Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

другий (магістерський)
(рівень вищої освіти)

на тему: *«Інтелектуальна система підтримки прийняття рішень вибору району проживання»*

Виконав: студент 6 курсу групи КН–61м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Томчук О. Є.
(прізвище та ініціали)

Керівник *Процик Ю. С.*
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Львів – 2021

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Крошний І. М.
"____" _____20__ року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Томчуку Остапові Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи *«Інтелектуальна система підтримки прийняття рішень вибору району проживання»*

керівник роботи *Процик Юрій Степанович, к.ф.-м.н.*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "*31*" *грудня* *2020 року* № *С-593*

2. Термін подання студентом роботи *10 грудня 2021 року*

3. Вихідні дані до роботи *Аналіз шляхів вирішення задачі, організаційна структура системи*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Стан проблемної області.
Розділ 2. Інформаційне забезпечення. Розділ 3. Математичне забезпечення.
Розділ 4. Програмне забезпечення. Розділ 5. Розроблення стартап-проекту.
Висновки. Список використаних джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
слайди для доповіді (підготовка матеріалу для доповіді загальним обсягом 10-12 слайдів)

6. Дата видачі завдання *18 грудня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Системний аналіз стану проблемної області	*18.12.2020 р.* *25.03.2021 р.*	
2.	Аналіз та попередня обробка статистичних даних	*26.03.2021 р.* *11.05.2021 р.*	
3.	Вибір і налаштування алгоритму кластеризації	*12.05.2021 р.* *28.07.2021 р.*	
4.	Розроблення програмного забезпечення системи вибору району проживання	*29.07.2021 р.* *12.10.2021 р.*	
5.	Тестування роботи системи	*13.10.2021 р.* *11.11.2021 р.*	
6.	Розроблення стартап-проекту	*12.11.2021 р.* *25.11.2021 р.*	
7.	Оформлення пояснювальної записки та здача на рецензування	*26.11.2021 р.* *10.12.2021 р.*	

Студент _____ Томчук О. Є.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Процик Ю. С.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 53 сторінки пояснювальної записки, 32 рисунки, 3 таблиці, 1 додаток, 14 джерел.

В даній роботі розроблено інтелектуальну систему підтримки прийняття рішень вибору району проживання на прикладі міста Окленда в Новій Зеландії.

Проведено кластеризацію районів міста на основі статистичних даних про злочини за останні 7 років. Як наслідок, виділено райони з підвищеним рівнем злочинності. Реалізовані засоби інтерактивної візуалізації наявної статистичної інформації та результатів кластеризації можуть бути використані при оцінці вартості нерухомості з урахуванням району розміщення.

Ключові слова:

СППР, кластеризація, візуалізація, карта, злочини, житловий район, Окленд, Python, Dash.

ABSTRACT

Diploma paper contains 53 pages of explanatory note, 32 pictures, 3 tables, 1 application, 14 used literary sources.

In this work, the intelligent decision support system for choosing a residential area on the example of Auckland in New Zealand.

Clustering of city areas on the basis of statistical data on crimes for the last 7 years is carried out. As a result, areas with high crime rates have been identified. Implemented tools of interactive visualization of available statistical information and clustering results can be used in estimating the value of real estate, taking into account the location.

Keywords:

DSS, clustering, visualization, map, crime, residential area, Auckland, Python, Dash.

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити інтелектуальну систему підтримки прийняття рішень вибору району проживання на основі інформації щодо рівня злочинності та інших статистичних даних у розрізі районів міста. Реалізувати кластеризацію районів за наявними ознаками та візуальне представлення отриманих результатів на карті. Передбачити можливість відображення статистичних даних для вибраного користувачем району. Інформація повинна бути представлена у зручній формі для її аналізу та динамічно оновлюватися при взаємодії з користувачем.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

СППР – Система підтримки прийняття рішень;

GeoJSON – Geographic JavaScript Object Notation,
формат зберігання географічних структур даних;

CRS – Coordinate Reference Systems,
система координат;

HTML – HyperText Markup Language,
мова розмітки гіпертекстових документів.

ВСТУП

На перший погляд здається, що при купівлі квартири головне знайти необхідні кошти для цього. Коли ж справа доходить до процесу вибору, з'ясовується, що є багато важливих чинників, які впливають на остаточне рішення. Серед них одне з найважливіших питань – вибір району. Враховувати треба не лише вартість квартир у різних частинах міста, але також спосіб життя та особливості конкретної локації.

В цій дипломній роботі розглядаються питання, пов'язані з розробленням інтелектуальної системи підтримки прийняття рішень (СППР) вибору району проживання. Існує кілька визначень інтелектуальних СППР, які, загалом, обертаються навколо одного й того самого функціоналу. Інтелектуальна СППР — це така система, яка асистує особам, що приймають рішення, у прийнятті цих самих рішень, використовуючи інструментарії дата майнінгу, моделювання та візуалізації. Вона має дружній інтерфейс користувача, є стійкою за якістю, інтерактивною та гнучкою за налаштуваннями.

Наведені міркування говорять про *актуальність* даної теми, особливо в епоху стрімкого зростання об'ємів інформації, що опрацьовуються людиною.

Об'єктом дослідження є процес розроблення інтелектуальних систем підтримки прийняття рішень.

Предмет дослідження – процес розроблення інтелектуальної системи підтримки прийняття рішень вибору району проживання.

Метою роботи є розроблення програмного та алгоритмічного забезпечення інтелектуальної системи підтримки прийняття рішень вибору району проживання на основі інформації щодо рівня злочинності та інших статистичних даних у розрізі районів міста.

Завдання, які потрібно вирішити:

- здійснити попередній аналіз та обробку даних;
- провести кластеризацію районів міста за наявними ознаками;
- оцінити якість кластеризації;

- розробити програмне забезпечення для візуалізації результатів кластеризації та іншої статистичної інформації на карті міста;
- реалізувати можливість відображення зведених статистичних даних для вибраного користувачем району.

Наукова новизна. Проведено кластеризацію районів міста Окленда в Новій Зеландії на основі статистичних даних про злочинність за останні декілька років. Як результат, виділено райони, що суттєво відрізняються від загальної маси підвищеним рівнем злочинності.

Практичне значення. Реалізовані засоби інтерактивної візуалізації наявної статистичної інформації та результатів кластеризації можуть бути використані при оцінці вартості нерухомості з урахуванням району розміщення, особами, що бажають придбати житло, а також ріелторами.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Види та можливості сучасних систем підтримки прийняття рішень

Під сучасними СППР розуміють спеціальне програмне забезпечення, що дозволяє менеджерам середньої та вищої ланки приймати виважені обґрунтовані рішення. Така програма функціонує як база даних з функціями їх накопичення, аналізу, формування зручних для роботи звітів. Вона дозволяє визначитися з вибором навіть за швидко змінюваних обставин і при високому відсотку невизначеності.

У світовій практиці такі інформаційно-програмні продукти отримали назву DSS-систем (Decision Support Systems). Вони широко використовуються для організації ефективного управління бізнесом та полегшують роботу менеджерів зі збирання та аналізу інформації, виявлення проблем та прийняття вірних рішень.

Залежно від способу впливу на процес прийняття рішення розрізняють пасивні, активні і комбіновані DSS-системи. Перші надають лише інформацію для прийняття рішень, другі пропонують альтернативні готові варіанти, треті передбачають тісну роботу в контакті: менеджер може коригувати запропоноване системою рішення та узгоджувати до набуття ним оптимальної форми.

Розрізняють 5 видів комп’ютерних СППР:

- Комунікативні. Орієнтовані на одночасну роботу кількох спеціалістів над одним спільним завданням.
- Інформаційні. Зосереджені на зборі та обробці даних, переважно аналізі часових рядів, функціонують, як СУБД у межах однієї компанії.
- Документальні. Призначені для обробки та аналізу документів різного формату зі структурованими та не структурованими даними.
- Інтелектуальні. Містять дані про рішення аналогічних завдань, норми та правила, на підставі яких вони приймалися, пропонують готові алгоритми, виходячи з накопиченого досвіду.
- Модельовані. Підбирають моделі бізнес-процесів за заданими умовами – статистичні, фінансові, аналітичні.

Основу будь-якої з вищезгаданих систем прийняття рішень становить база даних, її предметна область та інтерфейс користувача.

До прийняття рішення СППР «підштовхує» за допомогою наступних аналітичних методів:

- регресійний та дисперсійний аналіз;
- багатомірний та дискримінантний аналіз;
- аналіз виживання та прогнозування часових рядів;
- аналіз категоріальних даних;
- структурний, просторовий та факторний аналіз;
- систематизація запитів та засобів пошуку даних.

Реалізація функцій багатовимірного аналізу дозволяє спостерігати дані в динаміці, у різних напрямках та вимірах. За допомогою інструментів запитів формулюється звернення до баз даних, яке ідентифікується за змістом та зразком. Пошукові інструменти наділяють програмне забезпечення можливостями оперативного пошуку даних за зразками, моделями та визначенням інформаційних залежностей. Звучить все це складно, але на практиці набуває простої та доступної форму – потрібно тільки правильно та послідовно виконувати команди системи та дотримуватися інструкцій.

Сучасні види систем прийняття рішень наділені такими можливостями:

- формування статистики та її перевірка;
- складання трендових прогнозів;
- планування та контроль якості;
- фінансовий аналіз та прогнозування;
- аналіз ризиків та прихованих закономірностей;
- відстеження поведінки людини та формування клієнтських груп;
- управління активами;
- перерозподіл завдань між працівниками.

Впровадження СППР дозволяє керівникам середньої ланки та топ-менеджерам вирішувати такі завдання, як визначення стратегічних завдань бізнесу, управління

проектами, активами, витратами, ризиками, виробничими потужностями, змінами, взаємовідносинами з контрагентами.

1.2 Аналіз бібліотек Python для візуалізації даних

Кількість бібліотек і рішень для візуалізації в Python вражає: Matplotlib, Seaborn, Plotly, Altair, Folium – Leaflet.

Але яку з цих бібліотек краще обрати для візуалізації DataFrame? Деякі бібліотеки мають більше переваг для використання у деяких конкретних випадках. Розберемося у функціональності кожної з них та наведемо їх плюси і мінуси, приділяючи особливу увагу декільком показникам:

Інтерактивність

Чи потрібно, щоб візуалізація була інтерактивною?

Візуалізація деяких бібліотек, таких як Matplotlib, є простим статичним зображенням, що добре підходить для пояснення концепцій (у документі, на слайдах або в презентації).

Така бібліотека як Plotly дозволяє створювати інтерактивні графіки, які користувачі можуть вивчати, взаємодіючи з ними.

Синтаксис та гнучкість

Чим відрізняється синтаксис кожної бібліотеки? Бібліотеки низького рівня, такі як Matplotlib, дозволяють робити все, що завгодно, але за рахунок складнішого API. Деякі бібліотеки, такі як Altair, дуже декларативні, що спрощує побудову графіків за даними.

Тип даних та візуалізації

Чи є потреба у побудові нестандартного графіка, наприклад, географічного, що включає великий набір даних або графіка, який підтримується лише певною бібліотекою?

Matplotlib є найпопулярнішою бібліотекою Python для візуалізації даних. Всі, хто цікавиться Data Science, напевно, хоч раз зіштовхувалися з Matplotlib.

