current))
    start = 0
    last_stop_time = current['timestamp'].iloc[start]
    j = 0
    for i in range(start + 1, len(current)):
        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:
            if current['idx'].iloc[j] < 50 and current['idx'].iloc[i - 1] >= 490:
                interpolated_idx = current[['time_from_stop', 'idx']].iloc[j:i].copy()

interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan

interpolated_idx.sort_values(by=['time_from_stop'], inplace=True)
                interpolated_idx.set_index('time_from_stop', inplace=True)
                interpolated_idx['idx'].interpolate(method='linear', inplace=True)

interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan
                interpolated_idx.dropna(inplace=True)
                interpolated_idx.reset_index(inplace=True)
                for p in points:
                    if len(interpolated_idx.loc[interpolated_idx['idx'] == p]) == 0:
                        interpolated_idx = interpolated_idx.append({'idx': p, 'time_from_stop': numpy.nan}, ignore_index=True)
                interpolated_idx.sort_values(by=['idx'], inplace=True)
                interpolated_idx.set_index('idx', inplace=True)

interpolated_idx['time_from_stop'].interpolate(method='linear', inplace=True)
                fig.add_trace(go.Scatter(x=interpolated_idx.index, y=interpolated_idx['time_from_stop'],
                    mode='lines', # lines+markers
                    name=vn + ' ' + str(last_stop_time)))
                pivot_points = interpolated_idx.loc[interpolated_idx.index.isin(points)]
                pivot_points.index = pivot_points.index.astype(int)
                pivot_points[vn + ' ' + str(last_stop_time)] = pivot_points['time_from_stop']
```

```
            del pivot_points['time_from_stop']
            output_dfs.append(pivot_points)
            # if len(output_dfs) == 2:
            #     break
        j = i
        last_stop_time = current['timestamp'].iloc[i]
    current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()
/ 60
# plt.legend()
fig.show()
#%%
output_df = pandas.concat(output_dfs, axis=1, join='inner')
#%% md
### Pivot points of trips
#%%
import openpyxl
output_df.to_excel('../data/output_df.xlsx')
output_df.transpose()
#%% md
### Filter trips with not enough data at the start
#%%
# output_df1 = output_df.loc[output_df.iloc[20] !=
numpy.nan]
output_df1 = output_df.dropna(thresh=len(output_df) - 20,
axis=1)
output_df1.to_excel('../data/output_df1.xlsx')
output_df1
#%% md
### Normalize time
#%%
output_df2 = output_df1.diff()
output_df2.to_excel('../data/output_df2.xlsx')
output_df2
#%% md
### Trim data from front
#%%
output_df3 = output_df2.dropna()
output_df3.to_excel('../data/output_df3.xlsx')
output_df3
#%% md
### Descriptive data
#%%
output_df3_statistics = output_df3.transpose().describe()
output_df3_statistics
#%% md
### Descriptive data charts
#%%
fig = go.Figure(layout=go.Layout(
    title="Std of time difference",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=output_df3_statistics.columns,
y=output_df3_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))
#%% md
### Analyze data
#%%
output_df3.iloc[105].describe()
#%%
output_df3.iloc[105].idxmax()
#%%
output_df4 = output_df3.copy()
del output_df4['AE0548AA 2021-04-23 15:42:13+00:00']
output_df4_statistics = output_df4.transpose().describe()
fig = go.Figure(layout=go.Layout(
    title="Std of time difference without AE0548AA 2021-04-
23 15:42:13+00:00",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=output_df4_statistics.columns,
y=output_df4_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))
#%% md
```

```
### Analyze anomalies

#### Find the difference between current time and mean

#%%

output_df5 = output_df3.copy()

for i in range(len(output_df5)):

    row_mean_value = output_df5.iloc[i].mean()

    # print('mean:', row_mean_value)

    # print('before:' , output_df5.iloc[i])

    output_df5.iloc[i] -= row_mean_value

    # print('after:' , output_df5.iloc[i])

output_df5

#%%

output_df5.to_excel('../data/output_df5.xlsx');

#%% md

#### Group by car vn

#%%

output_df6 = output_df5.abs().groupby(lambda x: x.split('
')[0], axis=1).sum()

output_df6

#%%

output_df6.to_excel('../data/output_df6.xlsx')

#%% md

#### Find the sum of differences

#%%

output_df7 = output_df6.sum()

output_df7

#%%

output_df7.to_excel('../data/output_df7.xlsx')

#%% md

### Student's coefficient

#%%

# import stats

#

# student = lambda p, k: stats.t.ppf(1 - (1 - p) / 2, k)

test_chart_1_plotly_filtered_fixed_interpolated_geopy.ipynb:

#%%

import pandas

import numpy

import plotly.graph_objects as go

import chart_studio.plotly as py

pandas.options.mode.chained_assignment = None  #
default='warn'

#%%

df = pandas.read_csv('../data/export_2021-05-
12T15_51_16.094Z.csv')

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%% md

### Loaded data frame sample

#%%

df

#%% md

### Convert pos to latitude and longitude

#%%

# var b = c + "", a = b.split(".");

# if (1 < a.length) {

#     c = a[0];

#     a = a[1];

#     var e = a.length;

#     if (3 < e) {

#         b = "" + a[0];

#         for (var d = 2; d < e; d++) b += a[d];

#         b += a[1];

#         b = c + "." + b

#     }

#     c = +b

# }

def convertToLatLng(c):

    # print(c)

    b=c+""

    a=b.split(".")

    if 1 < len(a):

        c=a[0]

        a=a[1]

        e=len(a)

        if 3 < e:

            b=""+a[0]

            for d in range(2, e):

                b+=a[d]

            b+=a[1]

            b= c + "." + b
```

```
        c = b
    return c
positions = pandas.DataFrame()
positions['pos'] = numpy.zeros(len(df))
positions['idx'] = numpy.zeros(len(df))
end = len(df)
previous_progress = -1
for i in range(0, len(df)):
    progress = i / end * 100
    if progress - int(progress) < 0.01 and progress - previous_progress > 0.1:
        previous_progress = progress
        print(f"{progress:.2f}")
    # before = df['pos'].iloc[i]
    positions['pos'].iloc[i] = ':'.join([convertToLatLng(x) for x in df['pos'].iloc[i].split(':')])
    # print(before, df['pos'].iloc[i])
positions['posx'] = numpy.zeros(len(df))
positions['posy'] = numpy.zeros(len(df))
for i in range(0, len(df)):
    progress = i / end * 100
    if progress - int(progress) < 0.0001:
        print(progress)
    positions['idx'].iloc[i] = df['idx'].iloc[i]
    positions['posx'].iloc[i] = positions['pos'].iloc[i].split(':')[0]
    positions['posy'].iloc[i] = positions['pos'].iloc[i].split(':')[1]
positions
#%%
df1 = df.copy()
df1['posx'] = positions['posx']
df1['posy'] = positions['posy']
df1
#%%
vns = df['vn'].unique()
#%% md
### Unique serial numbers loaded
#%%
df['vn'].unique()
#%% md
### Interpolated chart of all trips
#%%
fig = go.Figure(layout=go.Layout(
    title="Trips",
    xaxis_title="Distance (idx)",
    yaxis_title="Time (m)",
    autosize=False,
    width=1920,
    height=1080,))
points = list(range(1, 500))
output_dfs = []
for vn in vns[:]:
    print(vn)
    current = df.loc[df['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]
    current['time_from_stop'] = numpy.zeros(len(current))
    start = 0
    last_stop_time = current['timestamp'].iloc[start]
    j = 0
    for i in range(start + 1, len(current)):
        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:
            if current['idx'].iloc[j] < 50 and current['idx'].iloc[i - 1] >= 490:
                interpolated_idx = current[['time_from_stop', 'idx']].iloc[j:i].copy()

interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan

interpolated_idx.sort_values(by=['time_from_stop'], inplace=True)
                interpolated_idx.set_index('time_from_stop', inplace=True)
                interpolated_idx['idx'].interpolate(method='linear', inplace=True)

interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan
                interpolated_idx.dropna(inplace=True)
                interpolated_idx.reset_index(inplace=True)
                for p in points:
                    if len(interpolated_idx.loc[interpolated_idx['idx'] == p]) == 0:
```

```
                interpolated_idx =
interpolated_idx.append({'idx': p, 'time_from_stop':
numpy.nan}, ignore_index=True)

            interpolated_idx.sort_values(by=['idx'],
inplace=True)

            interpolated_idx.set_index('idx', inplace=True)


interpolated_idx['time_from_stop'].interpolate(method='linea
r', inplace=True)

            fig.add_trace(go.Scatter(x=interpolated_idx.index,
y=interpolated_idx['time_from_stop'],

                mode='lines', # lines+markers

                name=vn + ' ' + str(last_stop_time)))

            pivot_points =
interpolated_idx.loc[interpolated_idx.index.isin(points)]

            pivot_points.index = pivot_points.index.astype(int)

            pivot_points[vn + ' ' + str(last_stop_time)] =
pivot_points['time_from_stop']

            del pivot_points['time_from_stop']

            output_dfs.append(pivot_points)

            # if len(output_dfs) == 2:

            #     break

        j = i

        last_stop_time = current['timestamp'].iloc[i]

    current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()
/ 60

# plt.legend()

fig.show()

#%%

output_df = pandas.concat(output_dfs, axis=1, join='inner')

#%% md

### Pivot points of trips

#%%

import openpyxl

output_df.to_excel('../data/output_df.xlsx')

output_df.transpose()

#%% md

### Filter trips with not enough data at the start

#%%

# output_df1 = output_df.loc[output_df.iloc[20] !=
numpy.nan]

output_df1 = output_df.dropna(thresh=len(output_df) - 20,
axis=1)

output_df1.to_excel('../data/output_df1.xlsx')

output_df1

#%% md

### Normalize time

#%%

output_df2 = output_df1.diff()

output_df2.to_excel('../data/output_df2.xlsx')

output_df2

#%% md

### Trim data from front

#%%

output_df3 = output_df2.dropna()

output_df3.to_excel('../data/output_df3.xlsx')

output_df3

#%% md

### Descriptive data

#%%

output_df3_statistics = output_df3.transpose().describe()

output_df3_statistics

#%% md

### Descriptive data charts

#%%

fig = go.Figure(layout=go.Layout(

    title="Std of time difference",

    xaxis_title="Distance (idx)",

    yaxis_title="std of time difference",

    autosize=False,

    width=1920,

    height=1080,))

fig.add_trace(go.Scatter(x=output_df3_statistics.columns,
y=output_df3_statistics.loc['std'],

    mode='lines', # lines+markers

    name='std'))

#%% md

### Analyze data

#%%

output_df3.iloc[105].describe()

#%%

output_df3.iloc[105].idxmax()

#%%

output_df4 = output_df3.copy()
```

```
del output_df4['AE0548AA 2021-04-23 15:42:13+00:00']

output_df4_statistics = output_df4.transpose().describe()

fig = go.Figure(layout=go.Layout(
    title="Std of time difference without AE0548AA 2021-04-23 15:42:13+00:00",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=output_df4_statistics.columns, y=output_df4_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))
#%% md
### Analyze anomalies
#### Find the difference between current time and mean
#%%
output_df5 = output_df3.copy()
for i in range(len(output_df5)):
    row_mean_value = output_df5.iloc[i].mean()
    # print('mean:', row_mean_value)
    # print('before:' , output_df5.iloc[i])
    output_df5.iloc[i] -= row_mean_value
    # print('after:' , output_df5.iloc[i])
output_df5
#%%
output_df5.to_excel('../data/output_df5.xlsx');
#%% md
#### Group by car vn
#%%
output_df6 = output_df5.abs().groupby(lambda x: x.split(' ')[0], axis=1).sum()
output_df6
#%%
output_df6.to_excel('../data/output_df6.xlsx')
#%% md
#### Find the sum of differences
#%%
output_df7 = output_df6.sum()
output_df7
#%%
output_df7.to_excel('../data/output_df7.xlsx')
#%% md
### Student's coefficient
#%%
# import stats
#
# student = lambda p, k: stats.t.ppf(1 - (1 - p) / 2, k)
test_chart_1_plotly_filtered_fixed_interpolated_osm.ipynb:
#%%
%%javascript
IPython.notebook.kernel.execute('nb_name = "' + IPython.notebook.notebook_name + '"')
#%%
# import dill
# state_db_name = '.'.join(nb_name.split('.')[:-1]) + '.state.db'
#%%
import pandas
import numpy
import plotly.graph_objects as go
import chart_studio.plotly as py
pandas.options.mode.chained_assignment = None  # default='warn'
#%%
df = pandas.read_csv('../data/export_2021-05-12T15_51_16.094Z.csv')
df['timestamp'] = pandas.to_datetime(df['timestamp'])
#%%
# dill.detect.trace(True)
# dill.dump_session(state_db_name)
#%% md
### Loaded data frame sample
#%%
df
#%% md
### Convert pos to latitude and longitude
#%%
# var b = c + "", a = b.split(".");
# if (1 < a.length) {
#     c = a[0];
#     a = a[1];
```

```
#     var e = a.length;
#     if (3 < e) {
#         b = "" + a[0];
#         for (var d = 2; d < e; d++) b += a[d];
#         b += a[1];
#         b = c + "." + b
#     }
#     c = +b
# }
def convertToLatLng(c):
    # print(c)
    b=c+""
    a=b.split(".")
    if 1 < len(a):
        c=a[0]
        a=a[1]
        e=len(a)
        if 3 < e:
            b=""+a[0]
            for d in range(2, e):
                b+=a[d]
            b+=a[1]
            b= c + "." + b
        c = b
    return c
positions = pandas.DataFrame()
positions['pos'] = numpy.zeros(len(df))
positions['idx'] = numpy.zeros(len(df))
end = len(df)
previous_progress = -1
for i in range(0, len(df)):
    progress = i / end * 100
    if progress - int(progress) < 0.01 and progress - previous_progress > 0.1:
        previous_progress = progress
        print(f"{progress:.2f}%")
    # before = df['pos'].iloc[i]
    positions['pos'].iloc[i] = ':'.join([convertToLatLng(x) for x in df['pos'].iloc[i].split(':')])
    # print(before, df['pos'].iloc[i])
positions['posx'] = numpy.zeros(len(df))
positions['posy'] = numpy.zeros(len(df))
for i in range(0, len(df)):
    progress = i / end * 100
    if progress - int(progress) < 0.0001:
        print(f"{progress:.2f}%")
    positions['idx'].iloc[i] = df['idx'].iloc[i]
    positions['posx'].iloc[i] = positions['pos'].iloc[i].split(':')[0]
    positions['posy'].iloc[i] = positions['pos'].iloc[i].split(':')[1]
positions
#%%
df1 = df.copy()
df1['posx'] = positions['posx']
df1['posy'] = positions['posy']
df1
#%%
df1.to_parquet('../data/test_chart_1_plotly_filtered_fixed_interpolated_osm.parquet')
#%%
df1 = pandas.read_parquet('../data/test_chart_1_plotly_filtered_fixed_interpolated_osm.parquet')
#%%
df1
#%% md
### Draw OSM map of route 120
#%%
from IPython.core.display_functions import display
from ipyleaflet import Map, basemaps, Circle
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
with open('../data/route_242.json') as f:
    route_info = json.load(f)
# print(route_info)
for point_str in route_info['points']['forward'].split(' '):
#     print(point)
    point = [float(pos) for pos in point_str.split(',')]
```

```
    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))

for point_str in route_info['points']['backward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m1)

#%%

from IPython.core.display_functions import display

from ipyleaflet import AntPath, Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m2 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_242.json') as f:

    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))

m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],

    color='#74b97f',

    pulse_color='#40ba5e',

    dash_array=[1, 10],

    delay=1000,))

# for point_str in route_info['points']['backward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m2)

#%% md

### Draw OSM map of route 119

#%%

from IPython.core.display_functions import display

from ipyleaflet import Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

# print(route_info)

for point_str in route_info['points']['forward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))

for point_str in route_info['points']['backward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m1)

#%%

from IPython.core.display_functions import display

from ipyleaflet import AntPath, Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m2 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:
```

```
    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]),
radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))

m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['backward'].split(' ')],

    color='#74b97f',

    pulse_color='#40ba5e',

    dash_array=[1, 10],

    delay=1000,))

# for point_str in route_info['points']['backward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(color="green", location=(point[0],
point[1]), radius=1))


display(m2)

#%% md

### Draw idxs on route

#%%

# from matplotlib import cm

# import numpy as np

# n_colors = 501

# colors = cm.rainbow(np.linspace(0, 1, n_colors))

from colorir import StackPalette

import random

colors = StackPalette.new_complementary(100)

for j in range(2):

    for i in range(100):

        colors.swap(i, (i + random.randint(0, 20)) % 100)

colors

#%%

from IPython.core.display_functions import display

from ipyleaflet import AntPath, Map, basemaps, Circle

import json

# Map centred on (60 degrees latitude et -2.2 degrees
longitude)

# Latitude, longitude

m2 = Map(max_zoom=30, center = (48.451293331087356,
35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]),
radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))

m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['backward'].split(' ')],

    color='#74b97f',

    pulse_color='#40ba5e',

    dash_array=[1, 10],

    delay=1000,))

# for point_str in route_info['points']['backward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(color="green", location=(point[0],
point[1]), radius=1))

display(m2)

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()
```

```
    current.reset_index(inplace=True)
#     current = current.loc[current['d'] == 0]


    j = -1


    trips_count = 0


    for idx in range(100, 130):
        print(idx)
        first = True
        for i in range(len(current) - 1):
    #         if i > 100:
    #             break
            row = current.iloc[i]
    #         next_row = current.iloc[i + 1]
            if row['idx'] == idx:
                if first:
                    first = False
                    m2.add_layer(Marker(title=str(idx),
location=(row['posx'], row['posy']), color=colors[idx %
100]))
                else:
                    m2.add_layer(Circle(location=(row['posx'],
row['posy']), color=colors[idx % 100], radius=1))
    #         if j >= 0 and row['idx'] <= next_row['idx']:
    #             coords = ((row['posx'],
row['posy']),(next_row['posx'], next_row['posy']))
    # #             res = client.directions(coords)
    # #             print(i, row['idx'], distance(coords[0],
coords[1]).meters)
    # #             print(res)
    #             m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))
    #         else:
    # #             if j > 0 and row['idx'] > 400:
    # #                 break
    #             j = -1
    #             if row['idx'] > 490 and next_row['idx'] < 50:
    #                 trips_count += 1
    #                 if trips_count > 10:
    #                     break
    #                 j = i + 1
    #
#markers.append(CircleMarker(location=(row['posx'],
row['posy'])))

# display(m2)

#%% md

__We can come to a conclusion that idxs cannot be used for
trips recognition. Probably, time can help us.__

#%%

import numpy as np

df1.groupby(by="vn").agg(min_timestamp=('timestamp',
np.min), max_timestamp=('timestamp', np.max))

#%%

vn

#%% md

### Using time for trip recognition

#%% md

#### 1. Calculate time between neighboring points

#%%

for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]


    j = -1


    trips_count = 0


    current['time_to_previous'] = current['time'].diff()
    current['idx_to_previous'] = current['idx'].diff()
    current['pos_as_previous'] =
current['pos'].eq(current['pos'].shift())
    current['distance_to_previous'] = current['pos'].copy()


    for i in range(len(current)):
        current['distance_to_previous'].iloc[i] =
distance((current['posx'].iloc[i], current['posy'].iloc[i]),
(current['posx'].iloc[i - 1], current['posy'].iloc[i - 1])).meters


#     for idx in range(500):
#         print(idx)
#         first = True
#         for i in range(len(current) - 1):
```

```
#     #         if i > 100:
#     #             break
#             row = current.iloc[i]
#     #         next_row = current.iloc[i + 1]
#             if row['idx'] == idx:
#                 if first:
#                     first = False
#                     m2.add_layer(Marker(title=str(idx), location=(row['posx'], row['posy']), color=colors[idx]))
#                 else:
#                     m2.add_layer(Circle(location=(row['posx'], row['posy']), color=colors[idx], radius=1))
    #         if j >= 0 and row['idx'] <= next_row['idx']:
    #             coords = ((row['posx'], row['posy']),(next_row['posx'], next_row['posy']))
    # #             res = client.directions(coords)
    # #             print(i, row['idx'], distance(coords[0], coords[1]).meters)
    # #             print(res)
    #             m.add_layer(Circle(location=(row['posx'], row['posy']), radius=1))
    #         else:
    # #             if j > 0 and row['idx'] > 400:
    # #                 break
    #             j = -1
    #             if row['idx'] > 490 and next_row['idx'] < 50:
    #                 trips_count += 1
    #                 if trips_count > 10:
    #                     break
    #                 j = i + 1
    # #markers.append(CircleMarker(location=(row['posx'], row['posy'])))
# current.copy().drop().to_excel('time_to_previous_report.xlsx')
current['timestamp'] = current['timestamp'].apply(lambda x: x.replace(tzinfo=None))
current.to_excel('time_to_previous_report1.xlsx')
current
#%%
from IPython.core.display_functions import display
from ipyleaflet import AntPath, Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m2 = Map(max_zoom=30, center = (48.451293331087356, 35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
# print(route_info)
# for point_str in route_info['points']['forward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))


m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],
    color='#7590ba',
    pulse_color='#3f6fba',
    dash_array=[1, 10],
    delay=1000,))
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))
display(m2)
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]


    j = -1


    trips_count = 0
```

```
    for idx in range(500):
        print(idx)
        first = True
        for i in range(len(current) - 1):
    #         if i > 100:
    #             break
            row = current.iloc[i]
    #         next_row = current.iloc[i + 1]
            if row['idx'] == idx:
                if first:
                    first = False
                    m2.add_layer(Marker(title=str(idx),
location=(row['posx'], row['posy']), color=colors[idx]))
                else:
                    m2.add_layer(Circle(location=(row['posx'],
row['posy']), color=colors[idx], radius=1))
    #         if j >= 0 and row['idx'] <= next_row['idx']:
    #             coords = ((row['posx'],
row['posy']),(next_row['posx'], next_row['posy']))
    # #             res = client.directions(coords)
    # #             print(i, row['idx'], distance(coords[0],
coords[1]).meters)
    # #             print(res)
    #             m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))
    #         else:
    # #             if j > 0 and row['idx'] > 400:
    # #                 break
    #             j = -1
    #             if row['idx'] > 490 and next_row['idx'] < 50:
    #                 trips_count += 1
    #                 if trips_count > 10:
    #                     break
    #                 j = i + 1
    #
#markers.append(CircleMarker(location=(row['posx'],
row['posy'])))
# display(m2)
#%% md
### Draw OSM map
#%%
from IPython.core.display_functions import display
from ipyleaflet import Map, basemaps
# Map centred on (60 degrees latitude et -2.2 degrees
longitude)
# Latitude, longitude
m = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)
display(m)
#%%
m.center, m.zoom
#%%
vns = df1['vn'].unique()
#%%
from ipyleaflet import Circle, MarkerCluster
from geopy.distance import distance
import openrouteservice
# client =
openrouteservice.Client(key='5b3ce3597851110001cf624874
906bca85ca48f39a8abde859e1d74c')
#markers = []
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]

    j = -1

    trips_count = 0

    for i in range(len(current) - 1):
#         if i > 100:
#             break
        row = current.iloc[i]
        next_row = current.iloc[i + 1]

        if j >= 0 and row['idx'] <= next_row['idx']:
            coords = ((row['posx'], row['posy']),(next_row['posx'],
next_row['posy']))
#             res = client.directions(coords)
#             print(i, row['idx'], distance(coords[0],
coords[1]).meters)
#             print(res)
```