Плюси:

1. Чітко відображені властивості даних

При аналізі даних можливість швидко переглянути розподіл може бути дуже корисною. Наприклад, якщо потрібно переглянути розподіл топ 100 користувачів з найбільшою кількістю передплатників, зазвичай Matplotlib буде цілком достатньо:

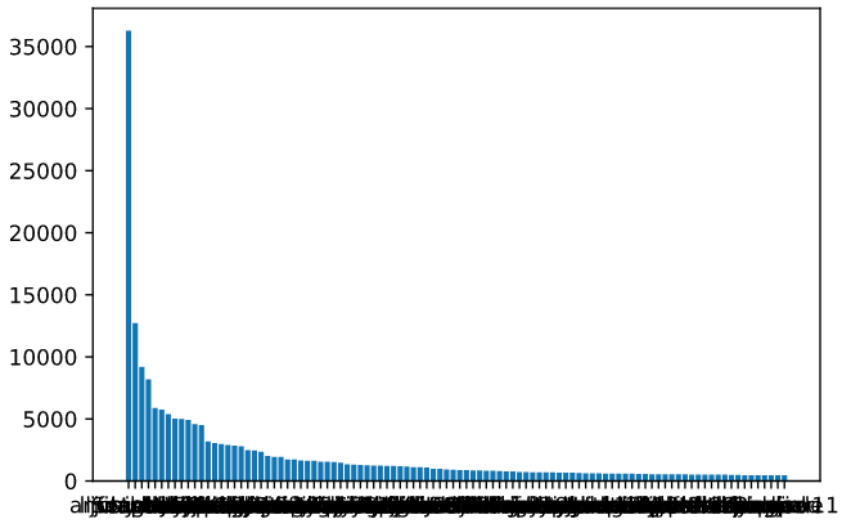

Рисунок 1.1 – Стовпчикова діаграма з Matplotlib

Мінуси:

Matplotlib може створити будь-який графік, але з ним важко побудувати або підігнати складні графіки так, щоб вони були презентабельними. Matplotlib має надзвичайно низькорівневий інтерфейс.

Seaborn – це бібліотека Python для візуалізації даних, побудована на базі Matplotlib. Вона більш високорівнева, що полегшує її використання.

Плюси:

1. Менше коду

Надає інтерфейс вищого рівня для побудови схожих графіків. Іншими словами, Seaborn зазвичай будує графіки, аналогічні Matplotlib, але з меншою кількістю коду та більш красивим дизайном.

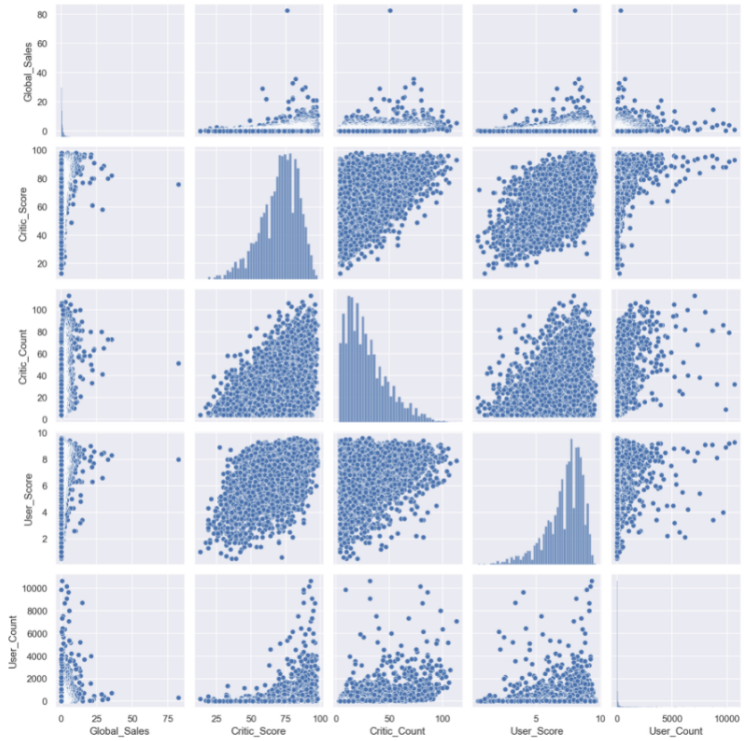

Рисунок 1.2 – Побудова «складних» візуалізацій з Seaborn (*pair plot*)

2. Робить стандартні графіки красивішими

Багато людей вибирають Seaborn для створення широко використовуваних графіків, таких як стовпчикові діаграми, коробчасті діаграми, гістограми і т. д., але не тільки тому, що це вимагатиме менше коду, вони ще й візуально приємніші.

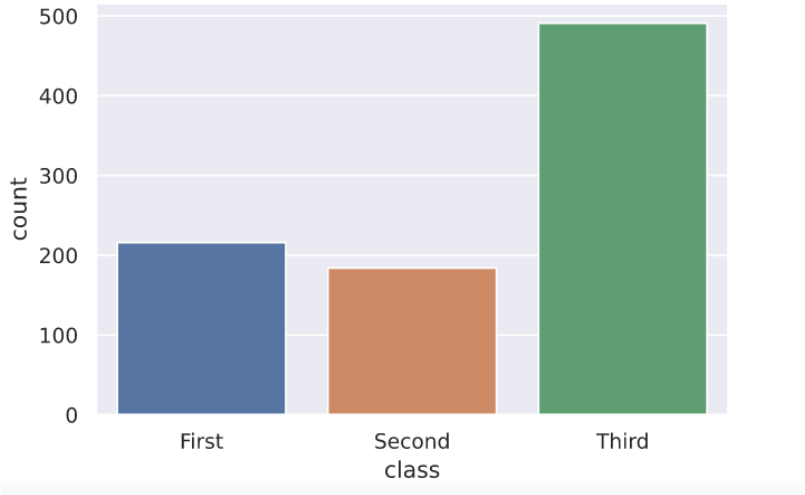

Рисунок 1.3 – Стовпчикова діаграма з Seaborn

Як видно з прикладу вище, кольори виглядають краще, ніж кольори за замовчуванням у Matplotlib.

Мінуси:

Seaborn більш обмежений і немає такої широкої колекції графіків, як Matplotlib.

Plotly – це open-source бібліотека, яка дозволяє будувати інтерактивні графіки без необхідності вникати в javascript код.

Плюси:

1. Схожий на R

Шанувальникам R, яким не вистачає його функціоналу при переході на Python, Plotly дасть таку саму якість графіків.

З версією **Plotly Express** можна легко і швидко створювати відмінні графіки одним рядком у Python:

```
import plotly.express as px
long_df = px.data.medals_long()
fig = px.bar(long_df, x="nation", y="count", color="medal", title="Long-Form Input")
fig.show()
```

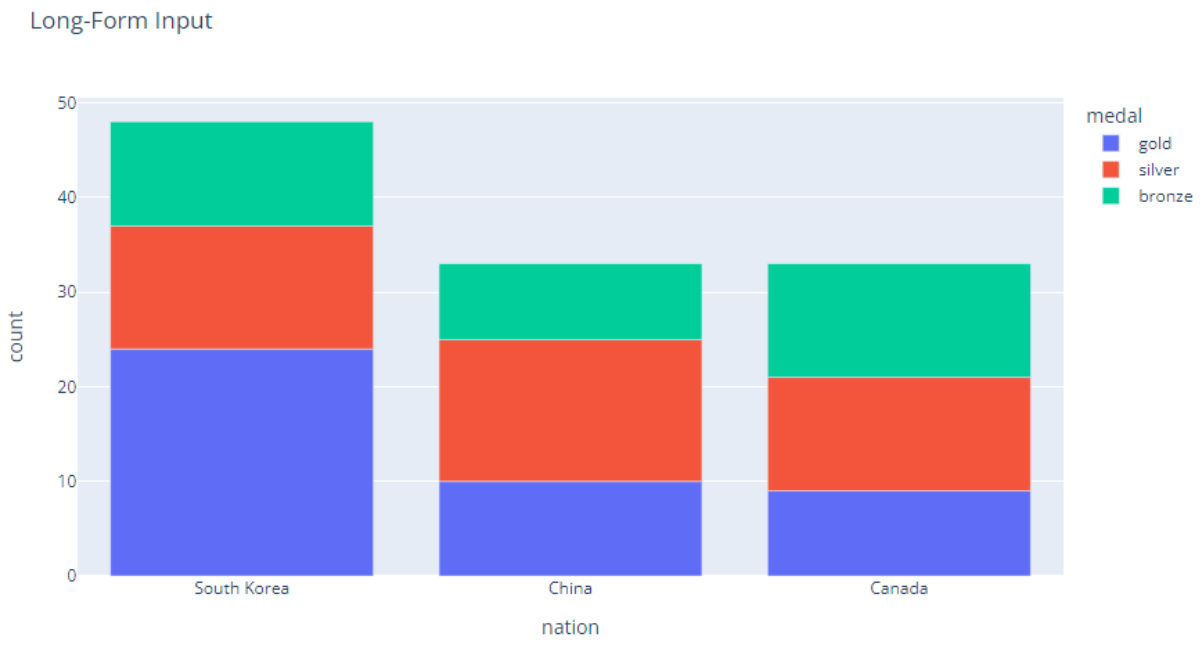

Рисунок 1.4 – Візуалізація з Plotly

2. Простота створення інтерактивних графіків

Plotly також полегшує створення інтерактивних графіків. Інтерактивні графіки не лише красиво виглядають, а й дозволяють користувачам уважніше вивчити кожну точку: подивитися точне числове значення при наведенні миші, приховати нецікаві ряди в візуалізації, наблизити певну ділянку графіка і т. д.

Стовпчикова діаграма з рисунку 1.1, що будувалася в Matplotlib з Plotly виглядатиме наступним чином:

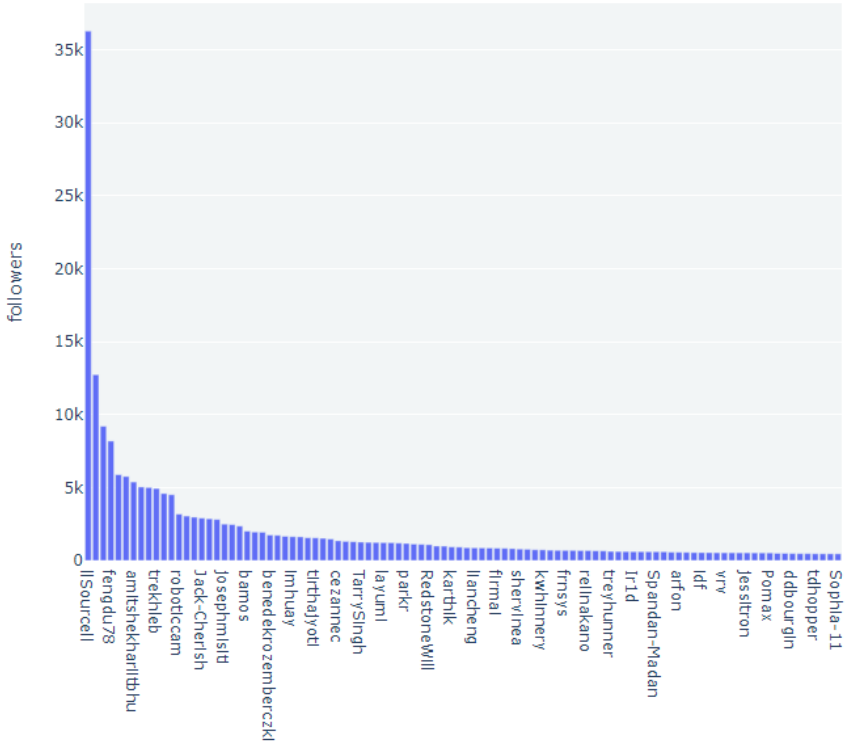

Рисунок 1.5 – Стовпчикова діаграма з Plotly

3. Легко робити складні графіки

За допомогою Plotly досить легко створювати складні графіки. Наприклад, якщо потрібно створити карту для візуалізації розташування користувачів GitHub, то можна знайти широту і довготу їх розташування, а потім використовувати ці дані, щоб відзначити місцезнаходження користувачів вже на карті:

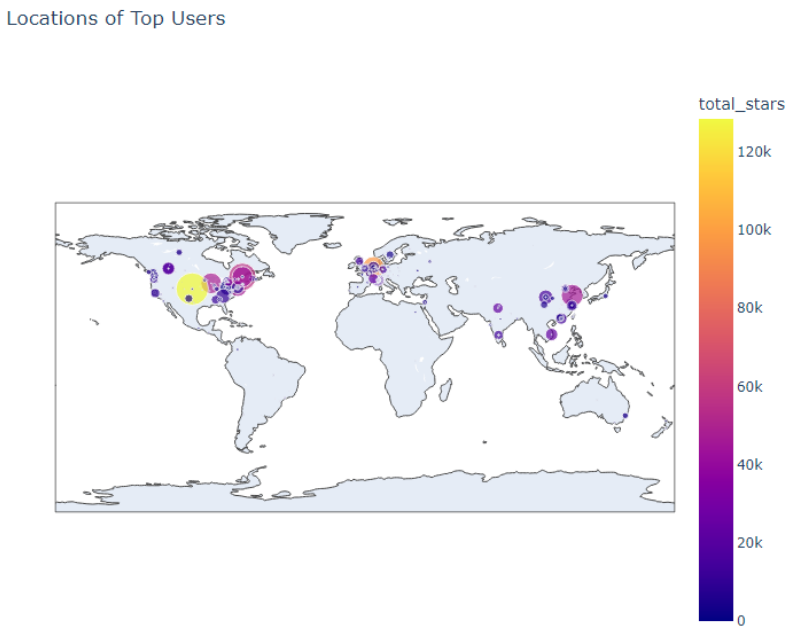

Рисунок 1.6 – Інтерактивна карта з Plotly

Altair – бібліотека візуалізації для Python, заснована на Vega, що ідеально підходить для графіків, що потребують великої кількості статистичних перетворень.