```
            m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))

        else:

#             if j > 0 and row['idx'] > 400:

#                 break

            j = -1

            if row['idx'] > 490 and next_row['idx'] < 50:

                trips_count += 1

                if trips_count > 10:

                    break

                j = i + 1

            #markers.append(CircleMarker(location=(row['posx'],
row['posy'])))

#marker_cluster = MarkerCluster(markers=markers)

#m.add_layer(marker_cluster)

#%% md

### OpenRouteService calculate distances between stops

#%%

from ipyleaflet import Map, Marker

import openrouteservice

import json

client =
openrouteservice.Client(key='5b3ce3597851110001cf624874
906bca85ca48f39a8abde859e1d74c')

stops_map = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

stops_info = []

# print(route_info)

for i in range(len(route_info['stops']['forward'])):

    stop = route_info['stops']['forward'][i]

    previous_stop = stop if i == 0 else
route_info['stops']['forward'][i - 1]

    print(stop, previous_stop)

    res = client.directions(((previous_stop['y'],
previous_stop['x']), (stop['y'], stop['x'])))

#     print(res)

    distance = res['routes'][0]['summary']['distance'] if
'distance' in res['routes'][0]['summary'] else 0

    if len(res['routes']) > 1:

        print('multiple routes detected', len(res['routes']))

    #     distance = 0

#     print(i, distance(stop['x'], stop['y']).meters)

    stops_map.add_layer(Marker(title=str(i) + ' ' + stop['n'] + '
distance: ' + str(distance), location=(stop['x'], stop['y'])))

    stop['distance'] = distance

    stops_info.append(stop)

display(stops_map)

#%%

pandas.DataFrame(stops_info).to_csv('../data/stops_info.csv')

pandas.DataFrame(stops_info)

#%% md

### Calculate average speed

#%%

from geopy.distance import distance

#markers = []

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]


    stops_speeds = [{'name': stop['n'], 'x': stop['x'], 'y': stop['y'],
'distance': stop['distance'], 'durations': []} for stop in
stops_info]

    stop_index = 0

    last_stop_time = 0


    j = -1


    trips_count = 0


    for i in range(len(current) - 1):

#         if i > 100:

#             break

        row = current.iloc[i]

        next_row = current.iloc[i + 1]


        if row['idx'] <= next_row['idx']:

            for stop_index_local in range(len(stops_speeds)):

                if distance((row['posx'], row['posy']),
(stops_speeds[stop_index_local]['x'],
stops_speeds[stop_index_local]['y'])).meters < 10:

                    if stop_index == stop_index_local - 1 and
last_stop_time > 0:
```

```
stops_speeds[stop_index_local]['durations'].append(row['time'] - last_stop_time)

                print('trip', trips_count, 'found for', stop_index, distance((row['posx'], row['posy']), (stops_speeds[stop_index_local]['x'], stops_speeds[stop_index_local]['y'])).meters)

                last_stop_time = row['time']

                stop_index = stop_index_local + 1

                break

#           coords = ((row['posx'], row['posy']),(stop['x'], next_row['posy']))

#           print(row)

#           print(i, row['idx'], distance(coords[0], coords[1]).meters)

#           print(res)

        else:

#           if j > 0 and row['idx'] > 400:

#               break

#           j = -1

            stop_index = 0

            last_stop_time = 0

            trips_count += 1

#           j = i + 1

#           if row['idx'] > 490 and next_row['idx'] < 50:

#               trips_count += 1

#               if trips_count > 1000:

#                   break

#               j = i + 1

            #markers.append(CircleMarker(location=(row['posx'], row['posy'])))

print(stops_speeds)

print(trips_count)

#marker_cluster = MarkerCluster(markers=markers)

#m.add_layer(marker_cluster)

#%%

#%% md

### Unique serial numbers loaded

#%%

df['vn'].unique()

#%% md

### Interpolated chart of all trips

#%%

fig = go.Figure(layout=go.Layout(

    title="Trips",

    xaxis_title="Distance (idx)",

    yaxis_title="Time (m)",

    autosize=False,

    width=1920,

    height=1080,))

points = list(range(1, 500))

output_dfs = []

for vn in vns[:]:

    print(vn)

    current = df.loc[df['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]

    current['time_from_stop'] = numpy.zeros(len(current))

    start = 0

    last_stop_time = current['timestamp'].iloc[start]

    j = 0

    for i in range(start + 1, len(current)):

        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:

            if current['idx'].iloc[j] < 50 and current['idx'].iloc[i - 1] >= 490:

                interpolated_idx = current[['time_from_stop', 'idx']].iloc[j:i].copy()


interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan


interpolated_idx.sort_values(by=['time_from_stop'], inplace=True)

                interpolated_idx.set_index('time_from_stop', inplace=True)

                interpolated_idx['idx'].interpolate(method='linear', inplace=True)


interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan

                interpolated_idx.dropna(inplace=True)

                interpolated_idx.reset_index(inplace=True)

                for p in points:

                    if len(interpolated_idx.loc[interpolated_idx['idx'] == p]) == 0:

                        interpolated_idx = interpolated_idx.append({'idx': p, 'time_from_stop': numpy.nan}, ignore_index=True)
```

```
            interpolated_idx.sort_values(by=['idx'],
inplace=True)

            interpolated_idx.set_index('idx', inplace=True)


interpolated_idx['time_from_stop'].interpolate(method='linea
r', inplace=True)

            fig.add_trace(go.Scatter(x=interpolated_idx.index,
y=interpolated_idx['time_from_stop'],

                mode='lines', # lines+markers

                name=vn + ' ' + str(last_stop_time)))

            pivot_points =
interpolated_idx.loc[interpolated_idx.index.isin(points)]

            pivot_points.index = pivot_points.index.astype(int)

            pivot_points[vn + ' ' + str(last_stop_time)] =
pivot_points['time_from_stop']

            del pivot_points['time_from_stop']

            output_dfs.append(pivot_points)

            # if len(output_dfs) == 2:

            #     break

        j = i

        last_stop_time = current['timestamp'].iloc[i]

    current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()
/ 60

# plt.legend()

fig.show()

#%%

output_df = pandas.concat(output_dfs, axis=1, join='inner')

#%% md

### Pivot points of trips

#%%

import openpyxl

output_df.to_excel('../data/output_df.xlsx')

output_df.transpose()

#%% md

### Filter trips with not enough data at the start

#%%

# output_df1 = output_df.loc[output_df.iloc[20] !=
numpy.nan]

output_df1 = output_df.dropna(thresh=len(output_df) - 20,
axis=1)

output_df1.to_excel('../data/output_df1.xlsx')

output_df1

#%% md

### Normalize time

#%%

output_df2 = output_df1.diff()

output_df2.to_excel('../data/output_df2.xlsx')

output_df2

#%% md

### Trim data from front

#%%

output_df3 = output_df2.dropna()

output_df3.to_excel('../data/output_df3.xlsx')

output_df3

#%% md

### Descriptive data

#%%

output_df3_statistics = output_df3.transpose().describe()

output_df3_statistics

#%% md

### Descriptive data charts

#%%

fig = go.Figure(layout=go.Layout(

    title="Std of time difference",

    xaxis_title="Distance (idx)",

    yaxis_title="std of time difference",

    autosize=False,

    width=1920,

    height=1080,))

fig.add_trace(go.Scatter(x=output_df3_statistics.columns,
y=output_df3_statistics.loc['std'],

    mode='lines', # lines+markers

    name='std'))

#%% md

### Analyze data

#%%

output_df3.iloc[105].describe()

#%%

output_df3.iloc[105].idxmax()

#%%

output_df4 = output_df3.copy()

del output_df4['AE0548AA 2021-04-23 15:42:13+00:00']

output_df4_statistics = output_df4.transpose().describe()
```

```
fig = go.Figure(layout=go.Layout(
    title="Std of time difference without AE0548AA 2021-04-23 15:42:13+00:00",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=output_df4_statistics.columns, y=output_df4_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))
#%% md
### Analyze anomalies
#### Find the difference between current time and mean
#%%
output_df5 = output_df3.copy()
for i in range(len(output_df5)):
    row_mean_value = output_df5.iloc[i].mean()
    # print('mean:', row_mean_value)
    # print('before:' , output_df5.iloc[i])
    output_df5.iloc[i] -= row_mean_value
    # print('after:' , output_df5.iloc[i])
output_df5
#%%
output_df5.to_excel('../data/output_df5.xlsx');
#%% md
#### Group by car vn
#%%
output_df6 = output_df5.abs().groupby(lambda x: x.split(' ')[0], axis=1).sum()
output_df6
#%%
output_df6.to_excel('../data/output_df6.xlsx')
#%% md
#### Find the sum of differences
#%%
output_df7 = output_df6.sum()
output_df7
#%%
output_df7.to_excel('../data/output_df7.xlsx')
#%% md
### Student's coefficient
#%%
# import stats
#
# student = lambda p, k: stats.t.ppf(1 - (1 - p) / 2, k)
#%%
import dill
import os
os.path.abspath("")
# print(__file__)
# dill.dump_session('')
test_chart_1_plotly_filtered_fixed_interpolated_osm-Copy1.ipynb
#%%
import pandas
import numpy
import plotly.graph_objects as go
import chart_studio.plotly as py
pandas.options.mode.chained_assignment = None  # default='warn'
#%%
df = pandas.read_csv('../data/export_2021-05-12T15_51_16.094Z.csv')
df['timestamp'] = pandas.to_datetime(df['timestamp'])
#%% md
### Loaded data frame sample
#%%
df
#%% md
### Convert pos to latitude and longitude
#%%
# var b = c + "", a = b.split(".");
# if (1 < a.length) {
#     c = a[0];
#     a = a[1];
#     var e = a.length;
#     if (3 < e) {
#         b = "" + a[0];
#         for (var d = 2; d < e; d++) b += a[d];
#         b += a[1];
```

```
#         b = c + "." + b
#     }
#     c = +b
# }
def convertToLatLng(c):
    # print(c)
    b=c+""
    a=b.split(".")
    if 1 < len(a):
        c=a[0]
        a=a[1]
        e=len(a)
        if 3 < e:
            b=""+a[0]
            for d in range(2, e):
                b+=a[d]
            b+=a[1]
            b= c + "." + b
        c = b
    return c
positions = pandas.DataFrame()
positions['pos'] = numpy.zeros(len(df))
positions['idx'] = numpy.zeros(len(df))
end = len(df)
previous_progress = -1
for i in range(0, len(df)):
    progress = i / end * 100
    if progress - int(progress) < 0.01 and progress - previous_progress > 0.1:
        previous_progress = progress
        print(f"{progress:.2f}%")
    # before = df['pos'].iloc[i]
    positions['pos'].iloc[i] = ':'.join([convertToLatLng(x) for x in df['pos'].iloc[i].split(':')])
    # print(before, df['pos'].iloc[i])
positions['posx'] = numpy.zeros(len(df))
positions['posy'] = numpy.zeros(len(df))
for i in range(0, len(df)):
    progress = i / end * 100
    if progress - int(progress) < 0.0001:
        print(f"{progress:.2f}%")
    positions['idx'].iloc[i] = df['idx'].iloc[i]
    positions['posx'].iloc[i] = positions['pos'].iloc[i].split(':')[0]
    positions['posy'].iloc[i] = positions['pos'].iloc[i].split(':')[1]
positions
#%%
df1 = df.copy()
df1['posx'] = positions['posx']
df1['posy'] = positions['posy']
df1
#%% md
### Draw OSM map of route 120
#%%
from IPython.core.display_functions import display
from ipyleaflet import Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
with open('../data/route_242.json') as f:
    route_info = json.load(f)
# print(route_info)
for point_str in route_info['points']['forward'].split(' '):
#     print(point)
    point = [float(pos) for pos in point_str.split(',')]
    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))
for point_str in route_info['points']['backward'].split(' '):
#     print(point)
    point = [float(pos) for pos in point_str.split(',')]
    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))

display(m1)
#%%
from IPython.core.display_functions import display
from ipyleaflet import AntPath, Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
```

```
# Latitude, longitude

m2 = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)

with open('../data/route_242.json') as f:

    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]),
radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))

m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['backward'].split(' ')],

    color='#74b97f',

    pulse_color='#40ba5e',

    dash_array=[1, 10],

    delay=1000,))

# for point_str in route_info['points']['backward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(color="green", location=(point[0],
point[1]), radius=1))


display(m2)

#%% md

### Draw OSM map of route 119

#%%

from IPython.core.display_functions import display

from ipyleaflet import Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees
longitude)

# Latitude, longitude

m1 = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

# print(route_info)

for point_str in route_info['points']['forward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(location=(point[0], point[1]),
radius=1))

for point_str in route_info['points']['backward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(color="green", location=(point[0],
point[1]), radius=1))


display(m1)

#%%

from IPython.core.display_functions import display

from ipyleaflet import AntPath, Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees
longitude)

# Latitude, longitude

m2 = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]),
radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))
```

```
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m2)
#%% md
### Draw OSM map
#%%
from IPython.core.display_functions import display
from ipyleaflet import Map, basemaps
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
display(m)
#%%
m.center, m.zoom
#%%
vns = df1['vn'].unique()
#%%
from ipyleaflet import Circle, MarkerCluster
from geopy.distance import distance
import openrouteservice
# client = openrouteservice.Client(key='5b3ce3597851110001cf624874906bca85ca48f39a8abde859e1d74c')
#markers = []
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]
    j = -1

    trips_count = 0

    for i in range(len(current) - 1):
#         if i > 100:
#             break
        row = current.iloc[i]
        next_row = current.iloc[i + 1]

        if j >= 0 and row['idx'] <= next_row['idx']:
            coords = ((row['posx'], row['posy']),(next_row['posx'], next_row['posy']))
#             res = client.directions(coords)
#             print(i, row['idx'], distance(coords[0], coords[1]).meters)
#             print(res)
            m.add_layer(Circle(location=(row['posx'], row['posy']), radius=1))
        else:
#             if j > 0 and row['idx'] > 400:
#                 break
            j = -1
            if row['idx'] > 490 and next_row['idx'] < 50:
                trips_count += 1
                if trips_count > 10:
                    break
                j = i + 1
            #markers.append(CircleMarker(location=(row['posx'], row['posy'])))
#marker_cluster = MarkerCluster(markers=markers)
#m.add_layer(marker_cluster)
#%% md
### Unique serial numbers loaded
#%%
df['vn'].unique()
#%% md
### Interpolated chart of all trips
#%%
fig = go.Figure(layout=go.Layout(
    title="Trips",
```

```
    xaxis_title="Distance (idx)",
    yaxis_title="Time (m)",
    autosize=False,
    width=1920,
    height=1080,))
points = list(range(1, 500))
output_dfs = []
for vn in vns[:]:
    print(vn)
    current = df.loc[df['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]
    current['time_from_stop'] = numpy.zeros(len(current))
    start = 0
    last_stop_time = current['timestamp'].iloc[start]
    j = 0
    for i in range(start + 1, len(current)):
        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:
            if current['idx'].iloc[j] < 50 and current['idx'].iloc[i - 1] >= 490:
                interpolated_idx = current[['time_from_stop', 'idx']].iloc[j:i].copy()

interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan

interpolated_idx.sort_values(by=['time_from_stop'], inplace=True)
                interpolated_idx.set_index('time_from_stop', inplace=True)
                interpolated_idx['idx'].interpolate(method='linear', inplace=True)

interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan
                interpolated_idx.dropna(inplace=True)
                interpolated_idx.reset_index(inplace=True)
                for p in points:
                    if len(interpolated_idx.loc[interpolated_idx['idx'] == p]) == 0:
                        interpolated_idx = interpolated_idx.append({'idx': p, 'time_from_stop': numpy.nan}, ignore_index=True)
                interpolated_idx.sort_values(by=['idx'], inplace=True)
                interpolated_idx.set_index('idx', inplace=True)
                interpolated_idx['time_from_stop'].interpolate(method='linear', inplace=True)
                fig.add_trace(go.Scatter(x=interpolated_idx.index, y=interpolated_idx['time_from_stop'],
                    mode='lines', # lines+markers
                    name=vn + ' ' + str(last_stop_time)))
                pivot_points = interpolated_idx.loc[interpolated_idx.index.isin(points)]
                pivot_points.index = pivot_points.index.astype(int)
                pivot_points[vn + ' ' + str(last_stop_time)] = pivot_points['time_from_stop']
                del pivot_points['time_from_stop']
                output_dfs.append(pivot_points)
                # if len(output_dfs) == 2:
                #     break
            j = i
            last_stop_time = current['timestamp'].iloc[i]
        current['time_from_stop'].iloc[i] = (current['timestamp'].iloc[i] - last_stop_time).total_seconds() / 60
# plt.legend()
fig.show()
#%%
output_df = pandas.concat(output_dfs, axis=1, join='inner')
#%% md
### Pivot points of trips
#%%
import openpyxl
output_df.to_excel('../data/output_df.xlsx')
output_df.transpose()
#%% md
### Filter trips with not enough data at the start
#%%
# output_df1 = output_df.loc[output_df.iloc[20] != numpy.nan]
output_df1 = output_df.dropna(thresh=len(output_df) - 20, axis=1)
output_df1.to_excel('../data/output_df1.xlsx')
output_df1
#%% md
### Normalize time
#%%
```

```
output_df2 = output_df1.diff()

output_df2.to_excel('../data/output_df2.xlsx')

output_df2

#%% md

### Trim data from front

#%%

output_df3 = output_df2.dropna()

output_df3.to_excel('../data/output_df3.xlsx')

output_df3

#%% md

### Descriptive data

#%%

output_df3_statistics = output_df3.transpose().describe()

output_df3_statistics

#%% md

### Descriptive data charts

#%%

fig = go.Figure(layout=go.Layout(
    title="Std of time difference",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))

fig.add_trace(go.Scatter(x=output_df3_statistics.columns,
y=output_df3_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))

#%% md

### Analyze data

#%%

output_df3.iloc[105].describe()

#%%

output_df3.iloc[105].idxmax()

#%%

output_df4 = output_df3.copy()

del output_df4['AE0548AA 2021-04-23 15:42:13+00:00']

output_df4_statistics = output_df4.transpose().describe()

fig = go.Figure(layout=go.Layout(
    title="Std of time difference without AE0548AA 2021-04-
23 15:42:13+00:00",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))

fig.add_trace(go.Scatter(x=output_df4_statistics.columns,
y=output_df4_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))

#%% md

### Analyze anomalies

#### Find the difference between current time and mean

#%%

output_df5 = output_df3.copy()

for i in range(len(output_df5)):
    row_mean_value = output_df5.iloc[i].mean()
    # print('mean:', row_mean_value)
    # print('before:' , output_df5.iloc[i])
    output_df5.iloc[i] -= row_mean_value
    # print('after:' , output_df5.iloc[i])

output_df5

#%%

output_df5.to_excel('../data/output_df5.xlsx');

#%% md

#### Group by car vn

#%%

output_df6 = output_df5.abs().groupby(lambda x: x.split('
')[0], axis=1).sum()

output_df6

#%%

output_df6.to_excel('../data/output_df6.xlsx')

#%% md

#### Find the sum of differences

#%%

output_df7 = output_df6.sum()

output_df7

#%%

output_df7.to_excel('../data/output_df7.xlsx')

#%% md
```

```
### Student's coefficient

#%%

# import stats

#

# student = lambda p, k: stats.t.ppf(1 - (1 - p) / 2, k)

test_chart_1_plotly_filtered_interpolated_osm-Copy2.ipynb:

#%%

import pandas

import numpy

import plotly.graph_objects as go

import chart_studio.plotly as py

pandas.options.mode.chained_assignment = None  # default='warn'

#%%

df = pandas.read_csv('../data/export_2021-05-12T15_51_16.094Z.csv')

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%% md

### Loaded data frame sample

#%%

df

#%% md

### Convert pos to latitude and longitude

#%%

# var b = c + "", a = b.split(".");

# if (1 < a.length) {

#     c = a[0];

#     a = a[1];

#     var e = a.length;

#     if (3 < e) {

#         b = "" + a[0];

#         for (var d = 2; d < e; d++) b += a[d];

#         b += a[1];

#         b = c + "." + b

#     }

#     c = +b

# }

def convertToLatLng(c):

    # print(c)

    b=c+""

    a=b.split(".")

    if 1 < len(a):

        c=a[0]

        a=a[1]

        e=len(a)

        if 3 < e:

            b=""+a[0]

            for d in range(2, e):

                b+=a[d]

            b+=a[1]

            b= c + "." + b

        c = b

    return c

positions = pandas.DataFrame()

positions['pos'] = numpy.zeros(len(df))

positions['idx'] = numpy.zeros(len(df))

end = len(df)

previous_progress = -1

for i in range(0, len(df)):

    progress = i / end * 100

    if progress - int(progress) < 0.01 and progress - previous_progress > 0.1:

        previous_progress = progress

        print(f"{progress:.2f}%")