Плюси:

1. Проста граматика візуалізації

Граматика, яка використовується для візуалізації, неймовірно проста для розуміння. Необхідно лише позначити зв'язок між стовпцями даних і каналами їх перетворення, а все інше обробляється автоматично. Це звучить досить абстрактно, але має вирішальне значення, коли користувач працює з даними, та робить візуалізацію інформації дуже швидкою та інтуїтивно зрозумілою.

2. Простота перетворення даних

3. Зв'язування кількох графіків

Altair також дозволяє створювати вражаючі зв'язки між графіками, наприклад з можливістю використовувати вибір інтервалу для фільтрації вмісту прикріпленої гістограми.

Мінуси:

Якщо використовувати стилі за замовчуванням, то прості діаграми не будуть оформлені стилістично так само добре, як у Seaborn або Plotly. Altair також не рекомендує використовувати набори даних з більш ніж 5000 екземплярами і рекомендує замість цього агрегувати дані перед візуалізацією. Немає можливості використовувати зовнішні стилізовані частини картки, такі як OSM, Mapbox тощо. API погано документовано.

Folium [1] поєднує простоту використання екосистеми Python і сильні сторони картографування бібліотеки leaflet.js. Це дозволяє візуалізувати налаштовувані, інтерактивні карти, а також передавати багаті векторні, растрові, HTML-візуалізації у вигляді маркерів на карті.

Плюси:

1. Дуже легко створювати карти з маркерами

Незважаючи на те, що Plotly також дозволяє створювати карти, Folium має ряд вбудованих наборів частин карт з OpenStreetMap, Mapbox і Stamen, а також підтримує

набори користувачів через API Mapbox або Cloudmade. Підтримуються зображення, відео, GeoJSON та TopoJSON.

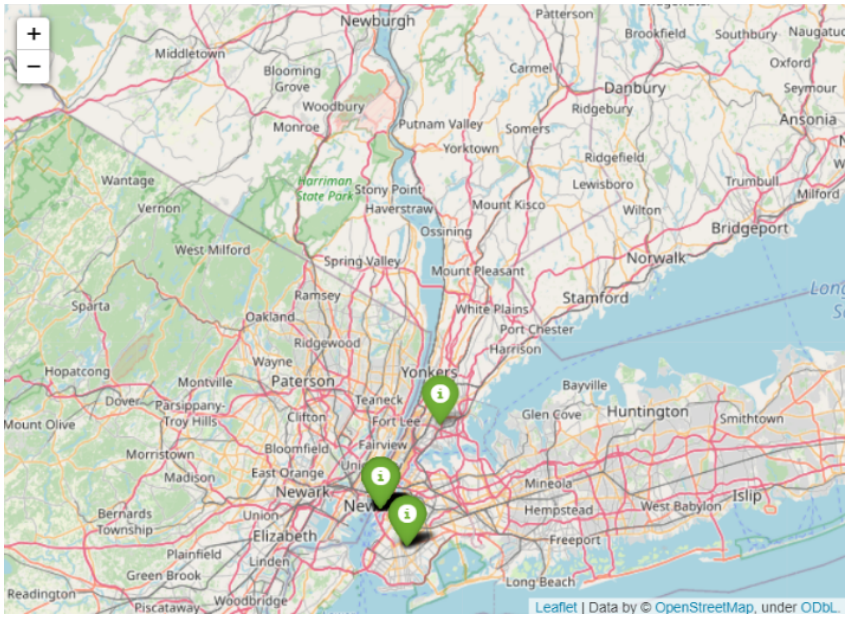

Рисунок 1.7 – Карта з маркерами

2. Опція відображення карти у вигляді HTML та інші опції вбудовування фрагмента у зовнішні вебпрограми
3. Плагіни

Folium має ряд плагінів, які можна використовувати зі своєю картою.

Висновки до розділу

У цьому розділі здійснено огляд можливостей сучасних СППР, а також проаналізовано ряд бібліотек для побудови візуалізацій на Python, а саме: Matplotlib, Seaborn, Plotly Altair та Folium. Вибір зроблено на користь Plotly та Folium. Перший чудово підходить для створення інтерактивних та якісних графіків і не вимагає написання великої кількості коду. Folium, подібно до Google Map, використовує відкриту вуличну карту, що дозволить візуалізувати наявну статистичну інформацію у розрізі районів міста Окленда.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Формат представлення географічних даних GeoJSON

GeoJSON – це формат представлення різних структур географічних даних [2]. Об’єкт GeoJSON може бути представлений геометрією (geometry), об’єктом (feature) або колекцією об’єктів (feature collection). GeoJSON підтримує такі геометричні типи: Point (точка), LineString (ламана), Polygon (полігон), MultiPoint (мультиточка), MultiLineString (мультиламана), MultiPolygon (мультиполігон) та GeometryCollection (колекція геометрій). Об’єкт (feature) складається з геометрії та додаткових властивостей, колекція об’єктів (feature collection) – з набору об’єктів (feature).

У GeoJSON існує ієрархія об’єктів виду GeoJSON Object => feature collection => feature => geometry. Щоб уникнути плутанини в термінах, називатимемо GeoJSON Object – об’єкт GeoJSON, feature collection – колекція елементарних об’єктів, feature – елементарний об’єкт, geometry – геометрія.

Завершена структура даних GeoJSON завжди об’єкт (в термінах JSON). GeoJSON об’єкт складається з набору пар ключ/значення, що також називаються властивостями. Ім’я кожної властивості – рядок. Значення властивості може бути рядком, числом, об’єктом, масивом або одним з літералів: “true”, “false” та “null”. Масив складається з елементів, де кожен елемент може набувати одне з значень, описаних вище.

2.1.1 Об’єкти GeoJSON

GeoJSON завжди є однією сутністю: геометрією, елементарним об’єктом або колекцією елементарних об’єктів.

- Об’єкт GeoJSON може мати довільну кількість властивостей (пар ключ/значення).
- Об’єкт GeoJSON повинен мати властивість “type”. Значення цієї властивості – рядок, який містить тип об’єкта GeoJSON.

- Значення властивості “type” має приймати одне з наступних значень: “Point”, “MultiPoint”, “LineString”, “MultiLineString”, “Polygon”, “MultiPolygon”, “GeometryCollection”, “Feature” або “FeatureCollection”. Регістр символів значення поля “type” має значення.
- Об’єкт GeoJSON може мати необов’язкову властивість “crs”, значення якої має містити об’єкт системи координат.
- Об’єкт GeoJSON може мати властивість “bbox”, значення якооїї є масивом координат вершин обмежуючого прямокутника.

Геометрія – це об’єкт GeoJSON, для якого як значення властивості “type” використовується один із рядків: “Point”, “MultiPoint”, “LineString”, “MultiLineString”, “Polygon”, “MultiPolygon”, “GeometryCollection”.

Усі геометрії, тип яких відрізняється від “Geometry Collection”, повинні мати властивість “coordinates”. Значення цієї властивості завжди є масивом. Структура елементів масиву визначається типом геометрії.

Координати – фундаментальна геометрична концепція. Властивість “coordinates” об’єкта геометрія складається з пари/триплету координат (у разі геометрії типу “Point”), масиву координат (об’єкти типу “LineString” або “MultiPoint”), масиву масивів координат (об’єкти типу “Polygons”, “MultiLineStrings”) або багатовимірного масиву координат (об’єкти типу “MultiPolygon”).

Координати визначаються масивом чисел. Цей масив повинен містити щонайменше два елементи, але їх може бути більше. Порядок елементів може бути таким: x, y, z (для даних у прямокутній системі координат – зміщення на схід, зміщення на північ, висота, для даних у географічній системі координат – довгота, широта, висота). Допускається введення додаткових елементів, проте їх інтерпретація виходить за межі цієї специфікації.

Для об’єктів типу “Point” властивість “coordinates” повинна містити одну пару/триплет координат, а для об’єктів типу “MultiPoint” – масив.

Для об’єктів типу “LineString” властивість “coordinates” має містити масив із двох і більше пар/триплетів. Тип “LinearRing” – це замкнутий “LineString”, що містить 4 і більше пар/триплетів координат. Перша та остання пара/триплет еквівалентні

(представлені однаковими точками). Хоча тип “LinearRing” явно не входить до списку типів геометрій, він використовується при описі типу “Polygon”.

Для об’єктів типу “MultiLineString” властивість “coordinates” повинна містити масив масивів пар/триплетів координат “LineString”.

Для об’єктів типу “Polygon” властивість “coordinates” повинна містити масив масивів пар/триплетів координат “LinearRing”. Для полігонів з кількома кільцями першим має йти опис зовнішнього кільця і лише потім внутрішніх чи отворів.

Для об’єктів типу “MultiPolygon” властивість “coordinates” повинна містити масив масивів пар/триплетів координат “Polygon”.

Об’єкт типу “GeometryCollection” – це геометрія, що представляє колекцію інших геометрій. Така колекція повинна мати властивість з ім’ям “geometries”. Значення цієї властивості – масив. Кожен елемент цього масиву є геометрією.

Об’єкт GeoJSON типу Feature – **елементарний об’єкт**.

- Елементарний об’єкт повинен мати властивість “geometry”. Значення цього поля – геометрія або JSON null.
- Елементарний об’єкт повинен мати властивість “properties”. Значення цієї властивості – об’єкт (будь-який об’єкт JSON або JSON null).
- Якщо елементарний об’єкт має певний ідентифікатор, його слід включити окремою властивістю з ім’ям “id”.

Об’єкт GeoJSON типу FeatureCollection – **колекція елементарних об’єктів**. Об’єкт типу FeatureCollection повинен містити властивість features. Значення цієї якості – масив. Кожен елемент цього масиву є елементарним об'єктом.

2.1.2 Об’єкти системи координат

Система координат (CRS) об’єкта GeoJSON визначається значенням властивості “crs” (далі – об’єкт CRS). Якщо об’єкт GeoJSON не має властивості “crs”, вона може бути успадкована від батьківського або прабатьківського об’єкта. Якщо значення “crs” не може бути визначено, то до GeoJSON об’єкту застосовується значення CRS за замовчуванням.

- За замовчуванням використовується географічна система координат WGS84 у десяткових градусах довготи та широти.
- Значення “crs” має представляти об’єкт JSON або значення JSON null. Значення властивості “crs” – null означає відсутність інформації про систему координат.
- Властивість “crs” слід розміщувати на верхньому рівні ієрархії об’єкта GeoJSON (порядок ієрархії: колекція елементарних об’єктів => елементарний об’єкт => геометрія) і не треба повторювати або перевизначати в дочірніх об’єктах.
- Не порожній об’єкт (не null) CRS має дві обов’язкові властивості: “type” і “properties”.
- Значенням властивості “type” має бути рядок, що описує тип об’єкта CRS.
- Значенням властивості “properties” має бути об’єкт.
- Система координат не повинна змінювати порядок проходження координат.

Система координат може бути описана шляхом вказування імені. У цьому випадку значення властивості “type” має містити рядок “name”. Значення властивості “properties” має представляти об’єкт, який містить властивість “name”. Значенням властивості “name” має бути рядок, що визначає систему координат. Краще використовувати ідентифікатори проекцій OGC CRS URN замість таких ідентифікаторів, як “EPSG:4326”.

Параметри об’єкта CRS можна отримати через мережу. У цьому випадку значення властивості “type” має бути “link”, а властивість “properties” є об’єктом Link.

Об’єкт Link має одну обов’язкову властивість: “href” та одну опціональну: “type”. Значення властивості “href” має бути URI. Значення опціональної властивості “type” має бути рядком, який описує формат представлення системи координат, що надається за URI. Пропоновані значення: “proj4”, “ogcwkt”, “esriwkt”, але можуть використовуватися й інші значення.

2.1.3 Обмежуючі прямокутники

Щоб включити інформацію про діапазон координат для геометрій, елементарних об’єктів та колекцій елементарних об’єктів, використовується

властивість “bbox” (обмежуючий прямокутник) об’єкта GeoJSON. Значення даної властивості має бути масивом розмірності $2n$, де n – розмірність геометрій, що входять в об’єкт, і містити мінімальні та максимальні значення координат усіх координатних осей. Порядок осей в описі обмежуючого прямокутника відповідає порядку осей, що використовуються в описах геометрій. Крім того, передбачається, що система координат обмежуючого прямокутника відповідає системі координат об’єкта GeoJSON, властивістю якого вона є.

2.2 Попередня обробка статистичних даних

Інтелектуальна система підтримки прийняття рішень вибору району проживання реалізується на прикладі міста Окленд в Новій Зеландії, що зумовлено наявністю відповідних статистичних даних. Усі подальші міркування можуть бути легко перенесені на будь-який інших регіон проживання.

Обробка та аналіз даних здійснюються в Jupyter Notebook, що є потужним інструментом для розробки та представлення проектів Data Science у інтерактивному вигляді.

Зчитування, маніпулювання та аналіз географічних даних здійснюється з використанням Geopandas [3]. Geopandas – бібліотека для роботи з географічними даними в Python, побудована на основі бібліотеки Pandas та Numpy[4,5]. Як і Pandas Dataframe, структура даних Geopandas містить GeodataFrame та GeoSeries. Geopandas пропонує не лише можливість легко читати та маніпулювати географічними даними, а також може виконувати безліч важливих геопросторових операцій. Також Geopandas забезпечує високоякісний інтерфейс для бібліотеки Matplotlib, використовуючи метод plot() на GeodataFrame / GeoSeries.

Статистичні дані щодо злочинів на території Нової Зеландії, а саме: категорія злочину, дата, час та місце, кількість постраждалих і т. д., взято з сайту поліції країни [6]. Тут місце злочину визначається меш-блоком (див. рис. 2.1), що є мінімальною географічною одиницею в Новій Зеландії, за якою збираються і

опрацьовуються статистичні дані. Надалі під районом міста будемо розуміти саме меш-блок.

Year Month	ANZSOC Division	ANZSOC Group	ANZSOC Subdivision	Area Unit	Location Type	Locn Type Division	Meshblock	Occurrence Day Of Week	Occurrence Hour Of Day	Territorial Authority	Weapon	Victimisations
2014-07	Abduction, Harassment and Other Related Offenc...	Abduction and Kidnapping	Abduction and Kidnapping	Mt Albert Central	.	Community Location	526700	Monday	12.0	Auckland	Not Applicable	1
2014-07	Abduction, Harassment and Other Related Offenc...	Abduction and Kidnapping	Abduction and Kidnapping	Mangere South	.	Community Location	727200	Thursday	6.0	Auckland	Not Applicable	1
2014-07	Abduction, Harassment and Other Related Offenc...	Abduction and Kidnapping	Abduction and Kidnapping	Ormiston	.	Community Location	712401	Tuesday	8.0	Auckland	Not Stated	1

Рисунок 2.1 – Статистичні дані про злочини на території міста (1294334 записів)

З метою подальшої кластеризації районів міста, необхідно належним чином представити інформацію, тобто сформувати матрицю об’єктів-ознак. Кожен район описується 42 ознаками, що є кількостями постраждалих у розрізі 6 категорій злочинів за 7 попередніх років (див. рис. 2.2).

Year	2014						2015						20
ANZSOC Division	Abduction, Harassment and Other Related Offences Against a Person	Acts Intended to Cause Injury	Robbery, Extortion and Related Offences	Sexual Assault and Related Offences	Theft and Related Offences	Unlawful Entry With Intent/Burglary, Break and Enter	Abduction, Harassment and Other Related Offences Against a Person	Acts Intended to Cause Injury	Robbery, Extortion and Related Offences	Sexual Assault and Related Offences	Theft and Related Offences	Unlawful Entry With Intent/Burglary, Break and Enter	Ab Ha an Re Of Ag Pe
Meshblock													
160900	0	0	0	0	0	0	0	2	0	1	9	1	
759514	0	0	0	0	0	0	0	0	0	0	6	1	
712219	0	0	0	0	0	0	0	0	0	0	1	0	
683300	0	0	2	1	4	7	0	0	2	0	6	11	
400000	0	0	0	0	2	2	0	0	0	0	2	2	

Рисунок 2.2 – Сумарна інформація за районами міста

Інформацію про географічні координати районів (їх межі) було взято з сайту Data.govt.nz:

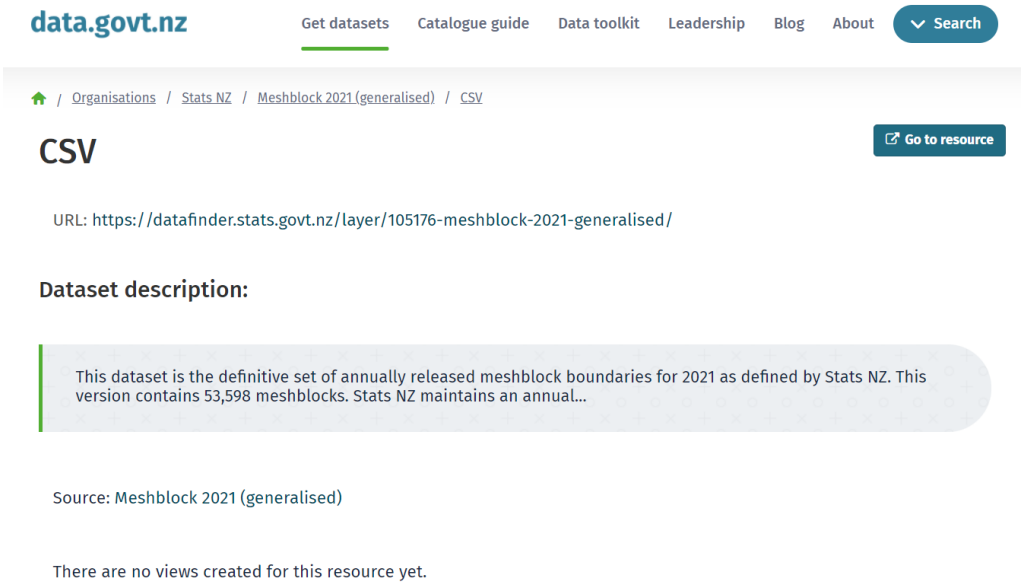

Рисунок 2.3 – Датасет з межами меш-блоків в Новій Зеландії

Для подальшого використання статистичної інформації в системі підтримки прийняття рішень дані було збережено в csv-файл.

Окрім того, опрацьовано наявну інформацію про шкільні заклади, завантажену з csv-файлу:

URL	Add1_Line1	Add1_Suburb	Add1_City	...	Decile	Roll_Date	Total	European	Māori	Pacific	Asian	MELAA
http://www.snellsbeach.school.nz	62 Dawson Road	NaN	Snells Beach	...	6.0	2020-11-01 00:00:00.0000000	332	221	53	29	9	19
http://www.somervilleintermediate.school.nz	39 Somerville Road	Howick South	Auckland	...	10.0	2020-11-01 00:00:00.0000000	958	357	49	42	377	111
http://www.tematauranga.school.nz	206 Finlayson Avenue	Clendon Park	Auckland	...	1.0	2020-11-01 00:00:00.0000000	398	3	115	248	29	2
http://www.wgpcollege.school.nz/	8 Stanmore Bay Road	NaN	Stanmore Bay	...	9.0	2020-11-01 00:00:00.0000000	1578	1047	228	45	107	108
http://www.horizon.school.nz/	410 Mahurangi East Road	NaN	Snells Beach	...	6.0	2020-11-01 00:00:00.0000000	213	149	23	10	7	18

Рисунок 2.4 – Статистичні дані по шкільних закладах

Зокрема, інтерес представляють статистичні дані по кількостях дітей європейців, азійців, корінного населення і т. д. На рисунку 2.5 відповідна інформація подана у відсотковому співвідношенні.

	Org_Name	Latitude	Longitude	European	Māori	Pacific	Asian	MELAA	Other	International
2	Snells Beach Primary School	-36.428453	174.723704	66.566265	15.963855	8.734940	2.710843	5.722892	0.301205	0.000000
6	Somerville Intermediate School	-36.912840	174.939070	37.265136	5.114823	4.384134	39.352818	11.586639	0.104384	2.192067
7	Te Matauranga	-37.035348	174.860313	0.753769	28.894472	62.311558	7.286432	0.502513	0.251256	0.000000

Рисунок 2.5 – Контингент школярів у відсотках

Наступний код реалізує побудову відображення всіх шкіл, а також для вибраного контингенту на карті міста:

```
y_map, x_map = data[['Latitude', 'Longitude']].mean().values
mymap = folium.Map(location=[y_map, x_map], zoom_start=12,
tiles='cartodbpositron')
color_list =
['#003f5c','#374c80','#7a5195','#bc5090','#ef5675','#ff764a','#ffa600']
percent_columns = data.columns[3:].tolist()
for column, color in zip(percent_columns, color_list):
    lgd_text = '<span style="color: transparent;  text-shadow: 0 0 0 {};
">&#9899</span> {}'.format(color, column)
    fg = folium.FeatureGroup(lgd_text)
    mymap.add_child(fg)
    df = data[data[column] > 0]
    for lat, lon, name, percent in zip(df['Latitude'], df['Longitude'],
df['Org_Name'], df[column]):
        popup_text = "<b>{}</b><br>{}: {}%".format(name, column,
round(percent,2))
        folium.CircleMarker(
            [lat, lon],
            radius=0.2*percent,
            color=color,
            fill_color=color,
            popup=popup_text,
            fill=True,
            fill_opacity=0.3
            ).add_to(fg)#.add_to(marker_cluster)
folium.LayerControl(collapsed=False).add_to(mymap)
mymap.save('Auckland_schools.html')
```

Карту створено з використанням бібліотеки Folium та збережено в html-файл для подальшого використання в системі.