    # before = df['pos'].iloc[i]

    positions['pos'].iloc[i] = ':'.join([convertToLatLng(x) for x in df['pos'].iloc[i].split(':')])

    # print(before, df['pos'].iloc[i])

positions['posx'] = numpy.zeros(len(df))

positions['posy'] = numpy.zeros(len(df))

for i in range(0, len(df)):

    progress = i / end * 100

    if progress - int(progress) < 0.0001:

        print(f"{progress:.2f}%")

    positions['idx'].iloc[i] = df['idx'].iloc[i]

    positions['posx'].iloc[i] = positions['pos'].iloc[i].split(':')[0]

    positions['posy'].iloc[i] = positions['pos'].iloc[i].split(':')[1]

positions

#%%

df1 = df.copy()
```

```
df1['posx'] = positions['posx']

df1['posy'] = positions['posy']

df1

#%% md

### Draw OSM map of route 120

#%%

from IPython.core.display_functions import display

from ipyleaflet import Map, basemaps, Circle

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_242.json') as f:

    route_info = json.load(f)

# print(route_info)

for point_str in route_info['points']['forward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))

for point_str in route_info['points']['backward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m1)

#%%

from IPython.core.display_functions import display

from ipyleaflet import AntPath, Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m2 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_242.json') as f:

    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))

m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],

    color='#74b97f',

    pulse_color='#40ba5e',

    dash_array=[1, 10],

    delay=1000,))

# for point_str in route_info['points']['backward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m2)

#%% md

### Draw OSM map of route 119

#%%

from IPython.core.display_functions import display

from ipyleaflet import Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

# print(route_info)

for point_str in route_info['points']['forward'].split(' '):

#     print(point)
```

```
    point = [float(pos) for pos in point_str.split(',')]
    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))
for point_str in route_info['points']['backward'].split(' '):
#     print(point)
    point = [float(pos) for pos in point_str.split(',')]
    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m1)
#%%
from IPython.core.display_functions import display
from ipyleaflet import AntPath, Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m2 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
# print(route_info)
# for point_str in route_info['points']['forward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))


m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],
    color='#7590ba',
    pulse_color='#3f6fba',
    dash_array=[1, 10],
    delay=1000,))
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m2)
#%% md
### Draw OSM map
#%%
from IPython.core.display_functions import display
from ipyleaflet import Map, basemaps
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
display(m)
#%%
m.center, m.zoom
#%%
vns = df1['vn'].unique()
#%%
from ipyleaflet import Circle, MarkerCluster
from geopy.distance import distance
import openrouteservice
# client = openrouteservice.Client(key='5b3ce3597851110001cf624874906bca85ca48f39a8abde859e1d74c')
#markers = []
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]

    j = -1

    trips_count = 0

    for i in range(len(current) - 1):
#         if i > 100:
#             break
```

```
        row = current.iloc[i]
        next_row = current.iloc[i + 1]


        if j >= 0 and row['idx'] <= next_row['idx']:
            coords = ((row['posx'], row['posy']),(next_row['posx'],
next_row['posy']))
#             res = client.directions(coords)
#             print(i, row['idx'], distance(coords[0],
coords[1]).meters)
#             print(res)
            m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))
        else:
#             if j > 0 and row['idx'] > 400:
#                 break
            j = -1
            if row['idx'] > 490 and next_row['idx'] < 50:
                trips_count += 1
                if trips_count > 10:
                    break
                j = i + 1
            #markers.append(CircleMarker(location=(row['posx'],
row['posy'])))
#marker_cluster = MarkerCluster(markers=markers)
#m.add_layer(marker_cluster)
#%% md
### OpenRouteService calculate distances between stops
#%%
from ipyleaflet import Map, Marker
import openrouteservice
client =
openrouteservice.Client(key='5b3ce3597851110001cf624874
906bca85ca48f39a8abde859e1d74c')
stops_map = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
stops_info = []
# print(route_info)
for i in range(len(route_info['stops']['forward'])):
    stop = route_info['stops']['forward'][i]
    print(stop, previous_stop)
    previous_stop = stop if i == 0 else
route_info['stops']['forward'][i - 1]
    res = client.directions(((previous_stop['y'],
previous_stop['x']), (stop['y'], stop['x'])))
#     print(res)
    distance = res['routes'][0]['summary']['distance'] if
'distance' in res['routes'][0]['summary'] else 0
    if len(res['routes']) > 1:
        print('multiple routes detected', len(res['routes']))
    #     distance = 0
#     print(i, distance(stop['x'], stop['y']).meters)
    stops_map.add_layer(Marker(title=str(i) + ' ' + stop['n'] + '
distance: ' + str(distance), location=(stop['x'], stop['y'])))
    stop['distance'] = distance
    stops_info.append(stop)
display(stops_map)
#%%
pandas.DataFrame(stops_info).to_csv('../data/stops_info.csv')
pandas.DataFrame(stops_info)
#%% md
### Calculate average speed
#%%
from geopy.distance import distance
#markers = []
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]


    stops_speeds = [{'name': stop['n'], 'x': stop['x'], 'y': stop['y'],
'distance': stop['distance'], 'durations': []} for stop in
stops_info]
    stop_index = 0
    last_stop_time = 0


    j = -1


    trips_count = 0


    for i in range(len(current) - 1):
#         if i > 100:
#             break
```

```
        row = current.iloc[i]
        next_row = current.iloc[i + 1]


        if j >= 0 and row['idx'] <= next_row['idx']:
            if last_stop_time == 0:
                for stop_index_local in range(len(stops_speeds)):
                    if distance((row['posx'], row['posy']),
(stops_speeds[stop_index_local]['x'],
stops_speeds[stop_index_local]['y'])).meters < 50:
                        if last_stop_time > 0:

stops_speeds[stop_index_local]['durations'].append(row['tim
e'] - last_stop_time)
                        print('found for', stop_index)
                        last_stop_time = row['time']
                        stop_index = stop_index_local + 1
                        break
            else:
                if stop_index > len(stops_speeds) - 1:
                    continue
                if distance((row['posx'], row['posy']),
(stops_speeds[stop_index]['x'],
stops_speeds[stop_index]['y'])).meters < 50:
                    if last_stop_time > 0:

stops_speeds[stop_index]['durations'].append(row['time'] -
last_stop_time)
                    print('found for', stop_index)
                    last_stop_time = row['time']
                    stop_index += 1
#               coords = ((row['posx'], row['posy']),(stop['x'],
next_row['posy']))
#               print(row)
#               print(i, row['idx'], distance(coords[0],
coords[1]).meters)
#               print(res)
        else:
#               if j > 0 and row['idx'] > 400:
#                   break
            j = -1
            stop_index = 0
            last_stop_time = 0
            j = i + 1
#               if row['idx'] > 490 and next_row['idx'] < 50:
#                   trips_count += 1
#                   if trips_count > 1000:
#                       break
#               j = i + 1
            #markers.append(CircleMarker(location=(row['posx'],
row['posy'])))
print(stops_speeds)
print(trips_count)
#marker_cluster = MarkerCluster(markers=markers)
#m.add_layer(marker_cluster)
#%% md
### Unique serial numbers loaded
#%%
df['vn'].unique()
#%% md
### Interpolated chart of all trips
#%%
fig = go.Figure(layout=go.Layout(
    title="Trips",
    xaxis_title="Distance (idx)",
    yaxis_title="Time (m)",
    autosize=False,
    width=1920,
    height=1080,))
points = list(range(1, 500))
output_dfs = []
for vn in vns[:]:
    print(vn)
    current = df.loc[df['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]
    current['time_from_stop'] = numpy.zeros(len(current))
    start = 0
    last_stop_time = current['timestamp'].iloc[start]
    j = 0
    for i in range(start + 1, len(current)):
        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:
            if current['idx'].iloc[j] < 50 and current['idx'].iloc[i -
1] >= 490:
                interpolated_idx = current[['time_from_stop',
'idx']].iloc[j:i].copy()
```

```
interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan


interpolated_idx.sort_values(by=['time_from_stop'], inplace=True)

            interpolated_idx.set_index('time_from_stop', inplace=True)

            interpolated_idx['idx'].interpolate(method='linear', inplace=True)


interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan

            interpolated_idx.dropna(inplace=True)

            interpolated_idx.reset_index(inplace=True)

            for p in points:

                if len(interpolated_idx.loc[interpolated_idx['idx'] == p]) == 0:

                    interpolated_idx = interpolated_idx.append({'idx': p, 'time_from_stop': numpy.nan}, ignore_index=True)

            interpolated_idx.sort_values(by=['idx'], inplace=True)

            interpolated_idx.set_index('idx', inplace=True)


interpolated_idx['time_from_stop'].interpolate(method='linear', inplace=True)

            fig.add_trace(go.Scatter(x=interpolated_idx.index, y=interpolated_idx['time_from_stop'],

                mode='lines', # lines+markers

                name=vn + ' ' + str(last_stop_time)))

            pivot_points = interpolated_idx.loc[interpolated_idx.index.isin(points)]

            pivot_points.index = pivot_points.index.astype(int)

            pivot_points[vn + ' ' + str(last_stop_time)] = pivot_points['time_from_stop']

            del pivot_points['time_from_stop']

            output_dfs.append(pivot_points)

            # if len(output_dfs) == 2:

            #     break

        j = i

        last_stop_time = current['timestamp'].iloc[i]

    current['time_from_stop'].iloc[i] = (current['timestamp'].iloc[i] - last_stop_time).total_seconds() / 60

# plt.legend()

fig.show()

#%%

output_df = pandas.concat(output_dfs, axis=1, join='inner')

#%% md

### Pivot points of trips

#%%

import openpyxl

output_df.to_excel('../data/output_df.xlsx')

output_df.transpose()

#%% md

### Filter trips with not enough data at the start

#%%

# output_df1 = output_df.loc[output_df.iloc[20] != numpy.nan]

output_df1 = output_df.dropna(thresh=len(output_df) - 20, axis=1)

output_df1.to_excel('../data/output_df1.xlsx')

output_df1

#%% md

### Normalize time

#%%

output_df2 = output_df1.diff()

output_df2.to_excel('../data/output_df2.xlsx')

output_df2

#%% md

### Trim data from front

#%%

output_df3 = output_df2.dropna()

output_df3.to_excel('../data/output_df3.xlsx')

output_df3

#%% md

### Descriptive data

#%%

output_df3_statistics = output_df3.transpose().describe()

output_df3_statistics

#%% md

### Descriptive data charts

#%%

fig = go.Figure(layout=go.Layout(

    title="Std of time difference",

    xaxis_title="Distance (idx)",

    yaxis_title="std of time difference",
```

```
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=output_df3_statistics.columns,
y=output_df3_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))
#%% md
### Analyze data
#%%
output_df3.iloc[105].describe()
#%%
output_df3.iloc[105].idxmax()
#%%
output_df4 = output_df3.copy()
del output_df4['AE0548AA 2021-04-23 15:42:13+00:00']
output_df4_statistics = output_df4.transpose().describe()
fig = go.Figure(layout=go.Layout(
    title="Std of time difference without AE0548AA 2021-04-
23 15:42:13+00:00",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=output_df4_statistics.columns,
y=output_df4_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))
#%% md
### Analyze anomalies
#### Find the difference between current time and mean
#%%
output_df5 = output_df3.copy()
for i in range(len(output_df5)):
    row_mean_value = output_df5.iloc[i].mean()
    # print('mean:', row_mean_value)
    # print('before:' , output_df5.iloc[i])
    output_df5.iloc[i] -= row_mean_value
    # print('after:' , output_df5.iloc[i])
output_df5
#%%
output_df5.to_excel('../data/output_df5.xlsx');
#%% md
#### Group by car vn
#%%
output_df6 = output_df5.abs().groupby(lambda x: x.split('
')[0], axis=1).sum()
output_df6
#%%
output_df6.to_excel('../data/output_df6.xlsx')
#%% md
#### Find the sum of differences
#%%
output_df7 = output_df6.sum()
output_df7
#%%
output_df7.to_excel('../data/output_df7.xlsx')
#%% md
### Student's coefficient
#%%
# import stats
#
# student = lambda p, k: stats.t.ppf(1 - (1 - p) / 2, k)
test_chart_1_filtered_fixed_interpolated_osm_with_new_dat
a.ipynb:
#%%
%%javascript
IPython.notebook.kernel.execute('nb_name = "' +
IPython.notebook.notebook_name + '"')
#%%
# import dill
# state_db_name = '.'.join(nb_name.split('.')[:-1]) + '.state.db'
#%%
import pandas
import numpy
import plotly.graph_objects as go
import chart_studio.plotly as py
pandas.options.mode.chained_assignment = None  #
default='warn'
#%%
df = pandas.read_csv('../data/admin.positions_history-
20230313-0016.csv')
```

```
df['timestamp'] = pandas.to_datetime(df['timestamp'])
#%%
# dill.detect.trace(True)
# dill.dump_session(state_db_name)
#%% md
### Loaded data frame sample
#%%
df
#%% md
### Convert pos to latitude and longitude
#%%
# var b = c + "", a = b.split(".");
# if (1 < a.length) {
#     c = a[0];
#     a = a[1];
#     var e = a.length;
#     if (3 < e) {
#         b = "" + a[0];
#         for (var d = 2; d < e; d++) b += a[d];
#         b += a[1];
#         b = c + "." + b
#     }
#     c = +b
# }
def convertToLatLng(c):
    # print(c)
    b=c+""
    a=b.split(".")
    if 1 < len(a):
        c=a[0]
        a=a[1]
        e=len(a)
        if 3 < e:
            b=""+a[0]
            for d in range(2, e):
                b+=a[d]
            b+=a[1]
            b= c + "." + b
        c = b
    return c
positions = pandas.DataFrame()
positions['pos'] = numpy.zeros(len(df))
positions['idx'] = numpy.zeros(len(df))
end = len(df)
previous_progress = -1
for i in range(0, len(df)):
    progress = i / end * 100
    if progress - int(progress) < 0.01 and progress - previous_progress > 0.1:
        previous_progress = progress
        print(f"{progress:.2f}%")
    # before = df['pos'].iloc[i]
    positions['pos'].iloc[i] = ':'.join([convertToLatLng(x) for x in df['pos'].iloc[i].split(':')])
    # print(before, df['pos'].iloc[i])
positions['posx'] = numpy.zeros(len(df))
positions['posy'] = numpy.zeros(len(df))
for i in range(0, len(df)):
    progress = i / end * 100
    if progress - int(progress) < 0.0001:
        print(f"{progress:.2f}%")
    positions['idx'].iloc[i] = df['idx'].iloc[i]
    positions['posx'].iloc[i] = positions['pos'].iloc[i].split(':')[0]
    positions['posy'].iloc[i] = positions['pos'].iloc[i].split(':')[1]
positions
#%%
df1 = df.copy()
df1['posx'] = positions['posx']
df1['posy'] = positions['posy']
df1
#%%
df1.to_parquet('../data/test_chart_1_plotly_filtered_fixed_interpolated_osm_with_new_data.parquet')
#%%
df1 = pandas.read_parquet('../data/test_chart_1_plotly_filtered_fixed_interpolated_osm_with_new_data.parquet')
#%%
df1
#%%
import numpy as np
```

```
df1.groupby(by="vn").agg(min_timestamp=('timestamp', np.min), max_timestamp=('timestamp', np.max), count=('timestamp', np.size)).sort_values(by='count', ascending=False)

#%%

vns[0]

#%% md

### Draw OSM map of route 120

#%%

from IPython.core.display_functions import display

from ipyleaflet import Map, basemaps, Circle

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_242.json') as f:

    route_info = json.load(f)

# print(route_info)

for point_str in route_info['points']['forward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))

for point_str in route_info['points']['backward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m1)

#%%

from IPython.core.display_functions import display

from ipyleaflet import AntPath, Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m2 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_242.json') as f:

    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))

m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],

    color='#74b97f',

    pulse_color='#40ba5e',

    dash_array=[1, 10],

    delay=1000,))

# for point_str in route_info['points']['backward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m2)

#%% md

### Draw OSM map of route 119

#%%

from IPython.core.display_functions import display

from ipyleaflet import Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

# print(route_info)
```

```
for point_str in route_info['points']['forward'].split(' '):
#     print(point)
    point = [float(pos) for pos in point_str.split(',')]
    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))
for point_str in route_info['points']['backward'].split(' '):
#     print(point)
    point = [float(pos) for pos in point_str.split(',')]
    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m1)
#%%
from IPython.core.display_functions import display
from ipyleaflet import AntPath, Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m2 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
# print(route_info)
# for point_str in route_info['points']['forward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))


m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],
    color='#7590ba',
    pulse_color='#3f6fba',
    dash_array=[1, 10],
    delay=1000,))
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m2)
#%% md
### Draw idxs on route
#%%
# from matplotlib import cm
# import numpy as np
# n_colors = 501
# colors = cm.rainbow(np.linspace(0, 1, n_colors))
from colorir import StackPalette
import random
colors = StackPalette.new_complementary(100)
for j in range(2):
    for i in range(100):
        colors.swap(i, (i + random.randint(0, 20)) % 100)
colors
#%%
from IPython.core.display_functions import display
from ipyleaflet import AntPath, Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m2 = Map(max_zoom=30, center = (48.451293331087356, 35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
# print(route_info)
# for point_str in route_info['points']['forward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))
```

```
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],
    color='#7590ba',
    pulse_color='#3f6fba',
    dash_array=[1, 10],
    delay=1000,))
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))
display(m2)
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]

    j = -1

    trips_count = 0

    for idx in range(500):
        print(idx)
        first = True
        for i in range(len(current) - 1):
    #         if i > 100:
    #             break
            row = current.iloc[i]
    #         next_row = current.iloc[i + 1]
            if row['idx'] == idx:
                if first:
                    first = False
                    m2.add_layer(Marker(title=str(idx), location=(row['posx'], row['posy']), color=colors[idx]))
                else:
                    m2.add_layer(Circle(location=(row['posx'], row['posy']), color=colors[idx], radius=1))
    #         if j >= 0 and row['idx'] <= next_row['idx']:
    #             coords = ((row['posx'], row['posy']),(next_row['posx'], next_row['posy']))
    # #             res = client.directions(coords)
    # #             print(i, row['idx'], distance(coords[0], coords[1]).meters)
    # #             print(res)
    #             m.add_layer(Circle(location=(row['posx'], row['posy']), radius=1))
    #         else:
    # #             if j > 0 and row['idx'] > 400:
    # #                 break
    #             j = -1
    #             if row['idx'] > 490 and next_row['idx'] < 50:
    #                 trips_count += 1
    #                 if trips_count > 10:
    #                     break
    #                 j = i + 1
    #
#markers.append(CircleMarker(location=(row['posx'], row['posy'])))
# display(m2)
```

#%% md

__We can come to a conclusion that idxs cannot be used for trips recognition. Probably, time can help us.__

#%% md

Using time for trip recognition

#%% md

1. Calculate time between neighboring points

#%%

```
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]

    j = -1

    trips_count = 0
```

```
    current['time_to_previous'] = current['time'].diff()
    current['idx_to_previous'] = current['idx'].diff()
    current['pos_as_previous'] =
current['pos'].eq(current['pos'].shift())


#     for idx in range(500):
#         print(idx)
#         first = True
#         for i in range(len(current) - 1):
#     #         if i > 100:
#     #             break
#             row = current.iloc[i]
#     #         next_row = current.iloc[i + 1]
#             if row['idx'] == idx:
#                 if first:
#                     first = False
#                     m2.add_layer(Marker(title=str(idx),
location=(row['posx'], row['posy']), color=colors[idx]))
#                 else:
#                     m2.add_layer(Circle(location=(row['posx'],
row['posy']), color=colors[idx], radius=1))
    #         if j >= 0 and row['idx'] <= next_row['idx']:
    #             coords = ((row['posx'],
row['posy']),(next_row['posx'], next_row['posy']))
    # #             res = client.directions(coords)
    # #             print(i, row['idx'], distance(coords[0],
coords[1]).meters)
    # #             print(res)
    #             m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))
    #         else:
    # #             if j > 0 and row['idx'] > 400:
    # #                 break
    #             j = -1
    #             if row['idx'] > 490 and next_row['idx'] < 50:
    #                 trips_count += 1
    #                 if trips_count > 10:
    #                     break
    #                 j = i + 1
    #
#markers.append(CircleMarker(location=(row['posx'],
row['posy'])))
#
current.copy().drop().to_excel('time_to_previous_report.xlsx'
)
current['timestamp'] = current['timestamp'].apply(lambda x:
x.replace(tzinfo=None))
current.to_excel('time_to_previous_report.xlsx')
current
#%%
from IPython.core.display_functions import display
from ipyleaflet import AntPath, Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees
longitude)
# Latitude, longitude
m2 = Map(max_zoom=30, center = (48.451293331087356,
35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
# print(route_info)
# for point_str in route_info['points']['forward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(location=(point[0], point[1]),
radius=1))


m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['forward'].split(' ')],
    color='#7590ba',
    pulse_color='#3f6fba',
    dash_array=[1, 10],
    delay=1000,))
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['backward'].split(' ')],
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
```

```
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))

display(m2)

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]


    j = -1


    trips_count = 0


    for idx in range(500):

        print(idx)

        first = True

        for i in range(len(current) - 1):

    #         if i > 100:

    #             break

            row = current.iloc[i]

    #         next_row = current.iloc[i + 1]

            if row['idx'] == idx:

                if first:

                    first = False

                    m2.add_layer(Marker(title=str(idx), location=(row['posx'], row['posy']), color=colors[idx]))

                else:

                    m2.add_layer(Circle(location=(row['posx'], row['posy']), color=colors[idx], radius=1))

    #         if j >= 0 and row['idx'] <= next_row['idx']:

    #             coords = ((row['posx'], row['posy']),(next_row['posx'], next_row['posy']))

    # #             res = client.directions(coords)

    # #             print(i, row['idx'], distance(coords[0], coords[1]).meters)

    # #             print(res)

    #             m.add_layer(Circle(location=(row['posx'], row['posy']), radius=1))

    #         else:

    # #             if j > 0 and row['idx'] > 400:

    # #                 break

    #             j = -1

    #             if row['idx'] > 490 and next_row['idx'] < 50:

    #                 trips_count += 1

    #                 if trips_count > 10:

    #                     break

    #             j = i + 1

    #
#markers.append(CircleMarker(location=(row['posx'], row['posy'])))

# display(m2)

#%% md

### Draw OSM map

#%%

from IPython.core.display_functions import display

from ipyleaflet import Map, basemaps

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

display(m)

#%%

m.center, m.zoom

#%%

vns = df1['vn'].unique()

#%%

from ipyleaflet import Circle, MarkerCluster

from geopy.distance import distance

import openrouteservice

# client = openrouteservice.Client(key='5b3ce3597851110001cf624874906bca85ca48f39a8abde859e1d74c')

#markers = []

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]


    j = -1


    trips_count = 0


    for i in range(len(current) - 1):

#         if i > 100:
```

```
#             break

        row = current.iloc[i]

        next_row = current.iloc[i + 1]


        if j >= 0 and row['idx'] <= next_row['idx']:

            coords = ((row['posx'], row['posy']),(next_row['posx'],
next_row['posy']))

#             res = client.directions(coords)

#             print(i, row['idx'], distance(coords[0],
coords[1]).meters)

#             print(res)

            m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))

        else:

#             if j > 0 and row['idx'] > 400:

#                 break

            j = -1

            if row['idx'] > 490 and next_row['idx'] < 50:

                trips_count += 1

                if trips_count > 10:

                    break

                j = i + 1

            #markers.append(CircleMarker(location=(row['posx'],
row['posy'])))

#marker_cluster = MarkerCluster(markers=markers)

#m.add_layer(marker_cluster)

#%% md

### OpenRouteService calculate distances between stops

#%%

from ipyleaflet import Map, Marker

import openrouteservice

import json

client =
openrouteservice.Client(key='5b3ce3597851110001cf624874
906bca85ca48f39a8abde859e1d74c')

stops_map = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

stops_info = []

# print(route_info)

for i in range(len(route_info['stops']['forward'])):

    stop = route_info['stops']['forward'][i]

    previous_stop = stop if i == 0 else
route_info['stops']['forward'][i - 1]

    print(stop, previous_stop)

    res = client.directions(((previous_stop['y'],
previous_stop['x']), (stop['y'], stop['x'])))

#     print(res)

    distance = res['routes'][0]['summary']['distance'] if
'distance' in res['routes'][0]['summary'] else 0

    if len(res['routes']) > 1:

        print('multiple routes detected', len(res['routes']))

    #     distance = 0

#     print(i, distance(stop['x'], stop['y']).meters)

    stops_map.add_layer(Marker(title=str(i) + ' ' + stop['n'] + '
distance: ' + str(distance), location=(stop['x'], stop['y'])))

    stop['distance'] = distance

    stops_info.append(stop)

display(stops_map)

#%%

pandas.DataFrame(stops_info).to_csv('../data/stops_info.csv')

pandas.DataFrame(stops_info)

#%%

stops_map.center

#%% md

### Calculate average speed

#%%

from geopy.distance import distance

#markers = []

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]


    stops_speeds = [{'name': stop['n'], 'x': stop['x'], 'y': stop['y'],
'distance': stop['distance'], 'durations': []} for stop in
stops_info]

    stop_index = 0

    last_stop_time = 0

    last_distance = -1


    j = -1
```

```
    trips_count = 0


    for i in range(len(current) - 1):
#         if i > 100:
#             break
        row = current.iloc[i]
        next_row = current.iloc[i + 1]


        if row['idx'] <= next_row['idx']:
            min_distance = 99999999
            min_distance_stop_index = -1
            for stop_index_local in range(len(stops_speeds)):
                current_distance = distance((row['posx'],
row['posy']), (stops_speeds[stop_index_local]['x'],
stops_speeds[stop_index_local]['y'])).meters
                if current_distance < 50 and current_distance <
min_distance:
                    min_distance = current_distance
                    min_distance_stop_index = stop_index_local
            if min_distance_stop_index >= 0:
                if stop_index == min_distance_stop_index and
last_distance != -1 and current_distance < last_distance:
                    pass
                elif stop_index == min_distance_stop_index - 1
and last_stop_time > 0:
                    print('trip', trips_count, 'found for', stop_index,
min_distance)


stops_speeds[min_distance_stop_index]['durations'].append(r
ow['time'] - last_stop_time)
                last_stop_time = row['time']
                last_distance = current_distance
                stop_index = min_distance_stop_index + 1
#             coords = ((row['posx'], row['posy']),(stop['x'],
next_row['posy']))
#             print(row)
#             print(i, row['idx'], distance(coords[0],
coords[1]).meters)
#             print(res)
        else:
#             if j > 0 and row['idx'] > 400:
#                 break
#             j = -1
            stop_index = 0
            last_stop_time = 0
            last_distance = -1
            trips_count += 1
#             j = i + 1
#             if row['idx'] > 490 and next_row['idx'] < 50:
#                 trips_count += 1
#                 if trips_count > 1000:
#                     break
#                 j = i + 1
        #markers.append(CircleMarker(location=(row['posx'],
row['posy'])))
print(stops_speeds)
print(trips_count)
#marker_cluster = MarkerCluster(markers=markers)
#m.add_layer(marker_cluster)
#%%
#%% md
### Unique serial numbers loaded
#%%
df['vn'].unique()
#%% md
### Interpolated chart of all trips
#%%
fig = go.Figure(layout=go.Layout(
    title="Trips",
    xaxis_title="Distance (idx)",
    yaxis_title="Time (m)",
    autosize=False,
    width=1920,
    height=1080,))
points = list(range(1, 500))
output_dfs = []
for vn in vns[:]:
    print(vn)
    current = df.loc[df['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]
    current['time_from_stop'] = numpy.zeros(len(current))
    start = 0
    last_stop_time = current['timestamp'].iloc[start]
```

```
    j = 0

    for i in range(start + 1, len(current)):

        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:

            if current['idx'].iloc[j] < 50 and current['idx'].iloc[i - 1] >= 490:

                interpolated_idx = current[['time_from_stop', 'idx']].iloc[j:i].copy()


interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan


interpolated_idx.sort_values(by=['time_from_stop'], inplace=True)

                interpolated_idx.set_index('time_from_stop', inplace=True)

                interpolated_idx['idx'].interpolate(method='linear', inplace=True)


interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan

                interpolated_idx.dropna(inplace=True)

                interpolated_idx.reset_index(inplace=True)

                for p in points:

                    if len(interpolated_idx.loc[interpolated_idx['idx'] == p]) == 0:

                        interpolated_idx = interpolated_idx.append({'idx': p, 'time_from_stop': numpy.nan}, ignore_index=True)

                interpolated_idx.sort_values(by=['idx'], inplace=True)

                interpolated_idx.set_index('idx', inplace=True)


interpolated_idx['time_from_stop'].interpolate(method='linear', inplace=True)

                fig.add_trace(go.Scatter(x=interpolated_idx.index, y=interpolated_idx['time_from_stop'],

                    mode='lines', # lines+markers

                    name=vn + ' ' + str(last_stop_time)))

                pivot_points = interpolated_idx.loc[interpolated_idx.index.isin(points)]

                pivot_points.index = pivot_points.index.astype(int)

                pivot_points[vn + ' ' + str(last_stop_time)] = pivot_points['time_from_stop']

                del pivot_points['time_from_stop']

                output_dfs.append(pivot_points)

                # if len(output_dfs) == 2:

                #     break

            j = i

            last_stop_time = current['timestamp'].iloc[i]

        current['time_from_stop'].iloc[i] = (current['timestamp'].iloc[i] - last_stop_time).total_seconds() / 60

# plt.legend()

fig.show()

#%%

output_df = pandas.concat(output_dfs, axis=1, join='inner')

#%% md

### Filter out duplicate position rows

#%%

from geopy.distance import distance

from tqdm.notebook import tqdm, trange

client = openrouteservice.Client(base_url='http://192.168.50.201:8080/ors')

#markers = []

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]

#     print(current)


    current = current[(current.shift()['pos'] != current['pos']) | current.index.isin([0])]

    current['time_to_previous'] = current['time'].diff()

    current['idx_to_previous'] = current['idx'].diff()


    current['timestamp'] = current['timestamp'].apply(lambda x: x.replace(tzinfo=None))

#     current.to_excel('with_filtered_duplicates.xlsx')


    current_with_dist = current.copy()

    current_with_dist['distance'] = numpy.zeros(len(current_with_dist))


    for i in tqdm(range(1, len(current_with_dist))):

        previous = current_with_dist.iloc[i - 1]

        this = current_with_dist.iloc[i]

        res = client.directions(((previous['posy'], previous['posx']), (this['posy'], this['posx'])))

    #     print(res)
```

```
        distance = res['routes'][0]['summary']['distance'] if 'distance' in res['routes'][0]['summary'] else -1

        if len(res['routes']) > 1:

            print('multiple routes detected', len(res['routes']))

        current_with_dist['distance'].iloc[i] = distance



current_with_dist.to_excel('with_filtered_duplicates_and_distances.xlsx')


current_with_dist.to_parquet('with_filtered_duplicates_and_distances.parquet')


#     stops_speeds = [{'name': stop['n'], 'x': stop['x'], 'y': stop['y'], 'distance': stop['distance'], 'durations': []} for stop in stops_info]

#     stop_index = 0

#     last_stop_time = 0

#     last_distance = -1


#     j = -1


#     trips_count = 0


#     for i in range(len(current) - 1):

# #         if i > 100:

# #             break

#         row = current.iloc[i]

#         next_row = current.iloc[i + 1]


#         if row['idx'] <= next_row['idx']:

#             min_distance = 99999999

#             min_distance_stop_index = -1

#             for stop_index_local in range(len(stops_speeds)):

#                 current_distance = distance((row['posx'], row['posy']), (stops_speeds[stop_index_local]['x'], stops_speeds[stop_index_local]['y'])).meters

#                 if current_distance < 50 and current_distance < min_distance:

#                     min_distance = current_distance

#                     min_distance_stop_index = stop_index_local

#             if min_distance_stop_index >= 0:

#                 if stop_index == min_distance_stop_index and last_distance != -1 and current_distance < last_distance:

#                     pass

#                 elif stop_index == min_distance_stop_index - 1 and last_stop_time > 0:

#                     print('trip', trips_count, 'found for', stop_index, min_distance)

#                     stops_speeds[min_distance_stop_index]['durations'].append(row['time'] - last_stop_time)

#                 last_stop_time = row['time']

#                 last_distance = current_distance

#                 stop_index = min_distance_stop_index + 1

# #             coords = ((row['posx'], row['posy']),(stop['x'], next_row['posy']))

# #             print(row)

# #             print(i, row['idx'], distance(coords[0], coords[1]).meters)

# #             print(res)

#         else:

# #             if j > 0 and row['idx'] > 400:

# #                 break

# #             j = -1

#             stop_index = 0

#             last_stop_time = 0

#             last_distance = -1

#             trips_count += 1

#             j = i + 1

#             if row['idx'] > 490 and next_row['idx'] < 50:

#                 trips_count += 1

#                 if trips_count > 1000:

#                     break

#                 j = i + 1

            #markers.append(CircleMarker(location=(row['posx'], row['posy'])))

# print(current)

# print(stops_speeds)

# print(trips_count)

#%% md

### Calculate average speed for current

#%%

dist_df = pandas.read_parquet('with_filtered_duplicates_and_distances.parquet')

dist_df
```

```
#%%

dist_df.loc[dist_df['idx_to_previous'] >= 0, 'speed'] =
dist_df['distance'] / dist_df['time_to_previous']

dist_df

#%%

dist_df['speed_kmh'] = dist_df['speed'] * 3.6

#%% md

### Plot anomaly speed points

#%%

dist_df_anomalies = dist_df.copy()

dist_df_anomalies.reset_index(inplace=True)

dist_df_anomalies[dist_df_anomalies['speed_kmh'] > 100]

#%%

from ipyleaflet import Map, Marker

anomalies_map = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)

for index in
dist_df_anomalies[dist_df_anomalies['speed_kmh'] >
100].index[:1]:

    prev = dist_df_anomalies.iloc[index - 1]

    point = dist_df_anomalies.iloc[index]

    print(prev[['posx', 'posy']], point[['posx', 'posy']])

    next_point = dist_df_anomalies.iloc[index + 1]


anomalies_map.add_layer(Marker(title=f"{str(prev['level_0']
)} prev {str(prev['speed_kmh'])} {prev['posx']},
{prev['posy']}", location=(prev['posx'], prev['posy'])))


anomalies_map.add_layer(Marker(title=f"{str(point['level_0']
)} {str(point['speed_kmh'])} {point['posx']}, {point['posy']}",
location=(point['posx'], point['posy'])))


anomalies_map.add_layer(Marker(title=str(next_point['level_
0']) + ' next ' + str(next_point['speed_kmh']),
location=(next_point['posx'], next_point['posy'])))



display(anomalies_map)

#%%

dist_df.loc[dist_df['speed'].isnull(), 'trip_start'] = 1

dist_df.loc[dist_df['speed'].notnull(), 'trip_start'] = 0

dist_df['trip_index'] = dist_df['trip_start'].cumsum()

dist_df

#%%

dist_df.loc[dist_df['trip_start'] != 1, 'trip_distance'] =
dist_df['distance']

dist_df['trip_distance'].fillna(0, inplace=True)

dist_df['cum_trip_distance'] =
dist_df.groupby(['trip_index'])['trip_distance'].cumsum()

dist_df

#%%

dist_df['cum_trip_distance'].max()

#%%

dist_df.to_excel('with_filtered_duplicates_and_distances_and
_speed.xlsx')

#%% md

### Merge segments

#%%

from sqlalchemy import create_engine

engine =
create_engine('mssql+pyodbc://jupyter:272811ssxA@localho
st/EwayResearch?driver=ODBC Driver 17 for SQL
Server&trusted_connection=yes', fast_executemany=True)

#%%

#%%

segments = pandas.DataFrame(np.arange(0,
dist_df['cum_trip_distance'].max(), 10),
columns=['segment_distance'])

segments.to_sql('Segments', engine, if_exists='replace')

segments

#%%

dist_df_by_segments = dist_df.copy()

dist_df_by_segments.reset_index(drop=True, inplace=True)

dist_df_by_segments.to_sql('Points', engine,
if_exists='replace')

# query = "SELECT * FROM dist_df_by_segments JOIN
segments ON cum_trip_distance - trip_distance >
segment_distance and segment_distance >
cum_trip_distance"

#%% md

#### Merge segments in sql server

#%%

sql_query = '''WITH Snapshots AS (

SELECT

                p.[index]

    ,[s]

    ,[hc]

    ,[wf]

    ,[ac]

    ,[d]

    ,[vi]
```

```
    ,[vn]
    ,[ang]
    ,[t]
    ,[idx]
    ,[idx_app_r]
    ,[idx_app_nb]
    ,[time]
    ,[spd]
    ,[pos]
    ,[timestamp]
    ,[posx]
    ,[posy]
    ,[time_to_previous]
    ,[idx_to_previous]
    ,[distance]
    ,[speed]
    ,[speed_kmh]
    ,[trip_start]
    ,[trip_index]
    ,[cum_trip_distance]
    ,[trip_distance]
            ,s.segment_distance
  FROM [Points] p
JOIN Segments s ON s.segment_distance BETWEEN p.[cum_trip_distance] - p.[trip_distance] AND p.[cum_trip_distance]
)
SELECT segment_distance, COUNT([index]) as count_of_points, MIN(speed_kmh) as min_speed, MAX(speed_kmh) as max_speed, AVG(speed_kmh) as avg_speed FROM Snapshots
GROUP BY segment_distance
ORDER BY segment_distance'''
merged_segments_df = pandas.read_sql(sql_query, engine.raw_connection().connection)
merged_segments_df
#%% md
### Descriptive metrics
#%%
merged_segments_df_statistics = merged_segments_df.describe()
merged_segments_df_statistics
#%%
from scipy.stats import ttest_1samp
ttest_1samp(merged_segments_df['avg_speed'], 0)
#%% md
### Pivot points of trips
#%%
import openpyxl
output_df.to_excel('../data/output_df.xlsx')
output_df.transpose()
#%% md
### Filter trips with not enough data at the start
#%%
# output_df1 = output_df.loc[output_df.iloc[20] != numpy.nan]
output_df1 = output_df.dropna(thresh=len(output_df) - 20, axis=1)
output_df1.to_excel('../data/output_df1.xlsx')
output_df1
#%% md
### Normalize time
#%%
output_df2 = output_df1.diff()
output_df2.to_excel('../data/output_df2.xlsx')
output_df2
#%% md
### Trim data from front
#%%
output_df3 = output_df2.dropna()
output_df3.to_excel('../data/output_df3.xlsx')
output_df3
#%% md
### Descriptive data
#%%
output_df3_statistics = output_df3.transpose().describe()
output_df3_statistics
#%% md
### Descriptive data charts
#%%
fig = go.Figure(layout=go.Layout(
    title="Std of time difference",
```

```
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=output_df3_statistics.columns,
y=output_df3_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))
#%% md
### Analyze data
#%%
output_df3.iloc[105].describe()
#%%
output_df3.iloc[105].idxmax()
#%%
output_df4 = output_df3.copy()
del output_df4['AE0548AA 2021-04-23 15:42:13+00:00']
output_df4_statistics = output_df4.transpose().describe()
fig = go.Figure(layout=go.Layout(
    title="Std of time difference without AE0548AA 2021-04-
23 15:42:13+00:00",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=output_df4_statistics.columns,
y=output_df4_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))
#%% md
### Analyze anomalies
#### Find the difference between current time and mean
#%%
output_df5 = output_df3.copy()
for i in range(len(output_df5)):
    row_mean_value = output_df5.iloc[i].mean()
    # print('mean:', row_mean_value)
    # print('before:' , output_df5.iloc[i])
    output_df5.iloc[i] -= row_mean_value
    # print('after:' , output_df5.iloc[i])
output_df5
#%%
output_df5.to_excel('../data/output_df5.xlsx');
#%% md
#### Group by car vn
#%%
output_df6 = output_df5.abs().groupby(lambda x: x.split('
')[0], axis=1).sum()
output_df6
#%%
output_df6.to_excel('../data/output_df6.xlsx')
#%% md
#### Find the sum of differences
#%%
output_df7 = output_df6.sum()
output_df7
#%%
output_df7.to_excel('../data/output_df7.xlsx')
#%% md
### Student's coefficient
#%%
from scipy.stats import t
student = lambda p, k: t.ppf(1 - (1 - p) / 2, k)
#%%
student(0.95, 10)
#%%
#%%
import dill
import os
os.path.abspath("")
# print(__file__)
# dill.dump_session('')
test_chart_1_plotly_filtered_fixed_interpolated_osm_with_n
ew_data-Copy1.ipynb:
#%%
%%javascript
IPython.notebook.kernel.execute('nb_name = "' +
IPython.notebook.notebook_name + '"')
#%%
# import dill
```

```
# state_db_name = '.'.join(nb_name.split('.')[:-1]) + '.state.db'

#%%

import pandas

import numpy

import plotly.graph_objects as go

import chart_studio.plotly as py

pandas.options.mode.chained_assignment = None  # default='warn'

#%%

df = pandas.read_csv('../data/admin.positions_history-20230313-0016.csv')

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%%

# dill.detect.trace(True)

# dill.dump_session(state_db_name)

#%% md

### Loaded data frame sample

#%%

df

#%% md

### Convert pos to latitude and longitude

#%%

# var b = c + "", a = b.split(".");

# if (1 < a.length) {

#     c = a[0];

#     a = a[1];

#     var e = a.length;

#     if (3 < e) {

#         b = "" + a[0];

#         for (var d = 2; d < e; d++) b += a[d];

#         b += a[1];

#         b = c + "." + b

#     }

#     c = +b

# }

def convertToLatLng(c):

    # print(c)

    b=c+""

    a=b.split(".")

    if 1 < len(a):

        c=a[0]

        a=a[1]

        e=len(a)

        if 3 < e:

            b=""+a[0]

            for d in range(2, e):

                b+=a[d]

            b+=a[1]

            b= c + "." + b

        c = b

    return c

positions = pandas.DataFrame()

positions['pos'] = numpy.zeros(len(df))

positions['idx'] = numpy.zeros(len(df))

end = len(df)

previous_progress = -1

for i in range(0, len(df)):

    progress = i / end * 100

    if progress - int(progress) < 0.01 and progress - previous_progress > 0.1:

        previous_progress = progress

        print(f"{progress:.2f}%")

    # before = df['pos'].iloc[i]

    positions['pos'].iloc[i] = ':'.join([convertToLatLng(x) for x in df['pos'].iloc[i].split(':')])

    # print(before, df['pos'].iloc[i])

positions['posx'] = numpy.zeros(len(df))

positions['posy'] = numpy.zeros(len(df))

for i in range(0, len(df)):

    progress = i / end * 100

    if progress - int(progress) < 0.0001:

        print(f"{progress:.2f}%")

    positions['idx'].iloc[i] = df['idx'].iloc[i]

    positions['posx'].iloc[i] = positions['pos'].iloc[i].split(':')[0]

    positions['posy'].iloc[i] = positions['pos'].iloc[i].split(':')[1]

positions

#%%

df1 = df.copy()

df1['posx'] = positions['posx']

df1['posy'] = positions['posy']
```

```
df1

#%%

df1.to_parquet('../data/test_chart_1_plotly_filtered_fixed_interpolated_osm_with_new_data.parquet')

#%%

df1 = pandas.read_parquet('../data/test_chart_1_plotly_filtered_fixed_interpolated_osm_with_new_data.parquet')

#%%

df1

#%%

df1.to_csv('../data/test_chart_1_plotly_filtered_fixed_interpolated_osm_with_new_data_from_parquet.csv')

#%%

import datetime

timestamp = 1619835011

str(datetime.datetime.fromtimestamp(timestamp))

#%%

df1['eway_timestamp'] = pandas.to_datetime(df1['time'], unit='s')

df1['timestamp_diff'] = df1['timestamp'].apply(lambda x: x.replace(tzinfo=None)) - df1['eway_timestamp'].apply(lambda x: x.replace(tzinfo=None))

df1

#%%

import numpy as np

df1.groupby(by="vn").agg(min_timestamp=('timestamp', np.min), max_timestamp=('timestamp', np.max), count=('timestamp', np.size)).sort_values(by='count', ascending=False)