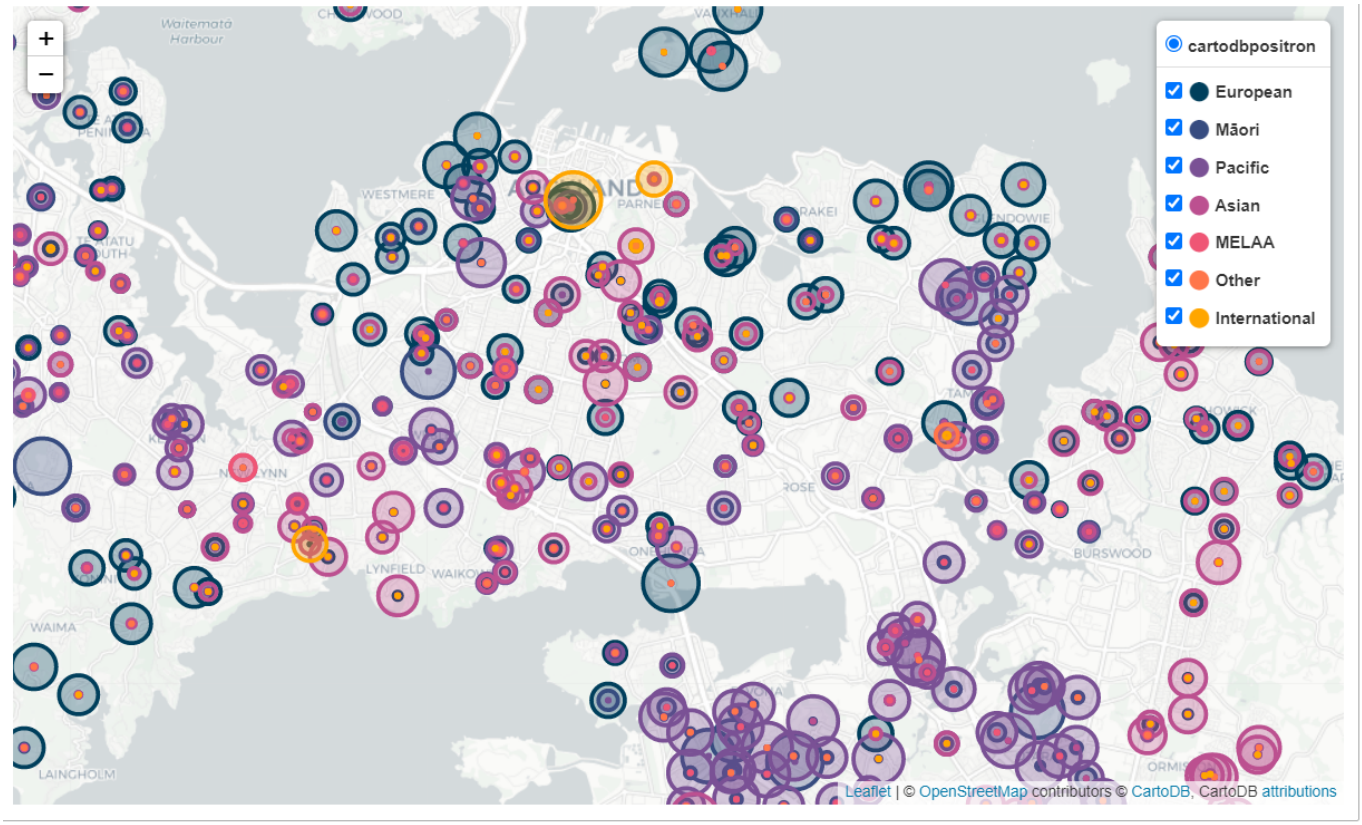

Рисунок 2.6 – Карта шкіл міста з урахуванням контингенту

Висновки до розділу

У розділі описано особливості роботи з форматом представлення географічних даних GeoJSON.

Проведено попередню обробку наявної статистичної інформації по шкільних закладах та реалізовано її візуалізацію на карті міста з використанням бібліотеки Folium. Підготовлено дані щодо злочинів на території Окленда для подальшої кластеризації його районів.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Кластеризація

Інтуїтивна постановка задачі кластеризації досить проста і представляє собою бажання сказати: “Ось тут у мене насипані точки. Я бачу, що вони об’єднуються в якісь групи. Було б непогано описати ці речі більш конкретно, щоб в разі появи нової точки на площині віднести її до правильної групи.” З такої постановки видно, що простору для фантазії виходить багато, і виникає відповідна множина алгоритмів вирішення цієї задачі. Перераховані алгоритми ні в якому разі не описують дану множину повністю, але є прикладами найпопулярніших методів кластеризації.

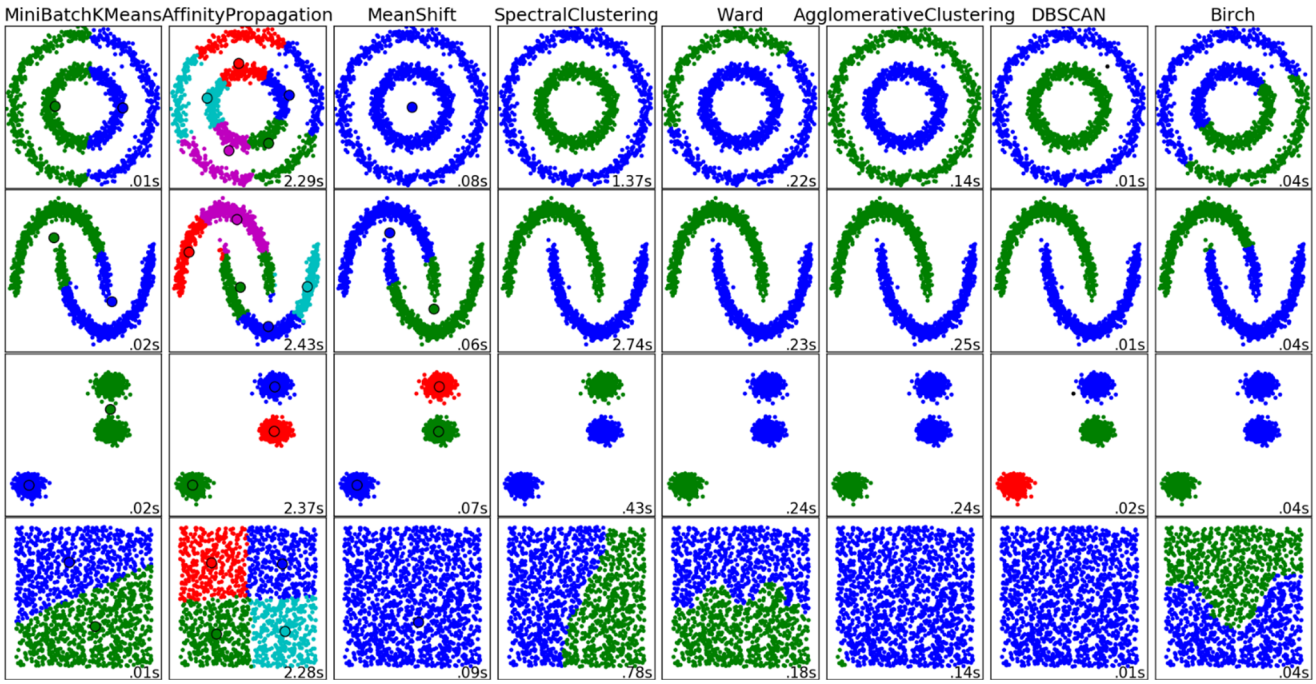

Рисунок 3.1– Приклади роботи алгоритмів кластеризації
з документації пакета scikit-learn

Більш формально, задача кластеризація – це задача навчання без учителя, що полягає в наступному: Нехай є навчальна вибірка $X^\ell = \{x_1, \dots, x_\ell\} \subset X$ і деяка функція відстані між об’єктами $\rho(x, x')$. Необхідно розбити вибірку на підмножини, що не перетинаються (кластери), так, щоб кожен кластер складався з об’єктів, близьких за метрикою ρ, а об’єкти різних кластерів суттєво відрізнялися. При цьому кожному об’єкту $x_i \in X^\ell$ приписується мітка (номер) кластера y_i.

3.1.1 K-means

Алгоритм К-середніх, напевно, найбільш популярний та простий алгоритм кластеризації і дуже легко представляється у вигляді простого псевдокоду:

1. Вибрати кількість кластерів k , яке здається оптимальною для вказаних даних.
2. Висипати випадковим чином в простір даних k точок (центроїдів).
3. Для кожної точки заданого набору даних порахувати, до якого центроїду вона ближче.
4. Перенести кожен центроїд в центр вибірки, яку було віднесено до цього центроїду.
5. Повторювати останні два кроки фіксоване число разів, або до тих пір поки центроїди не «зійдуться» (зазвичай це означає, що їх зміщення відносно попереднього положення не перевищує якогось наперед заданого невеликого значення).

Також варто зауважити, що алгоритм буде збігатися у випадку будь-якої метрики, тому для різних задач кластеризації в залежності від даних можна експериментувати не тільки з кількістю кроків або критерієм збіжності, але і з метрикою, за якою обчислюються відстані між точками і центроїдами кластерів.

Іншою особливістю цього алгоритму є те, що він чутливий до вихідного положення центроїдів кластерів в просторі. У такій ситуації рятує декілька послідовних запусків алгоритму з подальшим усередненням отриманих кластерів.

Ініціалізація центроїдів. Реалізація алгоритму в scikit-learn (`sklearn.cluster.KMeans`) володіє масою зручних налаштувань. Параметр `n_init` дозволяє задати кількість запусків з різних початкових наближень, що дасть більш стійкі центроїди для кластерів в разі скошених даних. До того ж ці запуски можна робити паралельно, не жертвуючи часом обчислення.

Для задання способу ініціалізації центроїдів використовується параметр `init`. Можливі варіанти:

- `k-means++` – «розумна» ініціалізація центроїдів для прискорення збіжності.
- `random` – випадкова ініціалізація центроїдів.
- `ndarray` – задана ініціалізація центроїдів.

Вибір кількості кластерів для K-means. На відміну від задачі класифікації або регресії, у випадку кластеризації складніше вибрати критерій, за допомогою якого було б просто подати задачу кластеризації як задачу оптимізації. У разі K-Means поширений ось такий критерій – сума квадратів відстаней від точок до центроїдів кластерів, до яких вони належать.

$$J(C) = \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - \mu_k\|^2 \to \min_C$$

Тут C – множина кластерів потужності K, μ_k – центр ваги кластера C_k.

Зрозуміло, що здоровий глузд у цьому є: ми хочемо, щоб точки розміщувалися купчасто біля центрів своїх кластерів. Але от невдача: мінімум такого функціонала буде досягатися тоді, коли кластерів стільки ж, скільки і точок (тобто кожна точка – це кластер з одного елемента). Для вирішення цього питання (вибору кількості кластерів) часто користуються такою евристикою: вибирають ту кількість кластерів, починаючи з якої описаний функціонал $J(C)$ спадає «вже не так швидко». Або більш формально:

$$D(k) = \frac{|J(C_k) - J(C_{k+1})|}{|J(C_{k-1}) - J(C_k)|} \to \min_k$$

Складність. Саме по собі розв'язання задачі K-Means NP-складе і для розмірності d, кількості кластерів k і кількості точок n розв'язується за $O(n^{dk+1})$. Для вирішення цієї проблеми часто використовуються евристики, наприклад MiniBatch K-Means, який для навчання використовує не повністю весь датасет, а лише маленькі його порції (batch) і оновлює центроїди використовуючи середнє за всю історію оновлень центроїда від всіх точок, що відносяться до нього.

3.1.2 Агломеративна кластеризація

Напевно найбільш простий та зрозумілий алгоритм кластеризації без фіксованої кількості кластерів — агломеративна кластеризація. Логіка алгоритму дуже проста:

1. Починається з того, що висипається на кожну точку свій кластер.
2. Сортуються попарні відстані між центрами кластерів за зростанням.
3. Береться пара найближчих кластерів, склеюється в один і перераховується центр кластера.
4. Повторюються п. 2 і 3 до тих пір, поки всі дані не будуть склеєні в один кластер.

Сам процес пошуку найближчих кластерів може відбуватися з використанням різних методів об’єднання точок:

1. Single linkage — мінімум попарних відстаней між точками з двох кластерів $d(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} \|x_i - x_j\|$
2. Complete linkage — максимум попарних відстаней між точками з двох кластерів $d(C_i, C_j) = \max_{x_i \in C_i, x_j \in C_j} \|x_i - x_j\|$
3. Average linkage — середнє попарних відстаней між точками з двох кластерів $d(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x_i \in C_i} \sum_{x_j \in C_j} \|x_i - x_j\|$
4. Centroid linkage — відстань між центроїдами двох кластерів $d(C_i, C_j) = \|\mu_i - \mu_j\|$

Перевага перших трьох підходів у порівнянні з четвертим в тому, що для них не потрібно буде перераховувати відстані кожен раз після склеювання, що сильно знижує обчислювальну складність алгоритму.

За підсумками виконання такого алгоритму можна також побудувати чудове дерево склеювання кластерів і дивлячись на нього визначити, на якому етапі було б

найбільш оптимально зупинити алгоритм. Або скористатися тим самим правилом, що і в K-means.