#%%

vns[0]

#%% md

### Draw OSM map of route 120

#%%

from IPython.core.display_functions import display

from ipyleaflet import Map, basemaps, Circle

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_242.json') as f:

    route_info = json.load(f)

# print(route_info)

for point_str in route_info['points']['forward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))

for point_str in route_info['points']['backward'].split(' '):

#     print(point)

    point = [float(pos) for pos in point_str.split(',')]

    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))


display(m1)

#%%

from IPython.core.display_functions import display

from ipyleaflet import AntPath, Map, basemaps

import json

# Map centred on (60 degrees latitude et -2.2 degrees longitude)

# Latitude, longitude

m2 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)

with open('../data/route_242.json') as f:

    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))

m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],
```

```
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))

display(m2)
```

#%% md

Draw OSM map of route 119

#%%

```
from IPython.core.display_functions import display
from ipyleaflet import Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m1 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
# print(route_info)
for point_str in route_info['points']['forward'].split(' '):
#     print(point)
    point = [float(pos) for pos in point_str.split(',')]
    m1.add_layer(Circle(location=(point[0], point[1]), radius=1))
for point_str in route_info['points']['backward'].split(' '):
#     print(point)
    point = [float(pos) for pos in point_str.split(',')]
    m1.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))

display(m1)
```

#%%

```
from IPython.core.display_functions import display
from ipyleaflet import AntPath, Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m2 = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
# print(route_info)
# for point_str in route_info['points']['forward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))

m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],
    color='#7590ba',
    pulse_color='#3f6fba',
    dash_array=[1, 10],
    delay=1000,))
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))

display(m2)
```

#%% md

Draw idxs on route

#%%

```
# from matplotlib import cm
# import numpy as np
# n_colors = 501
# colors = cm.rainbow(np.linspace(0, 1, n_colors))
```

```
from colorir import StackPalette

import random

colors = StackPalette.new_complementary(100)

for j in range(2):

    for i in range(100):

        colors.swap(i, (i + random.randint(0, 20)) % 100)

colors

#%%

from IPython.core.display_functions import display

from ipyleaflet import AntPath, Map, basemaps, Marker,
Circle

import json

# Map centred on (60 degrees latitude et -2.2 degrees
longitude)

# Latitude, longitude

m2 = Map(max_zoom=30, center = (48.451293331087356,
35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

# print(route_info)

# for point_str in route_info['points']['forward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(location=(point[0], point[1]),
radius=1))


m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['forward'].split(' ')],

    color='#7590ba',

    pulse_color='#3f6fba',

    dash_array=[1, 10],

    delay=1000,))

m2.add_layer(AntPath(

    locations=[[float(pos) for pos in point_str.split(',')] for
point_str in route_info['points']['backward'].split(' ')],

    color='#74b97f',

    pulse_color='#40ba5e',

    dash_array=[1, 10],

    delay=1000,))

# for point_str in route_info['points']['backward'].split(' '):

# #     print(point)

#     point = [float(pos) for pos in point_str.split(',')]

#     m.add_layer(Circle(color="green", location=(point[0],
point[1]), radius=1))

display(m2)

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]


    j = -1


    trips_count = 0


    for idx in range(500):

        print(idx)

        first = True

        for i in range(len(current) - 1):

    #         if i > 100:

    #             break

            row = current.iloc[i]

    #         next_row = current.iloc[i + 1]

            if row['idx'] == idx:

                if first:

                    first = False

                    m2.add_layer(Marker(title=str(idx),
location=(row['posx'], row['posy']), color=colors[idx]))

                else:

                    m2.add_layer(Circle(location=(row['posx'],
row['posy']), color=colors[idx], radius=1))

    #         if j >= 0 and row['idx'] <= next_row['idx']:

    #             coords = ((row['posx'],
row['posy']),(next_row['posx'], next_row['posy']))

    # #             res = client.directions(coords)

    # #             print(i, row['idx'], distance(coords[0],
coords[1]).meters)

    # #             print(res)

    #             m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))

    #         else:

    # #             if j > 0 and row['idx'] > 400:

    # #                 break

    #             j = -1
```

```
    #             if row['idx'] > 490 and next_row['idx'] < 50:
    #                 trips_count += 1
    #                 if trips_count > 10:
    #                     break
    #                 j = i + 1
    #
#markers.append(CircleMarker(location=(row['posx'],
row['posy'])))
# display(m2)
#%% md
__We can come to a conclusion that idxs cannot be used for
trips recognition. Probably, time can help us.__
#%% md
### Using time for trip recognition
#%% md
#### 1. Calculate time between neighboring points
#%%
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]

    j = -1

    trips_count = 0

    current['time_to_previous'] = current['time'].diff()
    current['idx_to_previous'] = current['idx'].diff()
    current['pos_as_previous'] =
current['pos'].eq(current['pos'].shift())

#     for idx in range(500):
#         print(idx)
#         first = True
#         for i in range(len(current) - 1):
#     #         if i > 100:
#     #             break
#             row = current.iloc[i]
#     #         next_row = current.iloc[i + 1]
#             if row['idx'] == idx:
#                 if first:
#                     first = False
#                     m2.add_layer(Marker(title=str(idx),
location=(row['posx'], row['posy']), color=colors[idx]))
#                 else:
#                     m2.add_layer(Circle(location=(row['posx'],
row['posy']), color=colors[idx], radius=1))
    #         if j >= 0 and row['idx'] <= next_row['idx']:
    #             coords = ((row['posx'],
row['posy']),(next_row['posx'], next_row['posy']))
    # #             res = client.directions(coords)
    # #             print(i, row['idx'], distance(coords[0],
coords[1]).meters)
    # #             print(res)
    #             m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))
    #         else:
    # #             if j > 0 and row['idx'] > 400:
    # #                 break
    #             j = -1
    #             if row['idx'] > 490 and next_row['idx'] < 50:
    #                 trips_count += 1
    #                 if trips_count > 10:
    #                     break
    #                 j = i + 1
    #
#markers.append(CircleMarker(location=(row['posx'],
row['posy'])))
#
current.copy().drop().to_excel('time_to_previous_report.xlsx'
)
current['timestamp'] = current['timestamp'].apply(lambda x:
x.replace(tzinfo=None))
current.to_excel('time_to_previous_report.xlsx')
current
#%%
from IPython.core.display_functions import display
from ipyleaflet import AntPath, Map, basemaps
import json
# Map centred on (60 degrees latitude et -2.2 degrees
longitude)
# Latitude, longitude
m2 = Map(max_zoom=30, center = (48.451293331087356,
35.04568597068743), zoom=12.0)
with open('../data/route_523.json') as f:
    route_info = json.load(f)
```

```
# print(route_info)
# for point_str in route_info['points']['forward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(location=(point[0], point[1]), radius=1))

m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['forward'].split(' ')],
    color='#7590ba',
    pulse_color='#3f6fba',
    dash_array=[1, 10],
    delay=1000,))
m2.add_layer(AntPath(
    locations=[[float(pos) for pos in point_str.split(',')] for point_str in route_info['points']['backward'].split(' ')],
    color='#74b97f',
    pulse_color='#40ba5e',
    dash_array=[1, 10],
    delay=1000,))
# for point_str in route_info['points']['backward'].split(' '):
# #     print(point)
#     point = [float(pos) for pos in point_str.split(',')]
#     m.add_layer(Circle(color="green", location=(point[0], point[1]), radius=1))
display(m2)
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]

    j = -1

    trips_count = 0

    for idx in range(500):
        print(idx)
        first = True
        for i in range(len(current) - 1):
    #         if i > 100:
    #             break
            row = current.iloc[i]
    #         next_row = current.iloc[i + 1]
            if row['idx'] == idx:
                if first:
                    first = False
                    m2.add_layer(Marker(title=str(idx), location=(row['posx'], row['posy']), color=colors[idx]))
                else:
                    m2.add_layer(Circle(location=(row['posx'], row['posy']), color=colors[idx], radius=1))
    #         if j >= 0 and row['idx'] <= next_row['idx']:
    #             coords = ((row['posx'], row['posy']),(next_row['posx'], next_row['posy']))
    # #             res = client.directions(coords)
    # #             print(i, row['idx'], distance(coords[0], coords[1]).meters)
    # #             print(res)
    #             m.add_layer(Circle(location=(row['posx'], row['posy']), radius=1))
    #         else:
    # #             if j > 0 and row['idx'] > 400:
    # #                 break
    #             j = -1
    #             if row['idx'] > 490 and next_row['idx'] < 50:
    #                 trips_count += 1
    #                 if trips_count > 10:
    #                     break
    #                 j = i + 1
    #
#markers.append(CircleMarker(location=(row['posx'], row['posy'])))
# display(m2)
#%% md
### Draw OSM map
#%%
from IPython.core.display_functions import display
from ipyleaflet import Map, basemaps
# Map centred on (60 degrees latitude et -2.2 degrees longitude)
# Latitude, longitude
m = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
```

```
display(m)

#%%

m.center, m.zoom

#%%

vns = df1['vn'].unique()

#%%

from ipyleaflet import Circle, MarkerCluster

from geopy.distance import distance

import openrouteservice

# client =
openrouteservice.Client(key='5b3ce3597851110001cf624874
906bca85ca48f39a8abde859e1d74c')

#markers = []

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]


    j = -1


    trips_count = 0


    for i in range(len(current) - 1):

#         if i > 100:

#             break

        row = current.iloc[i]

        next_row = current.iloc[i + 1]


        if j >= 0 and row['idx'] <= next_row['idx']:

            coords = ((row['posx'], row['posy']),(next_row['posx'],
next_row['posy']))

#             res = client.directions(coords)

#             print(i, row['idx'], distance(coords[0],
coords[1]).meters)

#             print(res)

            m.add_layer(Circle(location=(row['posx'],
row['posy']), radius=1))

        else:

#             if j > 0 and row['idx'] > 400:

#                 break

            j = -1

            if row['idx'] > 490 and next_row['idx'] < 50:

                trips_count += 1

                if trips_count > 10:

                    break

                j = i + 1

        #markers.append(CircleMarker(location=(row['posx'],
row['posy'])))

#marker_cluster = MarkerCluster(markers=markers)

#m.add_layer(marker_cluster)

#%% md

### OpenRouteService calculate distances between stops

#%%

from ipyleaflet import Map, Marker

import openrouteservice

import json

client =
openrouteservice.Client(key='5b3ce3597851110001cf624874
906bca85ca48f39a8abde859e1d74c')

stops_map = Map(center = (48.451293331087356,
35.04568597068743), zoom=12.0)

with open('../data/route_523.json') as f:

    route_info = json.load(f)

stops_info = []

# print(route_info)

for i in range(len(route_info['stops']['forward'])):

    stop = route_info['stops']['forward'][i]

    previous_stop = stop if i == 0 else
route_info['stops']['forward'][i - 1]

    print(stop, previous_stop)

    res = client.directions(((previous_stop['y'],
previous_stop['x']), (stop['y'], stop['x'])))

#     print(res)

    distance = res['routes'][0]['summary']['distance'] if
'distance' in res['routes'][0]['summary'] else 0

    if len(res['routes']) > 1:

        print('multiple routes detected', len(res['routes']))

    #     distance = 0

#     print(i, distance(stop['x'], stop['y']).meters)

    stops_map.add_layer(Marker(title=str(i) + ' ' + stop['n'] + '
distance: ' + str(distance), location=(stop['x'], stop['y'])))

    stop['distance'] = distance

    stops_info.append(stop)

display(stops_map)

#%%
```

```
pandas.DataFrame(stops_info).to_csv('../data/stops_info.csv')

pandas.DataFrame(stops_info)

#%%

stops_map.center

#%% md

### Calculate average speed

#%%

from geopy.distance import distance

#markers = []

for vn in vns[:1]:

    current = df1.loc[df1['vn'] == vn].copy()

    current.reset_index(inplace=True)

    current = current.loc[current['d'] == 0]


    stops_speeds = [{'name': stop['n'], 'x': stop['x'], 'y': stop['y'],
'distance': stop['distance'], 'durations': []} for stop in
stops_info]

    stop_index = 0

    last_stop_time = 0

    last_distance = -1


    j = -1


    trips_count = 0


    for i in range(len(current) - 1):

#         if i > 100:

#             break

        row = current.iloc[i]

        next_row = current.iloc[i + 1]


        if row['idx'] <= next_row['idx']:

            min_distance = 99999999

            min_distance_stop_index = -1

            for stop_index_local in range(len(stops_speeds)):

                current_distance = distance((row['posx'],
row['posy']), (stops_speeds[stop_index_local]['x'],
stops_speeds[stop_index_local]['y'])).meters

                if current_distance < 50 and current_distance <
min_distance:

                    min_distance = current_distance

                    min_distance_stop_index = stop_index_local

            if min_distance_stop_index >= 0:

                if stop_index == min_distance_stop_index and
last_distance != -1 and current_distance < last_distance:

                    pass

                elif stop_index == min_distance_stop_index - 1
and last_stop_time > 0:

                    print('trip', trips_count, 'found for', stop_index,
min_distance)


stops_speeds[min_distance_stop_index]['durations'].append(r
ow['time'] - last_stop_time)

                last_stop_time = row['time']

                last_distance = current_distance

                stop_index = min_distance_stop_index + 1

#                 coords = ((row['posx'], row['posy']),(stop['x'],
next_row['posy']))

#                 print(row)

#                 print(i, row['idx'], distance(coords[0],
coords[1]).meters)

#                 print(res)

        else:

#             if j > 0 and row['idx'] > 400:

#                 break

#             j = -1

            stop_index = 0

            last_stop_time = 0

            last_distance = -1

            trips_count += 1

#             j = i + 1

#             if row['idx'] > 490 and next_row['idx'] < 50:

#                 trips_count += 1

#                 if trips_count > 1000:

#                     break

#                 j = i + 1

        #markers.append(CircleMarker(location=(row['posx'],
row['posy'])))

print(stops_speeds)

print(trips_count)

#marker_cluster = MarkerCluster(markers=markers)

#m.add_layer(marker_cluster)

#%%

#%% md

### Unique serial numbers loaded
```

```
#%%

df['vn'].unique()

#%% md

### Interpolated chart of all trips

#%%

fig = go.Figure(layout=go.Layout(
    title="Trips",
    xaxis_title="Distance (idx)",
    yaxis_title="Time (m)",
    autosize=False,
    width=1920,
    height=1080,))
points = list(range(1, 500))
output_dfs = []
for vn in vns[:]:
    print(vn)
    current = df.loc[df['vn'] == vn].copy()
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]
    current['time_from_stop'] = numpy.zeros(len(current))
    start = 0
    last_stop_time = current['timestamp'].iloc[start]
    j = 0
    for i in range(start + 1, len(current)):
        if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:
            if current['idx'].iloc[j] < 50 and current['idx'].iloc[i - 1] >= 490:
                interpolated_idx = current[['time_from_stop', 'idx']].iloc[j:i].copy()

interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan

interpolated_idx.sort_values(by=['time_from_stop'], inplace=True)
                interpolated_idx.set_index('time_from_stop', inplace=True)
                interpolated_idx['idx'].interpolate(method='linear', inplace=True)

interpolated_idx['idx'].loc[interpolated_idx['idx'].shift(1) == interpolated_idx['idx']] = numpy.nan
                interpolated_idx.dropna(inplace=True)
                interpolated_idx.reset_index(inplace=True)
                for p in points:
                    if len(interpolated_idx.loc[interpolated_idx['idx'] == p]) == 0:
                        interpolated_idx = interpolated_idx.append({'idx': p, 'time_from_stop': numpy.nan}, ignore_index=True)
                interpolated_idx.sort_values(by=['idx'], inplace=True)
                interpolated_idx.set_index('idx', inplace=True)

interpolated_idx['time_from_stop'].interpolate(method='linear', inplace=True)
                fig.add_trace(go.Scatter(x=interpolated_idx.index, y=interpolated_idx['time_from_stop'],
                    mode='lines', # lines+markers
                    name=vn + ' ' + str(last_stop_time)))
                pivot_points = interpolated_idx.loc[interpolated_idx.index.isin(points)]
                pivot_points.index = pivot_points.index.astype(int)
                pivot_points[vn + ' ' + str(last_stop_time)] = pivot_points['time_from_stop']
                del pivot_points['time_from_stop']
                output_dfs.append(pivot_points)
                # if len(output_dfs) == 2:
                #     break
            j = i
            last_stop_time = current['timestamp'].iloc[i]
        current['time_from_stop'].iloc[i] = (current['timestamp'].iloc[i] - last_stop_time).total_seconds() / 60
# plt.legend()
fig.show()

#%%

output_df = pandas.concat(output_dfs, axis=1, join='inner')

#%% md

### Filter out duplicate position rows

#%%

from geopy.distance import distance
from tqdm.notebook import tqdm, trange
client = openrouteservice.Client(base_url='http://192.168.50.201:8080/ors')
#markers = []
for vn in vns[:1]:
    current = df1.loc[df1['vn'] == vn].copy()
```

```
    current.reset_index(inplace=True)
    current = current.loc[current['d'] == 0]
#     print(current)

    current = current[(current.shift()['pos'] != current['pos']) | current.index.isin([0])]
    current['time_to_previous'] = current['time'].diff()
    current['idx_to_previous'] = current['idx'].diff()

    current['timestamp'] = current['timestamp'].apply(lambda x: x.replace(tzinfo=None))
#     current.to_excel('with_filtered_duplicates.xlsx')

    current_with_dist = current.copy()
    current_with_dist['distance'] = numpy.zeros(len(current_with_dist))

    for i in tqdm(range(1, len(current_with_dist))):
        previous = current_with_dist.iloc[i - 1]
        this = current_with_dist.iloc[i]
        res = client.directions(((previous['posy'], previous['posx']), (this['posy'], this['posx'])))
    #     print(res)
        distance = res['routes'][0]['summary']['distance'] if 'distance' in res['routes'][0]['summary'] else -1
        if len(res['routes']) > 1:
            print('multiple routes detected', len(res['routes']))
        current_with_dist['distance'].iloc[i] = distance

current_with_dist.to_excel('with_filtered_duplicates_and_distances.xlsx')

current_with_dist.to_parquet('with_filtered_duplicates_and_distances.parquet')

#     stops_speeds = [{'name': stop['n'], 'x': stop['x'], 'y': stop['y'], 'distance': stop['distance'], 'durations': []} for stop in stops_info]
#     stop_index = 0
#     last_stop_time = 0
#     last_distance = -1

#     j = -1
#     trips_count = 0

#     for i in range(len(current) - 1):
# #         if i > 100:
# #             break
#         row = current.iloc[i]
#         next_row = current.iloc[i + 1]

#         if row['idx'] <= next_row['idx']:
#             min_distance = 99999999
#             min_distance_stop_index = -1
#             for stop_index_local in range(len(stops_speeds)):
#                 current_distance = distance((row['posx'], row['posy']), (stops_speeds[stop_index_local]['x'], stops_speeds[stop_index_local]['y'])).meters
#                 if current_distance < 50 and current_distance < min_distance:
#                     min_distance = current_distance
#                     min_distance_stop_index = stop_index_local
#             if min_distance_stop_index >= 0:
#                 if stop_index == min_distance_stop_index and last_distance != -1 and current_distance < last_distance:
#                     pass
#                 elif stop_index == min_distance_stop_index - 1 and last_stop_time > 0:
#                     print('trip', trips_count, 'found for', stop_index, min_distance)
#                     stops_speeds[min_distance_stop_index]['durations'].append(row['time'] - last_stop_time)
#                     last_stop_time = row['time']
#                     last_distance = current_distance
#                     stop_index = min_distance_stop_index + 1
# #             coords = ((row['posx'], row['posy']),(stop['x'], next_row['posy']))
# #             print(row)
# #             print(i, row['idx'], distance(coords[0], coords[1]).meters)
# #             print(res)
#         else:
# #             if j > 0 and row['idx'] > 400:
# #                 break
# #             j = -1
```

```
#             stop_index = 0
#             last_stop_time = 0
#             last_distance = -1
#             trips_count += 1
#             j = i + 1
#             if row['idx'] > 490 and next_row['idx'] < 50:
#                 trips_count += 1
#                 if trips_count > 1000:
#                     break
#                 j = i + 1
            #markers.append(CircleMarker(location=(row['posx'], row['posy'])))
# print(current)
# print(stops_speeds)
# print(trips_count)
#%% md
### Calculate average speed for current
#%%
dist_df = pandas.read_parquet('with_filtered_duplicates_and_distances.parquet')
dist_df
#%%
dist_df.loc[dist_df['idx_to_previous'] >= 0, 'speed'] = dist_df['distance'] / dist_df['time_to_previous']
dist_df
#%%
dist_df['speed_kmh'] = dist_df['speed'] * 3.6
#%% md
### Plot anomaly speed points
#%%
dist_df_anomalies = dist_df.copy()
dist_df_anomalies.reset_index(inplace=True)
dist_df_anomalies[dist_df_anomalies['speed_kmh'] > 100]
#%%
from ipyleaflet import Map, Marker
anomalies_map = Map(center = (48.451293331087356, 35.04568597068743), zoom=12.0)
for index in dist_df_anomalies[dist_df_anomalies['speed_kmh'] > 100].index[:1]:
    prev = dist_df_anomalies.iloc[index - 1]
    point = dist_df_anomalies.iloc[index]
    print(prev[['posx', 'posy']], point[['posx', 'posy']])
    next_point = dist_df_anomalies.iloc[index + 1]

anomalies_map.add_layer(Marker(title=f"{str(prev['level_0'])} prev {str(prev['speed_kmh'])} {prev['posx']}, {prev['posy']}", location=(prev['posx'], prev['posy'])))

anomalies_map.add_layer(Marker(title=f"{str(point['level_0'])} {str(point['speed_kmh'])} {point['posx']}, {point['posy']}", location=(point['posx'], point['posy'])))

anomalies_map.add_layer(Marker(title=str(next_point['level_0']) + ' next ' + str(next_point['speed_kmh']), location=(next_point['posx'], next_point['posy'])))

display(anomalies_map)
#%%
dist_df.loc[dist_df['speed'].isnull(), 'trip_start'] = 1
dist_df.loc[dist_df['speed'].notnull(), 'trip_start'] = 0
dist_df['trip_index'] = dist_df['trip_start'].cumsum()
dist_df
#%%
dist_df.loc[dist_df['trip_start'] != 1, 'trip_distance'] = dist_df['distance']
dist_df['trip_distance'].fillna(0, inplace=True)
dist_df['cum_trip_distance'] = dist_df.groupby(['trip_index'])['trip_distance'].cumsum()
dist_df
#%%
dist_df['cum_trip_distance'].max()
#%%
dist_df['trip_distance'].describe()
#%%
dist_df.to_excel('with_filtered_duplicates_and_distances_and_speed.xlsx')
#%% md
### Merge segments
#%%
from sqlalchemy import create_engine
engine = create_engine('mssql+pyodbc://jupyter:272811ssxA@localhost/EwayResearch?driver=ODBC Driver 17 for SQL Server&trusted_connection=yes', fast_executemany=True)
#%%
#%%
```