3.1.3 Метрики якості кластеризації

Задача оцінки якості кластеризації є більш складною у порівнянні з оцінкою якості класифікації. По-перше, такі оцінки не повинні залежати від самих значень міток, а тільки від розбиття вибірки. По-друге, не завжди відомі істинні мітки об'єктів, тому також потрібні оцінки, що дозволяють оцінити якість кластеризації, використовуючи тільки нерозмічену вибірку.

Виділяють *зовнішні* і *внутрішні* метрики якості. Зовнішні використовують інформацію про істинне розбиття на кластери, в той час як внутрішні метрики не використовують ніякої зовнішньої інформації і оцінюють якість кластеризації, грунтуючись тільки на наборі даних. Оптимальну кількість кластерів зазвичай визначають з використанням внутрішніх метрик.

Силует – це внутрішня метрика оцінки якості кластеризації. Даний коефіцієнт не передбачає знання істинних міток об'єктів і дозволяє оцінити якість кластеризації, використовуючи тільки саму (нерозмічену) вибірку і результат кластеризації. Спочатку силует визначається окремо для кожного об'єкта. Позначимо через a – середню відстань від даного об'єкта до об'єктів з того ж кластера, через b – середню відстань від даного об'єкта до об'єктів з найближчого кластера (відмінного від того, в якому лежить сам об'єкт). Тоді силуетом даного об'єкта називається величина:

$$\mathrm{s} = \frac{\mathrm{b} - \mathrm{a}}{\max(\mathrm{a}, \mathrm{b})}$$

Силуетом вибірки називається середня величина силуету об'єктів даної вибірки. Таким чином, силует показує, на скільки середня відстань до об'єктів свого кластера відрізняється від середньої відстані до об'єктів інших кластерів. Дана величина лежить в діапазоні $[-1,1]$. Значення, близькі до -1, відповідають поганим (розрізненим) кластеризаціям, значення, близькі до нуля, говорять про те, що кластери перетинаються і накладаються один на одний, значення, близькі до 1, відповідають «щільним», чітко виділеним кластерам. Таким чином, чим більший

силует, тим чіткіше виділені кластери, і вони є компактними, щільно згрупованими хмарами точок.

За допомогою силуету можна вибирати оптимальну кількість кластерів k (якщо вона заздалегідь не є відома) – вибирається кількість кластерів, що максимізує значення силуету. На відміну від попередніх метрик, силует залежить від форми кластерів і досягає великих значень на більш опуклих кластерах, одержуваних за допомогою алгоритмів, що базуються на відновленні щільності розподілу.

3.2 Метод головних компонент (PCA)

Метод головних компонент (Principal Component Analysis) – один з найбільш інтуїтивно простих і часто використовуваних методів для зниження розмірності даних і проекції їх на ортогональний підпростір ознак.

У зовсім загальному вигляді це можна представити як припущення про те, що всі спостереження швидше за все виглядають як деякий еліпсоїд в підпросторі вихідного простору і новий базис в цьому просторі співпадає з осями цього еліпсоїда. Це припущення дозволяє одночасно позбутися від сильно скорельованих ознак, так як вектори базису простору, на який проектуються дані, будуть ортогональними.

У загальному випадку розмірність цього еліпсоїда буде рівна розмірності вихідного простору, але припущення про те, що дані лежать в підпросторі меншої розмірності, дозволяє відкинути «зайвий» підпростір в новій проекції, а саме той підпростір, вздовж осей якого еліпсоїд буде найменш розтягнутий. Це робиться «жадібно», вибираючи по-черзі в якості нового елемента базису нового підпростору послідовно вісь еліпсоїда з решти, вздовж якої дисперсія буде максимальною.

3.3 Кластеризація районів міста

Розглянемо детально процес кластеризації районів міста. Кожен район характеризується 42 ознаками, що є кількостями постраждалих у розрізі 6 категорій злочинів за 7 попередніх років (див. рис. 2.2).

Після масштабування даних та застосування методу головних компонент [7, 8] кількість ознак було зменшено до 24 із збереженням 90% дисперсії вихідних ознак.

Для кластеризації даних обрано одні з найпростіших та інтуїтивно зрозумілих алгоритмів: K-Means та агломеративну кластеризацію [9], що реалізовані в модулі `sklearn.cluster` Python-бібліотеки scikit-learn (класи `KMeans` та `AgglomerativeClustering` відповідно).

Метрика Силует (Silhouette) [10] дозволяє оцінити якість кластеризації, грунтуючись тільки на нерозмічених даних і результаті кластеризації. Змінюючи кількість кластерів від 2 до 10 було отримано наступні оцінки.

Silhouette	2	3	4	5	6	7	8	9	10
K-means	0.939871	0.788525	0.781413	0.533748	0.535312	0.455961	0.437901	0.347377	0.331858
Agglomerative	0.938343	0.936359	0.668727	0.668768	0.667494	0.613286	0.613332	0.050812	0.051037

Рисунок 3.2 – Оцінки якості кластеризацій у розрізі кількості кластерів

Найкраща оцінка, яка відповідає розбиттю на 2 кластери методом K-Means є близькою до одиниці і говорить про отримання «щільних», чітко виділених кластерів, а кількості елементів в них (11382 в одному кластері і 81 в іншому) про виокремлення районів, що суттєво відрізняються від загальної маси. На рисунку 3.3 представлена карта із зазначенням таких районів.

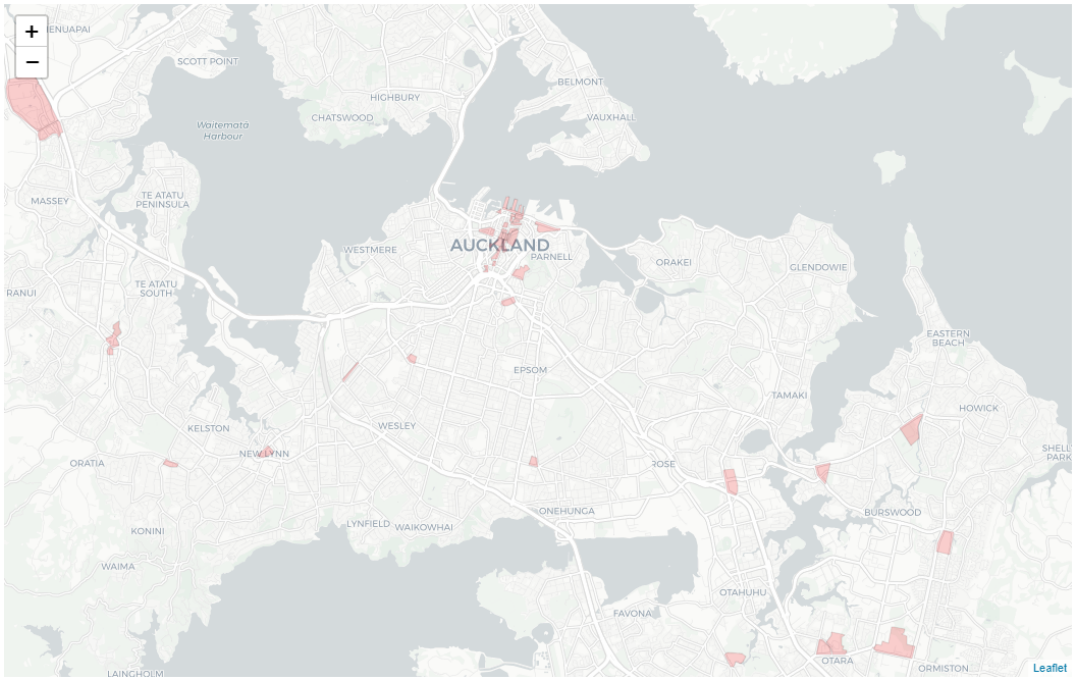

Рисунок 3.3 – Райони з підвищеним рівнем злочинності

Висновки до розділу

Сформульоване в дипломній роботі завдання передбачає розв’язання задачі кластеризації. У розділі розглядаються алгоритми K-Means та агломеративна кластеризація, а також метрика Силует, що використовується для вибору моделі та оптимальної кількості кластерів.

Найкраща оцінка (0,94) відповідає розбиттю районів міста Окленда на 2 кластери методом K-Means. Незбалансованість отриманих кластерів (11382 і 81 елементів) говорить про виокремлення елементів, що сильно відрізняються від інших за ознаками розбиття, тобто районів з підвищеним рівнем злочинності.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Dash – фреймворк для створення вебдодатків

Візуалізація даних є невід'ємною частиною аналітичних додатків. Dash – це фреймворк для візуалізації даних на Python [13], який дозволяє працювати не тільки з інтерактивними графіками, але й виводити їх на вебсайт.

Dash написаний за допомогою Flask, React.js, Plotly.js і є інструментом для інтерактивної візуалізації, яку можна інтегрувати у вебсайт (див. рис. 4.1). При цьому для роботи з ним не потрібні знання ні Flask, ні React, ні JavaScript.

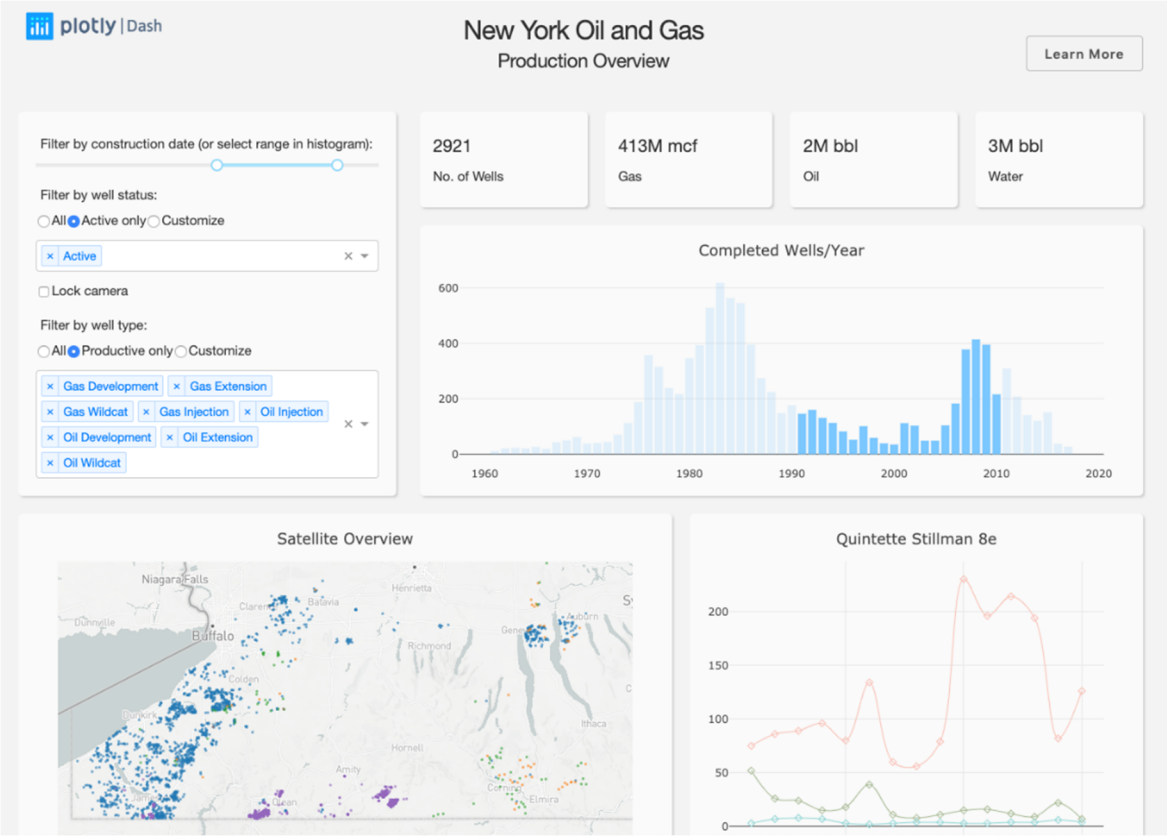

Рисунок 4.1 – Візуалізація даних з Dash

Знаючи лише Python, деякі теги HTML та базові графіки Plotly, можна реалізувати різні проекти з візуалізації даних, наприклад: 3D-зображення мозку людини, детектування об'єктів на зображенні на основі методів глибокого навчання (Deep Learning) та багато іншого.

Встановити Dash можна за допомогою пакетного менеджера Pip, який додатково встановить такі залежності як Plotly та Flask:

```
pip install dash
```

Створимо програму з Dash. Наведено код на Python, який потрібно зберегти в окремому файлі та запустити його через термінал. Запускається програма як звичайний Python-файл: `python my_app.py`. Після запуску буде виведена адреса, за якою потрібно перейти через браузер.

```
import dash
import dash_html_components as html
app = dash.Dash('my_app')
app.layout = html.H1('Hi!')
app.run_server(debug = True)
```

Ця програма лише виводить на вебсторінці слово ‘Hi’.

Ініціалізація програми здійснюється через `dash.Dash`, де вказується назва програми; Атрибут `layout` визначає, що буде на вебсторінці. Тут було вказано, що на сторінці відображатиметься лише заголовок `H1` (тег). Усі компоненти HTML знаходяться у модулі `dash_html_components`. Запускається програма через `run_server`. Для налагодження коду встановлюється `debug = True`. Якщо програма готова для роботи на сервері, то з метою безпеки потрібно змінити значення на `False`.

4.2 Особливості роботи системи

Інтелектуальну систему підтримки прийняття рішень вибору району проживання реалізовано на мові програмування Python з використанням бібліотеки для створення аналітичних вебдодатків Dash. Вона має наступний вигляд:

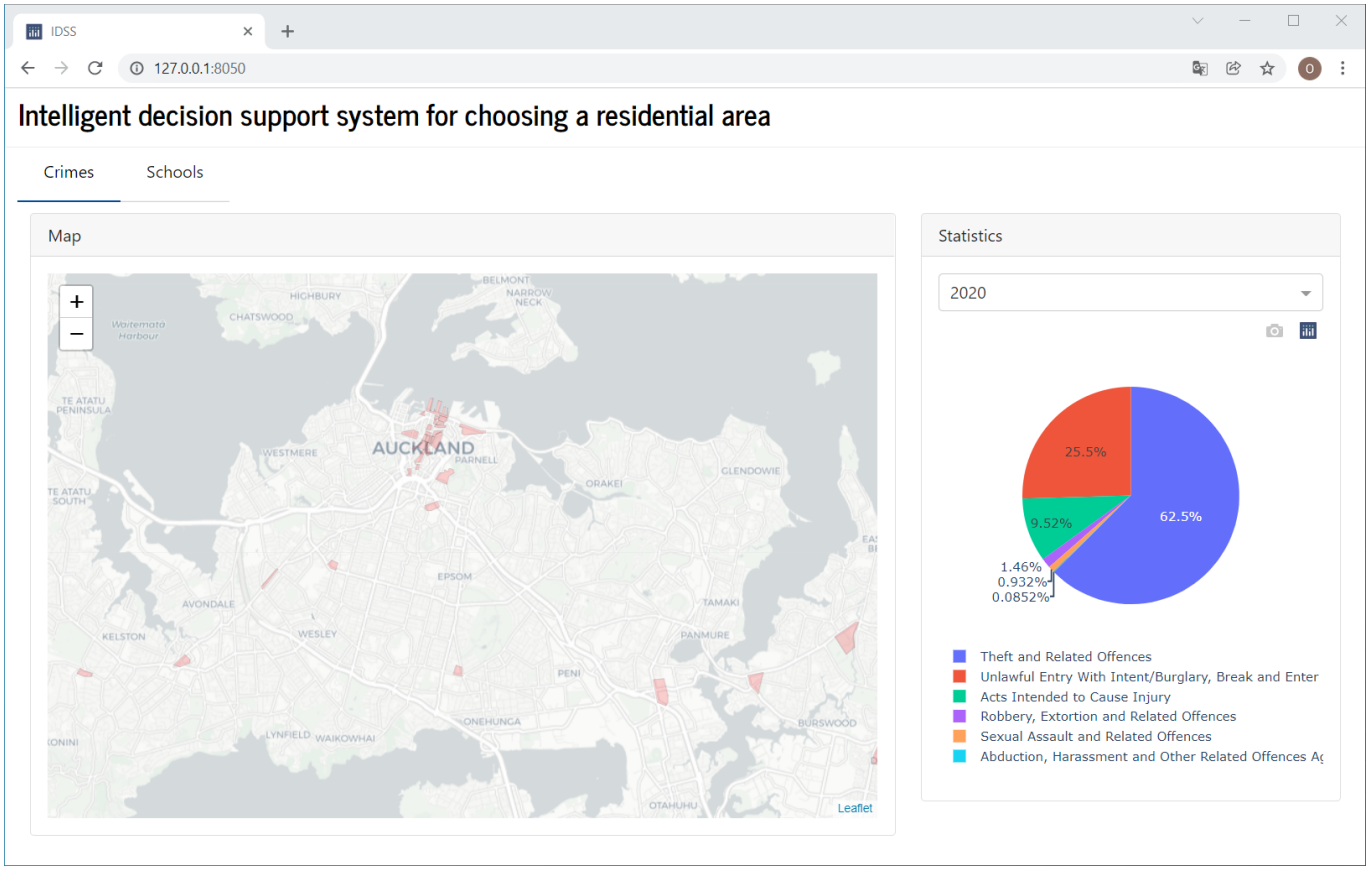

Рисунок 4.2 – Інтелектуальна СППР вибору району проживання

Статистична інформація про злочини та шкільні заклади міста Окленда відображається в окремих вкладках “Crimes” та “Schools”. Перша вкладка містить інтерактивну карту, де виділено райони міста з підвищеним рівнем злочинності, що були отримані в результаті кластеризації. Праворуч за замовчуванням відображається сумарна статистична інформація за попередній рік щодо кількості постраждалих у вигляді кругової діаграми.

Всі злочини розбито на 6 категорій:

- крадіжки (“Theft and Related Offences”);
- незаконне проникнення до житла (“Unlawful Entry With Intent/Burglary, Break and Enter”);
- дії, спрямовані на заподіяння шкоди (“Acts Intended to Cause Injury”);
- вимагання (“Robbery, Extortion and Related Offences”);
- сексуальне насильство (“Sexual Assault and Related Offences”);

- викрадення та домагання (“Abduction, Harassment and Other Related Offences Against a Person”).

Бачимо, що близько 2/3 кількості постраждалих є постраждалими від крадіжок.

Якщо навести курсор миші на відповідну ділянку діаграми, то можна подивитися на саму кількість (див. рис. 4.3).

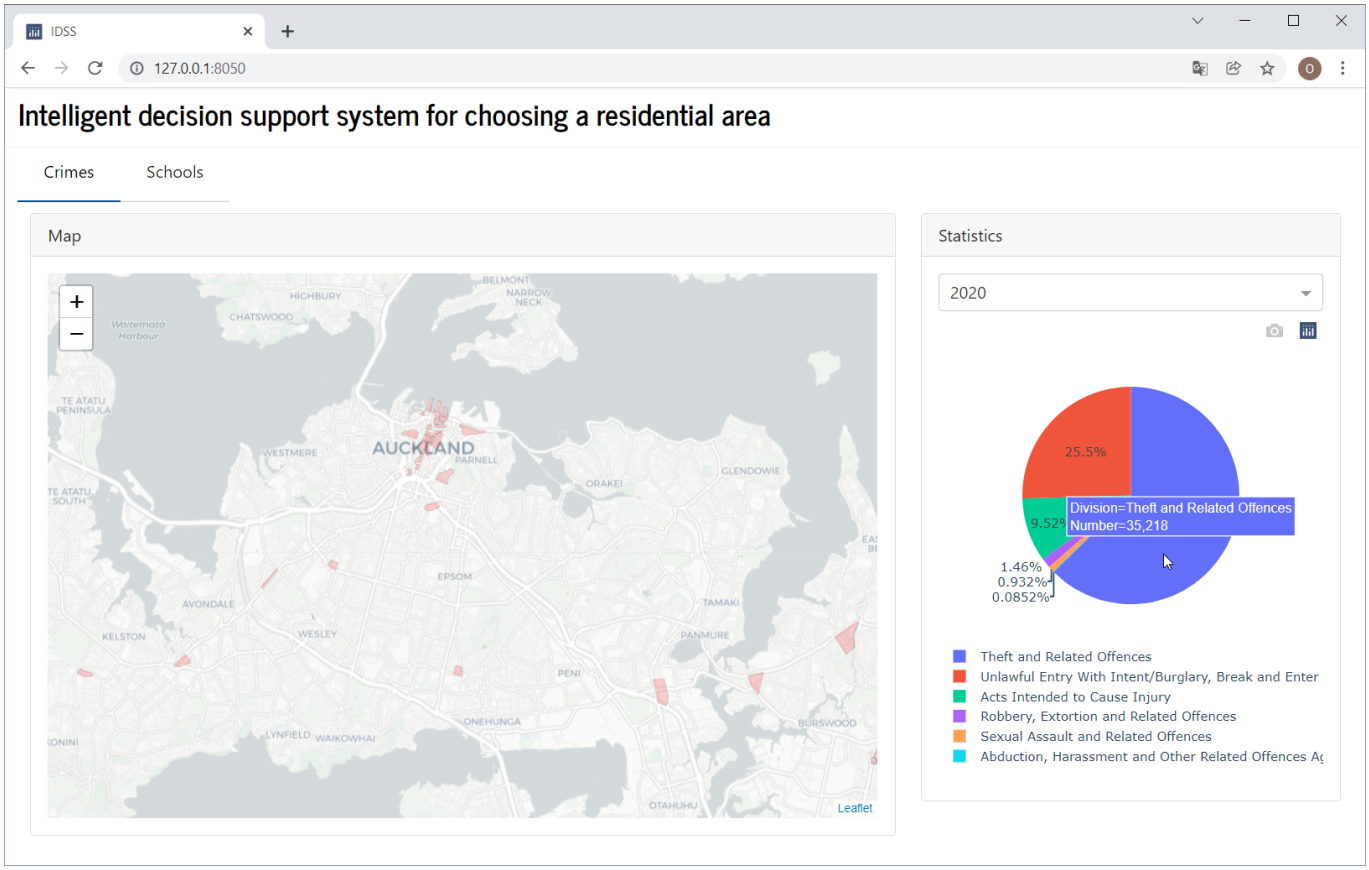

Рисунок 4.3 – Кількість постраждалих внаслідок крадіжок за 2020 рік

Окрім того, можна прибрати відображеннях певних категорій, що призведе до перерахунку відсотків (див. рис. 4.4).