```
segments = pandas.DataFrame(np.arange(0,
dist_df['cum_trip_distance'].max(), 10),
columns=['segment_distance'])

segments.to_sql('Segments', engine, if_exists='replace')

segments

#%%

dist_df_by_segments = dist_df.copy()

dist_df_by_segments.reset_index(drop=True, inplace=True)

dist_df_by_segments.to_sql('Points', engine,
if_exists='replace')

# query = "SELECT * FROM dist_df_by_segments JOIN
segments ON cum_trip_distance - trip_distance >
segment_distance and segment_distance >
cum_trip_distance"

#%% md

#### Merge segments in sql server

#%%

sql_query = '''WITH Snapshots AS (

SELECT

                p.[index]

    ,[s]

    ,[hc]

    ,[wf]

    ,[ac]

    ,[d]

    ,[vi]

    ,[vn]

    ,[ang]

    ,[t]

    ,[idx]

    ,[idx_app_r]

    ,[idx_app_nb]

    ,[time]

    ,[spd]

    ,[pos]

    ,[timestamp]

    ,[posx]

    ,[posy]

    ,[time_to_previous]

    ,[idx_to_previous]

    ,[distance]

    ,[speed]

    ,[speed_kmh]

    ,[trip_start]

    ,[trip_index]

    ,[cum_trip_distance]

    ,[trip_distance]

        ,s.segment_distance

 FROM [Points] p

JOIN Segments s ON s.segment_distance BETWEEN
p.[cum_trip_distance] - p.[trip_distance] AND
p.[cum_trip_distance]

)

SELECT segment_distance, COUNT([index]) as
count_of_points, MIN(speed_kmh) as min_speed,
MAX(speed_kmh) as max_speed, AVG(speed_kmh) as
avg_speed FROM Snapshots

GROUP BY segment_distance

ORDER BY segment_distance'''

merged_segments_df = pandas.read_sql(sql_query,
engine.raw_connection().connection)

merged_segments_df

#%% md

### Descriptive metrics

#%%

merged_segments_df_statistics =
merged_segments_df.describe()

merged_segments_df_statistics

#%%

from scipy.stats import ttest_1samp

ttest_1samp(merged_segments_df['avg_speed'], 0)

#%% md

### Pivot points of trips

#%%

import openpyxl

output_df.to_excel('../data/output_df.xlsx')

output_df.transpose()

#%% md

### Filter trips with not enough data at the start

#%%

# output_df1 = output_df.loc[output_df.iloc[20] !=
numpy.nan]

output_df1 = output_df.dropna(thresh=len(output_df) - 20,
axis=1)

output_df1.to_excel('../data/output_df1.xlsx')
```

```
output_df1

#%% md

### Normalize time

#%%

output_df2 = output_df1.diff()

output_df2.to_excel('../data/output_df2.xlsx')

output_df2

#%% md

### Trim data from front

#%%

output_df3 = output_df2.dropna()

output_df3.to_excel('../data/output_df3.xlsx')

output_df3

#%% md

### Descriptive data

#%%

output_df3_statistics = output_df3.transpose().describe()

output_df3_statistics

#%% md

### Descriptive data charts

#%%

fig = go.Figure(layout=go.Layout(
    title="Std of time difference",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))

fig.add_trace(go.Scatter(x=output_df3_statistics.columns,
y=output_df3_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))

#%% md

### Analyze data

#%%

output_df3.iloc[105].describe()

#%%

output_df3.iloc[105].idxmax()

#%%

output_df4 = output_df3.copy()

del output_df4['AE0548AA 2021-04-23 15:42:13+00:00']

output_df4_statistics = output_df4.transpose().describe()

fig = go.Figure(layout=go.Layout(
    title="Std of time difference without AE0548AA 2021-04-
23 15:42:13+00:00",
    xaxis_title="Distance (idx)",
    yaxis_title="std of time difference",
    autosize=False,
    width=1920,
    height=1080,))

fig.add_trace(go.Scatter(x=output_df4_statistics.columns,
y=output_df4_statistics.loc['std'],
    mode='lines', # lines+markers
    name='std'))

#%% md

### Analyze anomalies

#### Find the difference between current time and mean

#%%

output_df5 = output_df3.copy()

for i in range(len(output_df5)):
    row_mean_value = output_df5.iloc[i].mean()
    # print('mean:', row_mean_value)
    # print('before:' , output_df5.iloc[i])
    output_df5.iloc[i] -= row_mean_value
    # print('after:' , output_df5.iloc[i])

output_df5

#%%

output_df5.to_excel('../data/output_df5.xlsx');

#%% md

#### Group by car vn

#%%

output_df6 = output_df5.abs().groupby(lambda x: x.split('
')[0], axis=1).sum()

output_df6

#%%

output_df6.to_excel('../data/output_df6.xlsx')

#%% md

#### Find the sum of differences

#%%

output_df7 = output_df6.sum()

output_df7
```

```
#%%

output_df7.to_excel('../data/output_df7.xlsx')

#%% md

### Student's coefficient

#%%

from scipy.stats import t

student = lambda p, k: t.ppf(1 - (1 - p) / 2, k)

#%%

student(0.95, 10)

#%%

#%%

import dill

import os

os.path.abspath("")

# print(__file__)

# dill.dump_session('')

test_chart_1_plotly_geopy.ipynb:

#%%

import pandas

import numpy

import plotly.graph_objects as go

import geopy.distance

pandas.options.mode.chained_assignment = None  # default='warn'

#%%

df = pandas.read_csv('../data/export_2021-05-12T15_51_16.094Z.csv')

df = df.loc[df['d'] == 0]

df['timestamp'] = pandas.to_datetime(df['timestamp'])

#%%

# var b = c + "", a = b.split(".");

# if (1 < a.length) {

#     c = a[0];

#     a = a[1];

#     var e = a.length;

#     if (3 < e) {

#         b = "" + a[0];

#         for (var d = 2; d < e; d++) b += a[d];

#         b += a[1];

#         b = c + "." + b

#     }

#     c = +b

# }

def convertToLatLng(c):

    # print(c)

    b=c+""

    a=b.split(".")

    if 1 < len(a):

        c=a[0]

        a=a[1]

        e=len(a)

        if 3 < e:

            b=""+a[0]

            for d in range(2, e):

                b+=a[d]

            b+=a[1]

            b= c + "." + b

        c = b

    return c

positions = pandas.DataFrame()

positions['pos'] = numpy.zeros(len(df))

positions['idx'] = numpy.zeros(len(df))

end = len(df)

for i in range(0, len(df)):

    progress = i / end * 100

    if progress - int(progress) < 0.0001:

        print(progress)

    # before = df['pos'].iloc[i]

    positions['pos'].iloc[i] = ':'.join([convertToLatLng(x) for x in df['pos'].iloc[i].split(':')])

    # print(before, df['pos'].iloc[i])

#%%

#%%

positions['posx'] = numpy.zeros(len(df))

positions['posy'] = numpy.zeros(len(df))

for i in range(0, len(df)):

    progress = i / end * 100

    if progress - int(progress) < 0.0001:

        print(progress)

    positions['idx'].iloc[i] = df['idx'].iloc[i]
```

```
    positions['posx'].iloc[i] = positions['pos'].iloc[i].split(':')[0]
    positions['posy'].iloc[i] = positions['pos'].iloc[i].split(':')[1]
#%%
#%%
positions
#%%
fig = go.Figure(layout=go.Layout(
    autosize=False,
    width=1920,
    height=1080,))
fig.add_trace(go.Scatter(x=positions['posx'],
y=positions['posy'],
    # mode='lines', # lines+markers
    name='route'))
# for vn in vns:
#     print(vn)
#     current = df.loc[df['vn'] == vn].copy()
#
#     current.iloc[1:].loc[current['idx'].shift(-1) ==
current['idx'], 'idx'] = numpy.nan
#     current['idx'] = current['idx'].interpolate(method='linear')
#
#     current['time_from_stop'] = numpy.zeros(len(current))
#     current['m_from_stop'] = numpy.zeros(len(current))
#
#     start = 0
#     if current['idx'].iloc[start] >= 50:
#         for i in range(1, len(current)):
#             if current['idx'].iloc[i] < current['idx'].iloc[i - 1] and
current['idx'].iloc[i] < 50:
#                 start = i
#                 break
#     last_stop_time = current['timestamp'].iloc[start]
#     last_stop_position = current['pos'].iloc[start]
#     last_m_from_stop = 0
#     j = start
#     cut = False
#     for i in range(start + 1, len(current)):
#         # if not cut and current['idx'].iloc[i] -
current['idx'].iloc[j] > 50:
#         #     print(i)
#         #     cut = True
#         #     last_stop_time = current['timestamp'].iloc[i]
#         #     j = i
#         #     continue
#
#         if current['idx'].iloc[i] < current['idx'].iloc[i - 1]:
#             if current['time_from_stop'].iloc[i - 1] < 20000:
#                 #
fig.add_trace(go.Scatter(x=current['idx'].iloc[j:i],
y=current['time_from_stop'].iloc[j:i],
#                 #     mode='lines', # lines+markers
#                 #     name=vn + ' ' + str(last_stop_time)))
#
fig.add_trace(go.Scatter(x=current['m_from_stop'].iloc[j:i],
y=current['time_from_stop'].iloc[j:i],
#                     mode='lines', # lines+markers
#                     name='m ' + vn + ' ' + str(last_stop_time)))
#
#             if current['idx'].iloc[i] >= 50:
#                 for k in range(i + 1, len(current)):
#                     if current['idx'].iloc[k] < current['idx'].iloc[k -
1] and current['idx'].iloc[k] < 50:
#                         i = k
#                         break
#             break
#             j = i
#             cut = False
#             last_stop_time = current['timestamp'].iloc[i]
#             last_stop_position = current['pos'].iloc[i]
#             last_m_from_stop = 0
#             # print(last_stop_time)
#
#         current['time_from_stop'].iloc[i] =
(current['timestamp'].iloc[i] - last_stop_time).total_seconds()
#         current['m_from_stop'].iloc[i] = last_m_from_stop +
geopy.distance.distance(current['pos'].iloc[i].split(':'),
last_stop_position.split(':')).meters.real
#         last_stop_position = current['pos'].iloc[i]
#         last_m_from_stop = current['m_from_stop'].iloc[i]
    # df1 = pandas.concat([df1, current[['time', vn]]])
# plt.legend()
fig.show()
#%%
```

current[j:i]

index.js:

```
const EventSource = require('eventsource');

var window = {

   Event: function() {}

};

var document = {

   createElement: () => {}

}

function Element() {}

function _getDistanceBetweenPoints(a,b,c,f){b-
=f;a=Math.sin(a*Math.PI/180)*Math.sin(c*Math.PI/180)+M
ath.cos(a*Math.PI/180)*Math.cos(c*Math.PI/180)*Math.cos
(b*Math.PI/180);a=Math.acos(a);a=a/Math.PI*180;return
Math.abs(111189.57696*a)}

function _moveCoordinate(a,b,c,f){for(var
k=0,s=a,h=b;k<f;){switch(c){case "lat-top":s+=8E-
4;break;case "lat-bottom":s-=8E-4;break;case "lng-
right":h+=8E-4;break;case "lng-left":h-=8E-
4;break;default:return console.error("_moveCoordinate:
Wrong direction
"'+c+'"'),[s,h]}k=_getDistanceBetweenPoints(a,b,s,h)}return[
s,h]}

// var
eventMixin={on:function(a,b){this._eventHandlers||(this._ev
entHandlers={});this._eventHandlers[a]||(this._eventHandlers
[a]=[]);this._eventHandlers[a].push(b)},off:function(a,b){var
c=this._eventHandlers&&this._eventHandlers[a];if(c)for(var
f=0;f<c.length;f++)c[f]==b&&c.splice(f--
,1)},trigger:function(a){if(this._eventHandlers&&this._event
Handlers[a])for(var
b=this._eventHandlers[a],c=0;c<b.length;c++)b[c].apply(this,
[].slice.call(arguments,1))}};(new Function('{function
n(b,g){var
c=(b&65535)+(g&65535);return(b>>16)+(g>>16)+(c>>16)<
<16|c&65535}function
h(b,g,c,a,d,e){b=n(n(g,b),n(a,e));return n(b<<d|b>>>32-
d,c)}function k(b,g,c,a,d,e,f){return
h(g&c|~g&a,b,g,d,e,f)}function l(b,g,c,a,d,e,f){return
h(g&a|c&~a,b,g,d,e,f)}function m(b,g,c,a,d,e,f){return
h(c^(g|~a),b,g,d,e,f)}function
p(b,g){b[g>>5]|=128<<g%32;b[(g+64>>>9<<4)+14]=g;var
c,a=1732584193,d=-271733879,e=-
1732584194,f=271733878;for(c=0;c<b.length;c+=16){var
p=a;var q=d;var r=e;var t=f;a=k(a,d,e,f,b[c],7,-
680876936);f=k(f,a,d,e,b[c+1],12,-
389564586);e=k(e,f,a,d,b[c+2],17,606105819);d=k(d,e,f,a,b[
c+3],22,-1044525330);a=k(a,d,e,f,b[c+4],7,-
176418897);f=k(f,a,d,e,b[c+5],12,1200080426);e=k(e,f,a,d,b[
c+6],17,-1473231341);d=k(d,e,f,a,b[c+7],22,-
45705983);a=k(a,d,e,f,b[c+8],7,1770035416);f=k(f,a,d,e,b[c+
9],12,-1958414417);e=k(e,f,a,d,b[c+10],17,-
42063);d=k(d,e,f,a,b[c+11],22,-
1990404162);a=k(a,d,e,f,b[c+12],7,1804603682);f=k(f,a,d,e,
b[c+13],12,-40341101);e=k(e,f,a,d,b[c+14],17,-
1502002290);d=k(d,e,f,a,b[c+15],22,1236535329);a=l(a,d,e,f
,b[c+1],5,-165796510);f=l(f,a,d,e,b[c+6],9,-
1069501632);e=l(e,f,a,d,b[c+11],14,643717713);d=l(d,e,f,a,b
[c],20,-373897302);a=l(a,d,e,f,b[c+5],5,-
701558691);f=l(f,a,d,e,b[c+10],9,38016083);e=l(e,f,a,d,b[c+
15],14,-660478335);d=l(d,e,f,a,b[c+4],20,-
405537848);a=l(a,d,e,f,b[c+9],5,568446438);f=l(f,a,d,e,b[c+
14],9,-1019803690);e=l(e,f,a,d,b[c+3],14,-
187363961);d=l(d,e,f,a,b[c+8],20,1163531501);a=l(a,d,e,f,b[
c+13],5,-1444681467);f=l(f,a,d,e,b[c+2],9,-
51403784);e=l(e,f,a,d,b[c+7],14,1735328473);d=l(d,e,f,a,b[c
+12],20,-1926607734);a=h(d^e^f,a,d,b[c+5],4,-
378558);f=h(a^d^e,f,a,b[c+8],11,-
2022574463);e=h(f^a^d,e,f,b[c+11],16,1839030562);d=h(e^f
^a,d,e,b[c+14],23,-35309556);a=h(d^e^f,a,d,b[c+1],4,-
1530992060);f=h(a^d^e,f,a,b[c+4],11,1272893353);e=h(f^a^
d,e,f,b[c+7],16,-155497632);d=h(e^f^a,d,e,b[c+10],23,-
1094730640);a=h(d^e^f,a,d,b[c+13],4,681279174);f=h(a^d^e
,f,a,b[c],11,-358537222);e=h(f^a^d,e,f,b[c+3],16,-
722521979);d=h(e^f^a,d,e,b[c+6],23,76029189);a=h(d^e^f,a,
d,b[c+9],4,-640364487);f=h(a^d^e,f,a,b[c+12],11,-
421815835);e=h(f^a^d,e,f,b[c+15],16,530742520);d=h(e^f^a
,d,e,b[c+2],23,-995338651);a=m(a,d,e,f,b[c],6,-
198630844);f=m(f,a,d,e,b[c+7],10,1126891415);e=m(e,f,a,d,
b[c+14],15,-1416354905);d=m(d,e,f,a,b[c+5],21,-
57434055);a=m(a,d,e,f,b[c+12],6,1700485571);f=m(f,a,d,e,b
[c+3],10,-1894986606);e=m(e,f,a,d,b[c+10],15,-
1051523);d=m(d,e,f,a,b[c+1],21,-
2054922799);a=m(a,d,e,f,b[c+8],6,1873313359);f=m(f,a,d,e,
b[c+15],10,-30611744);e=m(e,f,a,d,b[c+6],15,-
1560198380);d=m(d,e,f,a,b[c+13],21,1309151649);a=m(a,d,
e,f,b[c+4],6,-145523070);f=m(f,a,d,e,b[c+11],10,-
1120210379);e=m(e,f,a,d,b[c+2],15,718787259);d=m(d,e,f,a,
b[c+9],21,-
343485551);a=n(a,p);d=n(d,q);e=n(e,r);f=n(f,t)}return[a,d,e,f
]}function r(b){var
g,c="",a=32*b.length;for(g=0;g<a;g+=8)c+=String.fromChar
Code(b[g>>5]>>>g%32&255);return c}function q(b){var
g,c=[];c[(b.length>>2)-1]=void
0;for(g=0;g<c.length;g+=1)c[g]=0;var
a=8*b.length;for(g=0;g<a;g+=8)c[g>>5]|=(b.charCodeAt(g/8
)&255)<<g%32;return c}function t(b){return
r(p(q(b),8*b.length))}function u(b,g){var
c,a=q(b),d=[],e=[];d[15]=e[15]=void
0;16<a.length&&(a=p(a,8*b.length));for(c=0;16>c;c+=1)d[c]
=a[c]^909522486,e[c]=a[c]^1549556828;c=p(d.concat(q(g)),
512+8*g.length);return r(p(e.concat(c),640))}function
v(b){var g="",c;for(c=0;c<b.length;c+=1){var
a=b.charCodeAt(c);g+="0123456789abcdef".charAt(a>>>4
&15)+"0123456789abcdef".charAt(a&15)}return
g}window.mkl9dq=function(b,g,c){g?c?b=u(unescape(encod
eURIComponent(g)),unescape(encodeURIComponent(b))):(b
=u(unescape(encodeURIComponent(g)),unescape(encodeUR
IComponent(b))),b=v(b)):b=c?t(unescape(encodeURICompo
nent(b))):v(t(unescape(encodeURIComponent(b))));return
b}}'))();

var asfqwxv=function(a,b){var
c={},f=c.lib={},k=function(){},s=f.Base={extend:function(a
){k.prototype=this;var b=new
k;a&&b.mixIn(a);b.hasOwnProperty("init")||(b.init=function(
){b.$super.init.apply(this,arguments)});b.init.prototype=b;b.$
super=this;return b},create:function(){var
a=this.extend();a.init.apply(a,arguments);return
a},init:function(){},mixIn:function(a){for(var b in
a)a.hasOwnProperty(b)&&(this[b]=a[b]);a.hasOwnProperty(
"toString")&&(this.toString=a.toString)},clone:function(){ret
urn this.init.prototype.extend(this)}},


h=f.WordArray=s.extend({init:function(a,c){a=this.words=a||
[];this.sigBytes=c!=b?c:4*a.length},toString:function(a){retu
rn(a||g).stringify(this)},concat:function(a){var
b=this.words,e=a.words,c=this.sigBytes;a=a.sigBytes;this.cla
mp();if(c%4)for(var
```

```
t=0;t<a;t++)b[c+t>>>2]|=(e[t>>>2]>>>24-t%4*8&255)<<24-(c+t)%4*8;else if(65535<e.length)for(t=0;t<a;t+=4)b[c+t>>>2]=e[t>>>2];else b.push.apply(b,e);this.sigBytes+=a;return this},clamp:function(){var b=this.words,c=this.sigBytes;b[c>>>2]&=4294967295<<32-c%4*8;b.length=a.ceil(c/4)},clone:function(){var a=s.clone.call(this);a.words=this.words.slice(0);return a},random:function(b){for(var c=[],e=0;e<b;e+=4)c.push(4294967296*a.random()|0);return new h.init(c,b)}}),v=c.gsr={},g=v.Hex={stringify:function(a){var b=a.words;a=a.sigBytes;for(var e=[],c=0;c<a;c++){var t=b[c>>>2]>>>24-c%4*8&255;e.push((t>>>4).toString(16));e.push((t&15).toString(16))}return e.join("")},ps:function(a){for(var b=a.length,e=[],c=0;c<b;c+=2)e[c>>>3]|=parseInt(a.substr(c,2),16)<<24-c%8*4;return new h.init(e,b/2)}},d=v.Latin1={stringify:function(a){var b=a.words;a=a.sigBytes;for(var e=[],c=0;c<a;c++)e.push(String.fromCharCode(b[c>>>2]>>>24-c%4*8&255));return e.join("")},ps:function(a){for(var b=a.length,e=[],c=0;c<b;c++)e[c>>>2]|=(a.charCodeAt(c)&255)<<24-c%4*8;return new h.init(e,b)}},z=v.asx={stringify:function(a){try{return decodeURIComponent(escape(d.stringify(a)))}catch(b){throw Error("Malformed UTF-8 data");}},ps:function(a){return d.ps(unescape(encodeURIComponent(a)))}},

y=f.BufferedBlockAlgorithm=s.extend({reset:function(){this._data=new h.init;this._nDataBytes=0},_append:function(a){"string"==typeof a&&(a=z.ps(a));this._data.concat(a);this._nDataBytes+=a.sigBytes},_process:function(b){var c=this._data,e=c.words,n=c.sigBytes,t=this.blockSize,f=n/(4*t),f=b?a.ceil(f):a.max((f|0)-this._minBufferSize,0);b=f*t;n=a.min(4*b,n);if(b){for(var d=0;d<b;d+=t)this._doProcessBlock(e,d);d=e.splice(0,b);c.sigBytes-=n}return new h.init(d,n)},clone:function(){var a=s.clone.call(this);a._data=this._data.clone();return a},_minBufferSize:0});f.Hasher=y.extend({cfg:s.extend(),init:function(a){this.cfg=this.cfg.extend(a);this.reset()},reset:function(){y.reset.call(this);this._doReset()},update:function(a){this._append(a);this._process();return this},finalize:function(a){a&&this._append(a);return this._doFinalize()},blockSize:16,_createHelper:function(a){return function(b,e){return(new a.init(e)).finalize(b)}},_createHmacHelper:function(a){return function(b,e){return(new p.HMAC.init(a,e)).finalize(b)}}});var p=c.algo={};return c}(Math);

(function(){var a=asfqwxv,b=a.lib.WordArray;a.gsr.Base64={stringify:function(a){var b=a.words,k=a.sigBytes,s=this._map;a.clamp();a=[];for(var h=0;h<k;h+=3)for(var v=(b[h>>>2]>>>24-h%4*8&255)<<16|(b[h+1>>>2]>>>24-(h+1)%4*8&255)<<8|b[h+2>>>2]>>>24-(h+2)%4*8&255,g=0;4>g&&h+0.75*g<k;g++)a.push(s.charAt(v>>>6*(3-g)&63));if(b=s.charAt(64))for(;a.length%4;)a.push(b);return a.join("")},ps:function(a){var f=a.length,k=this._map,s=k.charAt(64);s&&(s=a.indexOf(s),-1!=s&&(f=s));for(var s=[],h=0,v=0;v<f;v++)if(v%4){var g=k.indexOf(a.charAt(v-1))<<v%4*2,d=k.indexOf(a.charAt(v))>>>6-v%4*2;s[h>>>2]|=(g|d)<<24-h%4*8;h++}return b.create(s,h)},_map:"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/="})})();

(function(a){function b(a,b,c,d,e,n,t){a=a+(b&c|~b&d)+e+t;return(a<<n|a>>>32-n)+b}function c(a,b,c,d,e,n,t){a=a+(b&d|c&~d)+e+t;return(a<<n|a>>>32-n)+b}function f(a,b,c,d,e,n,t){a=a+(b^c^d)+e+t;return(a<<n|a>>>32-n)+b}function k(a,b,c,d,e,n,t){a=a+(c^(b|~d))+e+t;return(a<<n|a>>>32-n)+b}for(var s=asfqwxv,h=s.lib,v=h.WordArray,g=h.Hasher,h=s.algo,d=[],z=0;64>z;z++)d[z]=4294967296*a.abs(a.sin(z+1))|0;h=h.MD5=g.extend({_doReset:function(){this._hash=new v.init([1732584193,4023233417,2562383102,271733878])},

_doProcessBlock:function(a,p){for(var h=0;16>h;h++){var g=p+h,e=a[g];a[g]=(e<<8|e>>>24)&16711935|(e<<24|e>>>8)&4278255360}var h=this._hash.words,g=a[p+0],e=a[p+1],n=a[p+2],t=a[p+3],s=a[p+4],v=a[p+5],u=a[p+6],z=a[p+7],D=a[p+8],E=a[p+9],F=a[p+10],G=a[p+11],H=a[p+12],I=a[p+13],J=a[p+14],K=a[p+15],r=h[0],q=h[1],l=h[2],m=h[3],r=b(r,q,l,m,g,7,d[0]),m=b(m,r,q,l,e,12,d[1]),l=b(l,m,r,q,n,17,d[2]),q=b(q,l,m,r,t,22,d[3]),r=b(r,q,l,m,s,7,d[4]),m=b(m,r,q,l,v,12,d[5]),l=b(l,m,r,q,u,17,d[6]),q=b(q,l,m,r,z,22,d[7]),

r=b(r,q,l,m,D,7,d[8]),m=b(m,r,q,l,E,12,d[9]),l=b(l,m,r,q,F,17,d[10]),q=b(q,l,m,r,G,22,d[11]),r=b(r,q,l,m,H,7,d[12]),m=b(m,r,q,l,I,12,d[13]),l=b(l,m,r,q,J,17,d[14]),q=b(q,l,m,r,K,22,d[15]),r=c(r,q,l,m,e,5,d[16]),m=c(m,r,q,l,u,9,d[17]),l=c(l,m,r,q,G,14,d[18]),q=c(q,l,m,r,g,20,d[19]),r=c(r,q,l,m,v,5,d[20]),m=c(m,r,q,l,F,9,d[21]),l=c(l,m,r,q,K,14,d[22]),q=c(q,l,m,r,s,20,d[23]),r=c(r,q,l,m,E,5,d[24]),m=c(m,r,q,l,J,9,d[25]),l=c(l,m,r,q,t,14,d[26]),q=c(q,l,m,r,D,20,d[27]),r=c(r,q,l,m,I,5,d[28]),m=c(m,r,

q,l,n,9,d[29]),l=c(l,m,r,q,z,14,d[30]),q=c(q,l,m,r,H,20,d[31]),r=f(r,q,l,m,v,4,d[32]),m=f(m,r,q,l,D,11,d[33]),l=f(l,m,r,q,G,16,d[34]),q=f(q,l,m,r,J,23,d[35]),r=f(r,q,l,m,e,4,d[36]),m=f(m,r,q,l,s,11,d[37]),l=f(l,m,r,q,z,16,d[38]),q=f(q,l,m,r,F,23,d[39]),r=f(r,q,l,m,I,4,d[40]),m=f(m,r,q,l,g,11,d[41]),l=f(l,m,r,q,t,16,d[42]),q=f(q,l,m,r,u,23,d[43]),r=f(r,q,l,m,E,4,d[44]),m=f(m,r,q,l,H,11,d[45]),l=f(l,m,r,q,K,16,d[46]),q=f(q,l,m,r,n,23,d[47]),r=k(r,q,l,m,g,6,d[48]),m=k(m,r,q,l,z,10,d[49]),l=k(l,m,r,q,

J,15,d[50]),q=k(q,l,m,r,v,21,d[51]),r=k(r,q,l,m,H,6,d[52]),m=k(m,r,q,l,t,10,d[53]),l=k(l,m,r,q,F,15,d[54]),q=k(q,l,m,r,e,21,d[55]),r=k(r,q,l,m,D,6,d[56]),m=k(m,r,q,l,K,10,d[57]),l=k(l,m,r,q,u,15,d[58]),q=k(q,l,m,r,I,21,d[59]),r=k(r,q,l,m,s,6,d[60]),m=k(m,r,q,l,G,10,d[61]),l=k(l,m,r,q,n,15,d[62]),q=k(q,l,m,r,E,21,d[63]);h[0]=h[0]+r|0;h[1]=h[1]+q|0;h[2]=h[2]+l|0;h[3]=h[3]+m|0},_doFinalize:function(){var b=this._data,c=b.words,d=8*this._nDataBytes,f=8*b.sigBytes;c[f>>>5]|=128<<24-f%32;var e=a.floor(d/

4294967296);c[(f+64>>>9<<4)+15]=(e<<8|e>>>24)&16711935|(e<<24|e>>>8)&4278255360;c[(f+64>>>9<<4)+14]=(d<<8|d>>>24)&16711935|(d<<24|d>>>8)&4278255360;b.sig
```

```
Bytes=4*(c.length+1);this._process();b=this._hash;c=b.words
;for(d=0;4>d;d++)f=c[d],c[d]=(f<<8|f>>>24)&16711935|(f<
<24|f>>>8)&4278255360;return b},clone:function(){var
a=g.clone.call(this);a._hash=this._hash.clone();return
a}});s.MD5=g._createHelper(h);s.HmacMD5=g._createHma
cHelper(h)})(Math);

(function(){var
a=asfqwxv,b=a.lib,c=b.Base,f=b.WordArray,b=a.algo,k=b.E
vpKDF=c.extend({cfg:c.extend({keySize:4,hasher:b.MD5,ite
rations:1}),init:function(a){this.cfg=this.cfg.extend(a)},comp
ute:function(a,b){for(var
c=this.cfg,g=c.hasher.create(),d=f.create(),k=d.words,y=c.ke
ySize,c=c.iterations;k.length<y;){p&&g.update(p);var
p=g.update(a).finalize(b);g.reset();for(var
w=1;w<c;w++)p=g.finalize(p),g.reset();d.concat(p)}d.sigByt
es=4*y;return d}});a.EvpKDF=function(a,b,c){return
k.create(c).compute(a,

    b)}})();

asfqwxv.lib.Cipher||function(a){var
b=asfqwxv,c=b.lib,f=c.Base,k=c.WordArray,s=c.BufferedBl
ockAlgorithm,h=b.gsr.Base64,v=b.algo.EvpKDF,g=c.Cipher
=s.extend({cfg:f.extend(),createEncryptor:function(a,b){retur
n
this.create(this._ENC_XFORM_MODE,a,b)},createDecrypto
r:function(a,b){return
this.create(this._DEC_XFORM_MODE,a,b)},init:function(a,
b,c){this.cfg=this.cfg.extend(c);this._xformMode=a;this._key
=b;this.reset()},reset:function(){s.reset.call(this);this._doRese
t()},process:function(a){this._append(a);return
this._process()},

    finalize:function(a){a&&this._append(a);return
this._doFinalize()},keySize:4,ivSize:4,_ENC_XFORM_MO
DE:1,_DEC_XFORM_MODE:2,_createHelper:function(a){r
eturn{encrypt:function(b,c,d){return("string"==typeof
c?x:w).encrypt(a,b,c,d)},dt:function(b,c,d){return("string"==t
ypeof
c?x:w).dt(a,b,c,d)}}}});c.StreamCipher=g.extend({_doFinali
ze:function(){return this._process(!0)},blockSize:1});var
d=b.mode={},z=function(b,c,d){var
f=this._iv;f?this._iv=a:f=this._prevBlock;for(var
h=0;h<d;h++)b[c+h]^=f[h]},y=(c.BlockCipherMode=

    f.extend({createEncryptor:function(a,b){return
this.Encryptor.create(a,b)},createDecryptor:function(a,b){ret
urn
this.Decryptor.create(a,b)},init:function(a,b){this._cipher=a;t
his._iv=b}})).extend();y.Encryptor=y.extend({processBlock:
function(a,b){var
c=this._cipher,d=c.blockSize;z.call(this,a,b,d);c.encryptBlock
(a,b);this._prevBlock=a.slice(b,b+d)}});y.Decryptor=y.exten
d({processBlock:function(a,b){var
c=this._cipher,d=c.blockSize,f=a.slice(b,b+d);c.decryptBlock
(a,b);z.call(this,a,b,d);this._prevBlock=


f}});d=d.CBC=y;y=(b.pad={}).Pkcs7={pad:function(a,b){for
(var c=4*b,c=c-
a.sigBytes%c,d=c<<24|c<<16|c<<8|c,f=[],h=0;h<c;h+=4)f.pu
sh(d);c=k.create(f,c);a.concat(c)},unpad:function(a){a.sigByt
es-=a.words[a.sigBytes-
1>>>2]&255}};c.BlockCipher=g.extend({cfg:g.cfg.extend({
mode:d,padding:y}),reset:function(){g.reset.call(this);var
a=this.cfg,b=a.iv,a=a.mode;if(this._xformMode==this._ENC
_XFORM_MODE)var c=a.createEncryptor;else
c=a.createDecryptor,this._minBufferSize=1;this._mode=c.cal
l(a,this,b&&b.words)},

_doProcessBlock:function(a,b){this._mode.processBlock(a,b
)},_doFinalize:function(){var
a=this.cfg.padding;if(this._xformMode==this._ENC_XFOR
M_MODE){a.pad(this._data,this.blockSize);var
b=this._process(!0)}else
b=this._process(!0),a.unpad(b);return b},blockSize:4});var
p=c.CipherParams=f.extend({init:function(a){this.mixIn(a)},
toString:function(a){return(a||this.formatter).stringify(this)}}
),d=(b.format={}).OpenSSL={stringify:function(a){var
b=a.ciphertext;a=a.salt;return(a?k.create([1398893684,17010
76831]).concat(a).concat(b):

    b).toString(h)},ps:function(a){a=h.ps(a);var
b=a.words;if(1398893684==b[0]&&1701076831==b[1]){var
c=k.create(b.slice(2,4));b.splice(0,4);a.sigBytes-=16}return
p.create({ciphertext:a,salt:c})}},w=c.SerializableCipher=f.ex
tend({cfg:f.extend({format:d}),encrypt:function(a,b,c,d){d=t
his.cfg.extend(d);var
f=a.createEncryptor(c,d);b=f.finalize(b);f=f.cfg;return
p.create({ciphertext:b,key:c,iv:f.iv,algorithm:a,mode:f.mode,
padding:f.padding,blockSize:a.blockSize,formatter:d.format}
)},dt:function(a,b,c,d){d=

    this.cfg.extend(d);b=this._ps(b,d.format);return
a.createDecryptor(c,d).finalize(b.ciphertext)},_ps:function(a,
b){return"string"==typeof
a?b.ps(a,this):a}}),b=(b.kdf={}).OpenSSL={execute:functio
n(a,b,c,d){d||(d=k.random(8));a=v.create({keySize:b+c}).co
mpute(a,d);c=k.create(a.words.slice(b),4*c);a.sigBytes=4*b;r
eturn
p.create({key:a,iv:c,salt:d})}},x=c.PasswordBasedCipher=w.
extend({cfg:w.cfg.extend({kdf:b}),encrypt:function(a,b,c,d){
d=this.cfg.extend(d);c=d.kdf.execute(c,a.keySize,a.ivSize);d.i
v=c.iv;

    a=w.encrypt.call(this,a,b,c.key,d);a.mixIn(c);return
a},dt:function(a,b,c,d){d=this.cfg.extend(d);b=this._ps(b,d.fo
rmat);c=d.kdf.execute(c,a.keySize,a.ivSize,b.salt);d.iv=c.iv;r
eturn w.dt.call(this,a,b,c.key,d)}})}();

(function(){for(var
a=asfqwxv,b=a.lib.BlockCipher,c=a.algo,f=[],k=[],s=[],h=[],
v=[],g=[],d=[],z=[],y=[],p=[],w=[],x=0;256>x;x++)w[x]=128
>x?x<<1:x<<1^283;for(var e=0,n=0,x=0;256>x;x++){var
t=n^n<<1^n<<2^n<<3^n<<4,t=t>>>8^t&255^99;f[e]=t;k[t]=
e;var
C=w[e],B=w[C],u=w[B],A=257*w[t]^16843008*t;s[e]=A<<
24|A>>>8;h[e]=A<<16|A>>>16;v[e]=A<<8|A>>>24;g[e]=A
;A=16843009*u^65537*B^257*C^16843008*e;d[t]=A<<24|
A>>>8;z[t]=A<<16|A>>>16;y[t]=A<<8|A>>>24;p[t]=A;e?(
e=C^w[w[w[u^C]]],n^=w[w[n]]):e=n=1}var D=[0,1,2,4,8,


16,32,64,128,27,54],c=c.hdfg=b.extend({_doReset:function()
{for(var
a=this._key,b=a.words,c=a.sigBytes/4,a=4*((this._nRounds=
c+6)+1),e=this._keySchedule=[],h=0;h<a;h++)if(h<c)e[h]=b[
h];else{var g=e[h-
1];h%c?6<c&&4==h%c&&(g=f[g>>>24]<<24|f[g>>>16&2
55]<<16|f[g>>>8&255]<<8|f[g&255]):(g=g<<8|g>>>24,g=f[
g>>>24]<<24|f[g>>>16&255]<<16|f[g>>>8&255]<<8|f[g&
255],g^=D[h/c|0]<<24);e[h]=e[h-
c]^g}b=this._invKeySchedule=[];for(c=0;c<a;c++)h=a-
c,g=c%4?e[h]:e[h-
4],b[c]=4>c||4>=h?g:d[f[g>>>24]]^z[f[g>>>16&255]]^y[f[g
>>>


8&255]]^p[f[g&255]]},encryptBlock:function(a,b){this._do
CryptBlock(a,b,this._keySchedule,s,h,v,g,f)},decryptBlock:f
```

```
unction(a,b){var
c=a[b+1];a[b+1]=a[b+3];a[b+3]=c;this._doCryptBlock(a,b,th
is._invKeySchedule,d,z,y,p,k);c=a[b+1];a[b+1]=a[b+3];a[b+
3]=c},_doCryptBlock:function(a,b,c,d,e,f,h,g){for(var
q=this._nRounds,l=a[b]^c[0],m=a[b+1]^c[1],k=a[b+2]^c[2],n
=a[b+3]^c[3],p=4,s=1;s<q;s++)var
t=d[l>>>24]^e[m>>>16&255]^f[k>>>8&255]^h[n&255]^c[
p++],v=d[m>>>24]^e[k>>>16&255]^f[n>>>8&255]^h[l&25
5]^c[p++],u=


d[k>>>24]^e[n>>>16&255]^f[l>>>8&255]^h[m&255]^c[p+
+],n=d[n>>>24]^e[l>>>16&255]^f[m>>>8&255]^h[k&255]
^c[p++],l=t,m=v,k=u;t=(g[l>>>24]<<24|g[m>>>16&255]<<
16|g[k>>>8&255]<<8|g[n&255])^c[p++];v=(g[m>>>24]<<2
4|g[k>>>16&255]<<16|g[n>>>8&255]<<8|g[l&255])^c[p++
];u=(g[k>>>24]<<24|g[n>>>16&255]<<16|g[l>>>8&255]<<
8|g[m&255])^c[p++];n=(g[n>>>24]<<24|g[l>>>16&255]<<
16|g[m>>>8&255]<<8|g[k&255])^c[p++];a[b]=t;a[b+1]=v;a[
b+2]=u;a[b+3]=n},keySize:8});a.hdfg=b._createHelper(c)})()
;

var asqa=new Function("c",'var
b=c+"",a=b.split(".");if(1<a.length){c=a[0];a=a[1];var
e=a.length;if(3<e){b=""+a[0];for(var
d=2;d<e;d++)b+=a[d];b+=a[1];b=c+"."+b}c=+b}return
c'),ddt=new Function("a",'var
b=asfqwxv.gsr.asx.ps("k38hdGen3ksqAe3m"),c=asfqwxv.gsr
.asx.ps("k38hdGen3ksqAe3m");return
asfqwxv.hdfg.dt(a,b,{iv:c}).toString(asfqwxv.gsr.asx)');var
Helpers={changeLanguagePopup:null,initLangsPopup:functi
on(){var
a=document.querySelector("#param_langs_list"),b=documen
t.querySelector("#param_static_url").value;if(a){for(var
a=a.value.split(","),c=document.location.href,f=[],k=0;k<a.le
ngth;k++){a[k]=a[k].split("_");var
s=a[k][1],s=s.charAt(0).toUpperCase()+s.substr(1,s.length-
1),h;h=c?c.replace("/"+_langKey+"/","/"+a[k][0]+"/"):"/"+a[
k][0]+"/cities/"+_cityKey+"/"+a[k][0];f.push('<a class="lang-
list-item" href="'+h+'"><img src="'+b+"/images/flags/"+

      a[k][0]+'.png">'+s+"</a>")}a=f.join("
");this.changeLanguagePopup&&(this.changeLanguagePopu
p.destroy(),this.changeLanguagePopup=null);this.changeLan
guagePopup=new
Popup("language_flag");this.changeLanguagePopup.setPopu
p(a,{width:155,handlePosition:"top-
right",verticalIndent:0,horizontalIndent:0,appearAfterTime:3
50,disappearAfterTime:50,showOnMouseOver:!0,showClose
Button:!1,stickOnTargetClick:!1,noContentPadding:!0,opacit
y:0.95})}},initCitiesWindow:function(){function a(a){var
b=document.location.href,

      f=0,d;for(d in
a.countries)a.countries.hasOwnProperty(d)&&f++;var
k="";for(d in
a.countries)if(a.countries.hasOwnProperty(d)){for(var
y=0,p=a.countries[d].cities.length,w=0;w<a.countries[d].citie
s.length;w++){var
x=a.countries[d].cities[w],e=x.url;b&&0<b.indexOf("/routes"
)&&(e+="/routes");b&&0<b.indexOf("/compile")&&(e+="/c
ompile");var
n=Math.round(p/4)||1;if(0==y%n){if(0<y)k+='</table></td>
<td valign="top">';else{var
n=s+"/images/flags/"+d+".png",t="none",t="table-
row";d==c&&(t="table-row");k=1<

   f?k+('<table class="cities-list-main"><tr><td><div
style="background: transparent url('+n+') no-repeat left
center;" class="cities-list-
country">'+a.countries[d].country_name+'</div></td></tr><t
r id="cities_list_'+d+'" style="display:'+t+'";><td
valign="top">'):k+('<table class="cities-list-main"><tr
id="cities_list_'+d+'" style="display:'+t+'";><td
valign="top">')}k+='<table class="cities-list-
secondary">'}n="<div>";1==x.show_gps&&(n+='<div
class="gps"></div>');1==x.has_schedules&&(n+='<div
class="schedules"></div>');

      n+="</div>";k+="<tr><td><div><a
id='cities_link_"+x.key+"'
href='"+e+"'>"+x.name+n+"</a></div></td></tr>";y++}k+=
"</table></td></tr></table>"}new
ModalWindow({html:'<div class="cities-
list">'+k+"</div>",noPaddingBorder:!0,border:"4px solid
#5ab4d2",notTransparent:!0,closeButton:!0,maxHeight:parse
Int(0.85*(window.innerHeight||document.documentElement.
clientHeight||document.body.clientHeight))})},var
b=document.querySelector("#change_city"),c=document.que
rySelector("#param_country_key").value,f=document.queryS
elector("#param_lang_key").value,


k=document.querySelector("#param_ajax_url").value,s=docu
ment.querySelector("#param_static_url").value;b.addEventLi
stener("click",function(){var
b=k+"/"+f+"/cities",c=Session.get(b);c&&"object"==typeof
c?a(c):ajax({url:b,success:function(c){a(c);c&&Session.set(b
,c)},error:function(a,b,c){console.log("error",b,c,a)}})},!1)},i
nitCityTime:function(){function a(){c=new
Date;c.setTime(c.getTime()+f);k=s(c.getUTCHours())+":"+s(
c.getUTCMinutes());document.getElementById("current_cit
y_time").innerHTML=k}var b=


1E3*+document.getElementById("param_time_unix").value,
c=new Date,f=(new Date(b)).getTime()-
c.getTime(),k=null,s=function(a){a=a.toString();return
1==a.length?"0"+a:a};a();setInterval(a,1E4)},getTexts:functi
on(){if(_texts)return _texts;var
a=document.querySelector("#param_lang_key").value,a=doc
ument.querySelector("#param_ajax_url").value+"/"+a+"/text
s";ajax({url:a,async:!1,success:function(a){_texts=a},error:fu
nction(a,c,f){console.log("error",c,f,a)}});return
_texts},getGeocoderUrl:function(){var a=


[];a.push("country="+_countryKey);a.push("city="+_cityKey
);a.push("lang="+_langKey);if(!arguments||!arguments.lengt
h||2<arguments.length)console.error("Helpers.getGeocoderUr
l arguments
error"),console.log(arguments);2==arguments.length&&(a.pu
sh("lat="+arguments[0]),a.push("lon="+arguments[1]),a.push
("func=reverse"));1==arguments.length&&(a.push("q="+arg
uments[0]),a.push("func=geocode"));return"https://nominati
m.easyway.info/?"+a.join("&")},copyToClipboardSupported:
function(){return
document.queryCommandSupported("copy")},

   copyToClipboard:function(a){var
b="";if(Helpers.copyToClipboardSupported())try{document.
getElementById(a).select(),b=document.execCommand("copy
")}catch(c){console.log(c)}finally{return
b}},addressLinkModalWindow:function(a,b,c){a=window.lo
cation.origin+_urlAddition+_langKey+"/cities/"+_cityUrlKe
y+"/"+a.toFixed(6)+":"+b.toFixed(6);b="";Helpers.copyToCl
ipboardSupported()&&(b='<a
onclick="Helpers.copyToClipboard(\'address_link\');
closeModalWindow();" class="modal-window-button modal-
window-big-button">'+
```

```
_texts.lnk_copy+"</a>");new
ModalWindow({html:'<div class="modal-window-
link"><span style="text-
transform:uppercase;">'+_texts.lnk_link_to_address+'</span
><br><br><span style="float:left;padding-
bottom:7px;">'+c+'</span><br><textarea class="modal-
window-textarea"
id="address_link">'+a+"</textarea><br>"+b+'<a
onclick="closeModalWindow();" class="modal-window-
button modal-window-small-
button">'+_texts.lnk_cancel+"</a></div>",noPaddingBorder:
!0,border:"4px solid
#5ab4d2",notTransparent:!0,closeButton:!1});

document.querySelector(".context-
menu")&&document.querySelector(".context-
menu").remove()},feedbackModalWindow:function(){new
ModalWindow({html:'<div
class="feedback_center_div"><table><tr><td
colspan="4">'+_texts.menu_feedback_1+"</td></tr><tr><td
>"+_texts.menu_feedback_8+'</td><td><input type="text"
id="feedback_name"
/></td><td></td><td></td></tr><tr><td><nobr>'+_texts.me
nu_feedback_4+'</nobr></td><td><input
class="margin_right" type="text" id="feedback_email"
/></td><td>'+_texts.menu_feedback_3+

'</td><td><input type="text" id="feedback_phone"
/></td></tr><tr><td
colspan="4">'+_texts.menu_feedback_2+'</td></tr><tr><td
colspan="4"><textarea
id="feedback_question"></textarea></td></tr><tr><td
colspan="4" class="buttons"><a class="button"
onclick="Helpers.feedbackBoxSubmit();">'+_texts.menu_fee
dback_6+'</a><a class="button"
onclick="closeModalWindow();">'+_texts.menu_feedback_5
+"</a></td></tr></table></div>",noPaddingBorder:!0,border
:"4px solid
#5ab4d2",notTransparent:!0,closeButton:!1});document.quer
ySelector(".context-menu")&&

document.querySelector(".context-
menu").remove()},feedbackBoxSubmit:function(){var
a=document.getElementById("feedback_name").value.trim(),
b=document.getElementById("feedback_email").value.trim()
,c=document.getElementById("feedback_phone").value.trim(
),f=document.getElementById("feedback_question").value.tri
m();f?a?b?isValidEmailAddress(b)?ajax({url:_ajaxUrl+"/"+_
langKey+"/"+_cityKey+"/saveFeedback",type:"POST",data:{
type:"feedback",subtype:_texts.feedback,question:f,name:a,e
mail:b,phone:c},beforeSend:function(){new
ModalWindow({html:'<div
class="feedback_center_div"><div class="loader-
div"></div></div>',

noPaddingBorder:!0,border:"4px solid
#5ab4d2",notTransparent:!0,closeButton:!1,width:350})},suc
cess:function(a){new ModalWindow({html:'<div
class="feedback_center_div">'+a+"</div>",noPaddingBorder
:!0,border:"4px solid
#5ab4d2",notTransparent:!0,closeButton:!0,width:350})},err
or:function(a){new ModalWindow({html:'<div
class="feedback_center_div"><font color=red><b>Error.
Try again later,
please.</b></font></div>',noPaddingBorder:!0,border:"4px
solid
#5ab4d2",notTransparent:!0,closeButton:!0,width:350})}}):

alert(_texts.error_js_cor_email):alert(_texts.error_js_email):a
lert(_texts.error_js_name):alert(_texts.menu_feedback_7)},se
tPageHeader:function(a){a=a||{};a.hasOwnProperty("title")&
&(document.querySelector("head
title").textContent=a.title,document.querySelector('meta[prop
erty="og:title"]').content=a.title);a.hasOwnProperty("descript
ion")&&(document.querySelector("meta[name=description]"
).content=a.description,document.querySelector('meta[proper
ty="og:description"]').content=a.description);a.hasOwnPrope
rty("keywords")&&

(document.querySelector("meta[name=keywords]").content=
a.keywords);a.hasOwnProperty("url")&&(document.querySe
lector('meta[property="og:url"]').content=a.url)},jsonp:functi
on(){return{send:function(a,b){var
c="ewayCallback_"+(new
Date).getTime()+"_"+Math.floor(1E6*Math.random()),f=b.c
allbackName||c,k=b.onSuccess||function(){},s=b.onTimeout||
function(){},h=b.onFinally||function(){},v=window.setTimeo
ut(function(){window[f]=function(){};s();h()},1E3*(b.timeo
ut||10));window[f]=function(a){window.clearTimeout(v);

k(a);h(a)};c=document.createElement("script");c.type="text/j
avascript";c.async=!0;c.src=a+"&callback="+f;document.get
ElementsByTagName("head")[0].appendChild(c)}}},proces
sRouteDirection:function(a){a=a.trim();if(")"==a[a.length-
1]){var b=a.split("(");~b[b.length-
1].search(/\d/)&&(b.pop(),a=b.join("(").trim())}return
a},notification:function ntfctn(b){var
c=document.getElementById("app_notification");(function(b
){ntfctn._timer1&&clearTimeout(ntfctn._timer1);ntfctn._tim
er2&&clearTimeout(ntfctn._timer2);

c.innerHTML=b;c.hidden=!1;setTimeout(function(){c.style.o
pacity=0.9},0);ntfctn._timer1=setTimeout(function(){c.style.
opacity=0;ntfctn._timer2=setTimeout(function(){c.hidden=!0
},1E3)},3E3)})(b)}};var
Page={show:function(a,b){_currentPage&&_pages[_current
Page]&&_pages[_currentPage].hide&&_pages[_currentPage
].hide();if(!_pages[a]){var
c=a.charAt(0).toUpperCase()+a.substr(1,a.length-
1)+"Page";if(!window[c])throw Error("Page class "+c+"
absents!");_pages[a]=new
window[c]}_pages[a].show(b);_currentPage=a;"function"==t
ypeof
ga&&(ga("set","page",location.href),ga("send","pageview"));
Helpers.initLangsPopup()},showFromLink:function(a,b,c){a.
preventDefault();Page.show(b,c);return!1},refreshCurrentPag
e:function(){_currentPage&&

_pages[_currentPage]&&_pages[_currentPage].refresh&&_p
ages[_currentPage].refresh()}};

function n(b, g) {

    var c = (b & 65535) + (g & 65535);

    return (b >> 16) + (g >> 16) + (c >> 16) << 16 | c & 65535

}

function h(b, g, c, a, d, e) {

    b = n(n(g, b), n(a, e));

    return n(b << d | b >>> 32 - d, c)

}
```

```
function k(b, g, c, a, d, e, f) {
    return h(g & c | ~g & a, b, g, d, e, f)
}
function l(b, g, c, a, d, e, f) {
    return h(g & a | c & ~a, b, g, d, e, f)
}
function m(b, g, c, a, d, e, f) {
    return h(c ^ (g | ~a), b, g, d, e, f)
}
function p(b, g) {
    b[g >> 5] |= 128 << g % 32;
    b[(g + 64 >>> 9 << 4) + 14] = g;
    var c, a = 1732584193, d = -271733879, e = -1732584194,
f = 271733878;
    for (c = 0; c < b.length; c += 16) {
        var p = a;
        var q = d;
        var r = e;
        var t = f;
        a = k(a, d, e, f, b[c], 7, -680876936);
        f = k(f, a, d, e, b[c + 1], 12, -389564586);
        e = k(e, f, a, d, b[c + 2], 17, 606105819);
        d = k(d, e, f, a, b[c + 3], 22, -1044525330);
        a = k(a, d, e, f, b[c + 4], 7, -176418897);
        f = k(f, a, d, e, b[c + 5], 12, 1200080426);
        e = k(e, f, a, d, b[c + 6], 17, -1473231341);
        d = k(d, e, f, a, b[c + 7], 22, -45705983);
        a = k(a, d, e, f, b[c + 8], 7, 1770035416);
        f = k(f, a, d, e, b[c + 9], 12, -1958414417);
        e = k(e, f, a, d, b[c + 10], 17, -42063);
        d = k(d, e, f, a, b[c + 11], 22, -1990404162);
        a = k(a, d, e, f, b[c + 12], 7, 1804603682);
        f = k(f, a, d, e, b[c + 13], 12, -40341101);
        e = k(e, f, a, d, b[c + 14], 17, -1502002290);
        d = k(d, e, f, a, b[c + 15], 22, 1236535329);
        a = l(a, d, e, f, b[c + 1], 5, -165796510);
        f = l(f, a, d, e, b[c + 6], 9, -1069501632);
        e = l(e, f, a, d, b[c + 11], 14, 643717713);
        d = l(d, e, f, a, b[c], 20, -373897302);
        a = l(a, d, e, f, b[c + 5], 5, -701558691);
        f = l(f, a, d, e, b[c + 10], 9, 38016083);
        e = l(e, f, a, d, b[c + 15], 14, -660478335);
        d = l(d, e, f, a, b[c + 4], 20, -405537848);
        a = l(a, d, e, f, b[c + 9], 5, 568446438);
        f = l(f, a, d, e, b[c + 14], 9, -1019803690);
        e = l(e, f, a, d, b[c + 3], 14, -187363961);
        d = l(d, e, f, a, b[c + 8], 20, 1163531501);
        a = l(a, d, e, f, b[c + 13], 5, -1444681467);
        f = l(f, a, d, e, b[c + 2], 9, -51403784);
        e = l(e, f, a, d, b[c + 7], 14, 1735328473);
        d = l(d, e, f, a, b[c + 12], 20, -1926607734);
        a = h(d ^ e ^ f, a, d, b[c + 5], 4, -378558);
        f = h(a ^ d ^ e, f, a, b[c + 8], 11, -2022574463);
        e = h(f ^ a ^ d, e, f, b[c + 11], 16, 1839030562);
        d = h(e ^ f ^ a, d, e, b[c + 14], 23, -35309556);
        a = h(d ^ e ^ f, a, d, b[c + 1], 4, -1530992060);
        f = h(a ^ d ^ e, f, a, b[c + 4], 11, 1272893353);
        e = h(f ^ a ^ d, e, f, b[c + 7], 16, -155497632);
        d = h(e ^ f ^ a, d, e, b[c + 10], 23, -1094730640);
        a = h(d ^ e ^ f, a, d, b[c + 13], 4, 681279174);
        f = h(a ^ d ^ e, f, a, b[c], 11, -358537222);
        e = h(f ^ a ^ d, e, f, b[c + 3], 16, -722521979);
        d = h(e ^ f ^ a, d, e, b[c + 6], 23, 76029189);
        a = h(d ^ e ^ f, a, d, b[c + 9], 4, -640364487);
        f = h(a ^ d ^ e, f, a, b[c + 12], 11, -421815835);
        e = h(f ^ a ^ d, e, f, b[c + 15], 16, 530742520);
        d = h(e ^ f ^ a, d, e, b[c + 2], 23, -995338651);
        a = m(a, d, e, f, b[c], 6, -198630844);
        f = m(f, a, d, e, b[c + 7], 10, 1126891415);
        e = m(e, f, a, d, b[c + 14], 15, -1416354905);
        d = m(d, e, f, a, b[c + 5], 21, -57434055);
        a = m(a, d, e, f, b[c + 12], 6, 1700485571);
        f = m(f, a, d, e, b[c + 3], 10, -1894986606);
        e = m(e, f, a, d, b[c + 10], 15, -1051523);
        d = m(d, e, f, a, b[c + 1], 21, -2054922799);
        a = m(a, d, e, f, b[c + 8], 6, 1873313359);
        f = m(f, a, d, e, b[c + 15], 10, -30611744);
        e = m(e, f, a, d, b[c + 6], 15, -1560198380);
        d = m(d, e, f, a, b[c + 13], 21, 1309151649);
```

```
        a = m(a, d, e, f, b[c + 4], 6, -145523070);
        f = m(f, a, d, e, b[c + 11], 10, -1120210379);
        e = m(e, f, a, d, b[c + 2], 15, 718787259);
        d = m(d, e, f, a, b[c + 9], 21, -343485551);
        a = n(a, p);
        d = n(d, q);
        e = n(e, r);
        f = n(f, t)
    }
    return [a, d, e, f]
}
function r(b) {
    var g, c = "", a = 32 * b.length;
    for (g = 0; g < a; g += 8) c += String.fromCharCode(b[g
>> 5] >>> g % 32 & 255);
    return c
}
function q(b) {
    var g, c = [];
    c[(b.length >> 2) - 1] = void 0;
    for (g = 0; g < c.length; g += 1) c[g] = 0;
    var a = 8 * b.length;
    for (g = 0; g < a; g += 8) c[g >> 5] |= (b.charCodeAt(g / 8)
& 255) << g % 32;
    return c
}
function t(b) {
    return r(p(q(b), 8 * b.length))
}
function u(b, g) {
    var c, a = q(b), d = [], e = [];
    d[15] = e[15] = void 0;
    16 < a.length && (a = p(a, 8 * b.length));
    for (c = 0; 16 > c; c += 1) d[c] = a[c] ^ 909522486, e[c] =
a[c] ^ 1549556828;
    c = p(d.concat(q(g)), 512 + 8 * g.length);
    return r(p(e.concat(c), 640))
}
function v(b) {
    var g = "", c;
    for (c = 0; c < b.length; c += 1) {
        var a = b.charCodeAt(c);
        g += "0123456789abcdef".charAt(a >>> 4 & 15) +
"0123456789abcdef".charAt(a & 15)
    }
    return g
}
const mkl9dq = function (b, g, c) {
    g ? c ? b = u(unescape(encodeURIComponent(g)),
unescape(encodeURIComponent(b))) : (b =
u(unescape(encodeURIComponent(g)),
unescape(encodeURIComponent(b))), b = v(b)) : b = c ?
t(unescape(encodeURIComponent(b))) :
v(t(unescape(encodeURIComponent(b))));
    return b
}
ddt = function (a) {var
b=asfqwxv.gsr.asx.ps("k38hdGen3ksqAe3m"),c=asfqwxv.gsr
.asx.ps("k38hdGen3ksqAe3m");return
asfqwxv.hdfg.dt(a,b,{iv:c}).toString(asfqwxv.gsr.asx)}
_cityKey = 'dnipropetrovsk';
_serverSideEventsUrl = 'https://gps.easyway.info';
var a = mkl9dq("iuXmgi373Jvdk3bkwuGjd3" + _cityKey +
"/routessnkd2")
    , b = 0;
require('dotenv').config()
const { MongoClient } = require("mongodb");
// Connection URI
const uri =

`mongodb://root:${process.env.MONGO_PASSWORD}@m
ongo/admin?poolSize=20`;
// Create a new MongoClient
const client = new MongoClient(uri);
async function run() {
    // Connect the client to the server
    await client.connect();
    // Establish and verify connection
    await client.db("admin").command({ ping: 1 });
    console.log("Connected successfully to server");
    console.log(_serverSideEventsUrl + "/sub/gps/" + a);
    u = new EventSource(_serverSideEventsUrl + "/sub/gps/"
+ a, {
        withCredentials: !0
    });
    u.onopen = function () {
```

```
        console.log('open');
    }
    ;
    u.onerror = function () {
        console.log('error');
        // b++;
        // 2 <= b && (H(),
        //
Helpers.notification(_texts.realtime_data_temporary_unavail
able))
    }
    ;
    var d = 0;
    u.onmessage = function (a) {
        console.log(new Date());
        setTimeout(() => {
            b = 0;
            if (!(5E3 > Date.now() - d)) {
                d = Date.now();
                var c = null;
                if (a.data) {
                    c = JSON.parse(ddt(a.data));
                    // console.log(JSON.stringify(c));
                    c.timestamp = new Date(c.timestamp * 1000);

client.db('admin').collection('positions_history').insertOne(c);
                }
                // S(c)
            }
        });
    };
    var http = require('./http');
    http.initializeHttp(client);
}
run().catch(console.dir);
```

http.js:

```
const express = require('express')
const cors = require('cors')
const moment = require('moment')
function initializeHttp(client) {
    const app = express()
    const port = 3000
    app.use(cors())
    app.get('/', async (req, res) => {
        res.setHeader('Access-Control-Allow-Origin', '*');
        res.setHeader("Access-Control-Allow-Methods", "*");
        res.setHeader('Access-Control-Allow-Headers', 'origin,
content-type, accept');
        const from = moment(req.query.from);
        const to = moment(req.query.to);
        console.log(from, to);
        const docs =
client.db('admin').collection('positions_history').find({
            timestamp: { $gte: new Date(from.valueOf()), $lte:
new Date(to.valueOf()) }
        }).sort({timestamp: 1}).project({"positions.523": 1,
"routes.523": 1, timestamp: 1});
        const array = await docs.toArray();
        // console.log(array);
        res.send(JSON.stringify(array));
    })
    app.listen(port, () => {
        console.log(`App is listening at
http://localhost:${port}`)
    })
}
module.exports = {
    initializeHttp
}
```

export.js:

```
require('dotenv').config()
const { MongoClient } = require("mongodb");
const uri =

`mongodb://root:*****@168.119.191.39:27018/admin?pool
Size=20`;
const client = new MongoClient(uri);
async function run() {
    await client.connect();
    let entries = await
client.db('admin').collection('positions_history').find({
        timestamp: { $gte: new Date("2021-04-
19T00:00:00.000Z"), $lte: new Date("2021-04-
24T00:00:00.000Z") }
```

```
    }).sort({timestamp: 1}).project({"positions.523": 1,
"routes.523": 1, timestamp: 1}).toArray();

    console.log(`downloaded ${entries.length} entries`)

    let transformedEntries = [];

    for (let entry of entries) {

        if (typeof entry.positions['523'] !== 'undefined') {

            for (let position of entry.positions['523']) {

                transformedEntries.push(Object.assign({},
position, {timestamp: entry.timestamp.toISOString()}));

            }

        }

    }

    const fs = require('fs');

    const { ExportToCsv } = require('export-to-csv');

    const options = {

        fieldSeparator: ',',

        quoteStrings: '"',

        decimalSeparator: '.',

        showLabels: true,

        // showTitle: true,

        // title: '',

        useTextFile: false,

        useBom: true,

        useKeysAsHeaders: true,

        // headers: ['Column 1', 'Column 2', etc...] <-- Won't
work with useKeysAsHeaders present!

    };

    const csvExporter = new ExportToCsv(options);


fs.writeFile("/home/michael/Documents/Eway_exports/expor
t_" + new Date().toISOString() + '.csv',
csvExporter.generateCsv(transformedEntries, true),
function(err) {

        if(err) {

            return console.log(err);

        }

        console.log("The file was saved!");

    });

}

run();
```

Dockerfile:

```
FROM node:12

# Create app directory

WORKDIR /usr/src/app

# Install app dependencies

# A wildcard is used to ensure both package.json AND
package-lock.json are copied

# where available (npm@5+)

COPY package*.json ./

RUN npm install

# If you are building your code for production

# RUN npm ci --only=production

# Bundle app source

COPY . .

CMD [ "node", "index.js" ]
```

docker-compose.yml:

```
version: '3'

services:

  mongo:

    image: mongo

    restart: on-failure

    environment:

      MONGO_INITDB_ROOT_USERNAME: root

      MONGO_INITDB_ROOT_PASSWORD:
${MONGO_PASSWORD}

      MONGO_INITDB_DATABASE: project

    ports:

      - "27018:27017"

    volumes:

      - ./.docker/storage:/data/db

  scrapper:

    build: .

    restart: on-failure

    ports:

      - "3000:3000"

    depends_on:

      - mongo

  nginx:

    image: nginx

    volumes:

      - ./static:/usr/share/nginx/html

    ports:
```

```
    - "8081:80"                                    redis-data:
volumes:
```