Аналогічну інформацію можна вивести і за попередні роки (починаючи з 2014), вибравши потрібний рік з випадаючого списку (див. рис. 4.5)

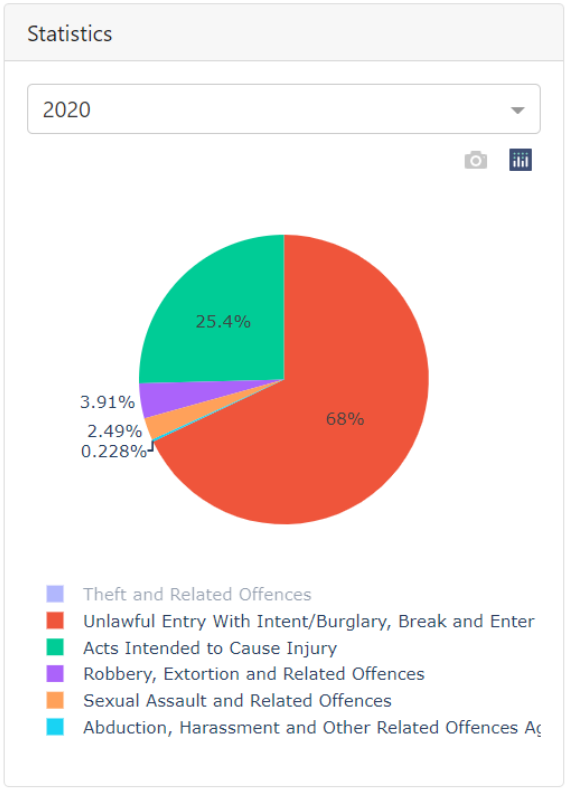

Рисунок 4.4 – Співвідношення кількості постраждалих без категорії “Theft and Related Offences” (2020 рік)

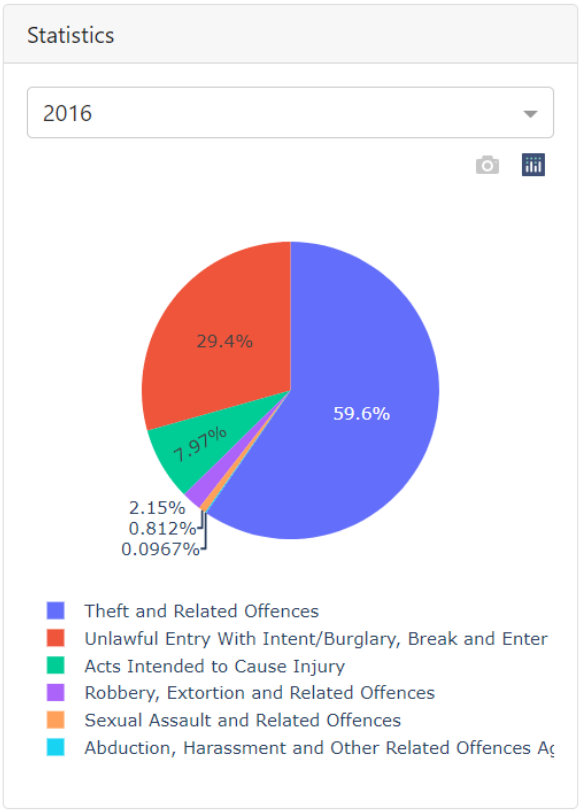

Рисунок 4.5 – Відображення сумарної статистичної інформації за 2016 рік

Як було зауважено раніше, на карті відображено тільки райони з підвищеним рівнем злочинності. Це було зроблено з метою збільшення швидкодії системи. Якщо на карті відмітити всі райони (більше 10000), кожен з яких має складну геометрію (див. рис. 4.6), то інформація на такій карті буде оновлюватися з великими затримками.


```
"type": "FeatureCollection",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
"features": [
{ "type": "Feature", "properties": { "meshblock": 178900 }, "geometry": { "type": "MultiPolygon", "coordinates": [ [ [ [ 174.7040026335280006, -36.7233370160774020 ], [
174.7041675508039870, -36.7235795828332030 ], [ 174.7043254500819960, -36.7237440000970 ], [ 174.7044858669069920, -36.7238822161016 ], [ 174.7049876830440010, -36.7241906498857990 ], [
174.7051575671889910, -36.7243257331447030 ], [ 174.7053181999299910, -36.7244918666296980 ], [ 174.7054494328050110, -36.7246680830412020 ], [ 174.7056063336389970, -36.7250693495923030 ],
[ 174.7057098827970090, -36.7252615165270980 ], [ 174.7060277001770030, -36.7255593329644010 ], [ 174.7062985997999930, -36.7251934495637970 ], [ 174.7065598831540, -36.7249374668808030 ],
[ 174.7069819505830140, -36.7246277333894970 ], [ 174.7073084172700130, -36.7244733168811699 ], [ 174.7080082337639910, -36.7242764099552030 ], [ 174.7083846332589870, -36.7242264006738990
], [ 174.7092359166130110, -36.7242445502032899 ], [ 174.7095822835119920, -36.7242835171360010 ], [ 174.7099130665359890, -36.7243541999404 ], [ 174.7107555162679940,
-36.7246133663773970 ], [ 174.7110544660789910, -36.7247383665755602 ], [ 174.7113376834479940, -36.7248784498992010 ], [ 174.7118658003099940, -36.7251925998601010 ], [
174.7120791167770050, -36.7253535664819 ], [ 174.7122463829929870, -36.7251717169740030 ], [ 174.7125595165310020, -36.7248887003187010 ], [ 174.7128158332299960, -36.7247307670400803 ], [
174.7132756419250090, -36.7245315313229030 ], [ 174.7135852496339980, -36.7244315019068040 ], [ 174.7142136299690090, -36.7243036489564010 ], [ 174.7146242708060130, -36.7242691637069 ],
[ 174.7149340165840010, -36.7277493999622990 ], [ 174.7151006006110090, -36.7292175667585990 ], [ 174.7155130664710040, -36.7310159835340 ], [ 174.7157531668250100, -36.7316726165822020 ], [
174.7165435335400050, -36.7333081171050402 ], [ 174.7174529328370060, -36.7347568283649010 ], [ 174.7177107133429960, -36.7355084985602010 ], [ 174.7177342956809980, -36.7356640666255030 ],
[ 174.7176967068849970, -36.7357813966692040 ], [ 174.7176074501610120, -36.7358738170458 ], [ 174.7175223389455970, -36.7359203276609990 ], [ 174.7172925667809980, -36.7359582495812020
], [ 174.7171698500139880, -36.7359383401757 ], [ 174.7170845553370100, -36.7358996676466990 ], [ 174.7167438721900700, -36.7356545346979 ], [ 174.7166277707989930, -36.7358015237403990 ],
[ 174.7155813880839900, -36.7352530562657980 ], [ 174.7150370659470010, -36.7350595381654020 ], [ 174.7146859769009950, -36.7349549272245990 ], [ 174.7139537664729970, -36.7347802497786020
], [ 174.7127682336540030, -36.7345950999593 ], [ 174.7119477165360020, -36.7344602498493980 ], [ 174.7102038170359890, -36.7341708001322970 ], [ 174.7097229499360080, -36.7341004502034
], [ 174.7092692667970080, -36.7340138662566030 ], [ 174.7083639825740140, -36.7337761495969990 ], [ 174.7079013828889910, -36.7336222831756980 ], [ 174.7075040172769890,
-36.7334556501242990 ], [ 174.7070985835349860, -36.7332568994372010 ], [ 174.7060552492729930, -36.7326870496059 ], [ 174.7039370007209980, -36.7313643994705 ], [ 174.7017845501040090,
-36.7299882833251980 ], [ 174.7012520162300010, -36.7296073828961980 ], [ 174.6999104499900, -36.7285941497888870 ], [ 174.6989830667459960, -36.7278469998431020 ], [ 174.6980405999200060,
-36.7270602165848030 ], [ 174.6973012737149990, -36.7264457167623990 ], [ 174.6954253166719870, -36.7250285162158010 ], [ 174.6958493339480070, -36.7248247327947010 ], [ 174.6983493838990,
-36.7249603826736010 ], [ 174.6986175498429930, -36.7250224667902 ], [ 174.6989717669709990, -36.7251877166685010 ], [ 174.6993100666489910, -36.7250043334459020 ], [ 174.6997981000959900,
-36.7248066329367970 ], [ 174.7012549840810040, -36.7243731328536 ], [ 174.7017506838110140, -36.7242109829447970 ], [ 174.7021696169090030, -36.7239990828162990 ], [ 174.7038313831170060,
-36.7230720496581 ], [ 174.7040026335280006, -36.7233370160774020 ] ] ] ] } },
```


Рисунок 4.6 – Координати меж одного з районів міста (MultiPolygon)

Якщо натиснути ліву клавішу миші в будь-якому місці на території міста, то за географічними координатами точки буде визначено район, якому вона належить, і він відобразиться на карті (див. рис. 4.7).

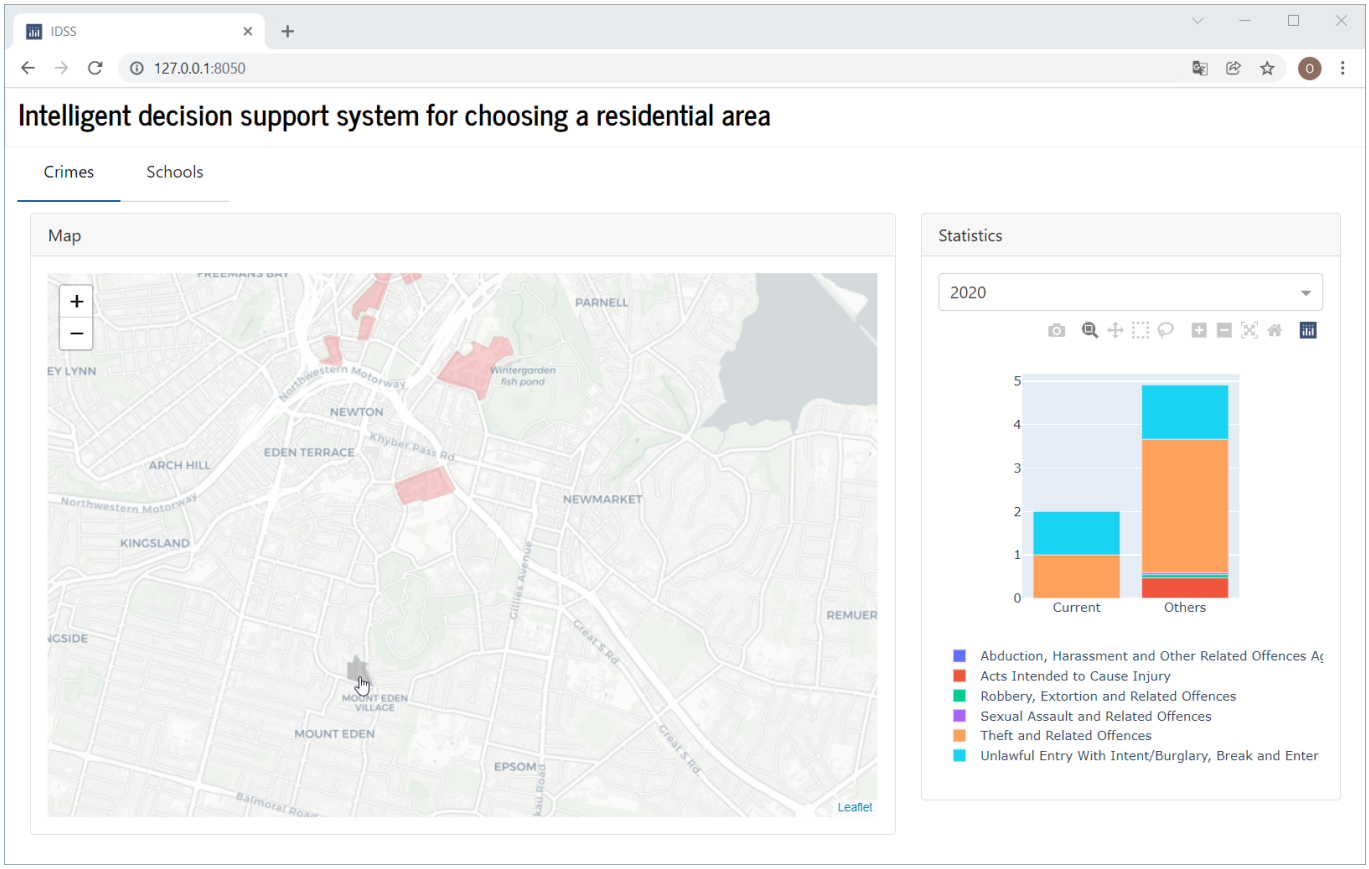

Рисунок 4.7 – Вибір району на карті міста

Окрім того, оновиться статистична інформація справа від карти. Тепер там відображатиметься стовпчикова діаграма (див. рис. 4.7) для порівняння кількості постраждалих у вибраному районі з середнім значенням, обчисленим за всіма іншими районами міста. Інформація виводиться за попередній рік у розрізі категорій злочинів. Як і раніше, можна прибрати відображення певних категорій або взагалі залишити тільки якусь одну для порівняння:

Рисунок 4.8 – Кількість постраждалих внаслідок крадіжок у вибраному районі у порівнянні з середнім показником по всіх інших (2020 рік)

Якщо є потреба, то можна переглянути інформацію за інший рік, вибравши його з випадаючого списку:

Рисунок 4.9 – Статистична інформація для вибраного району за 2019 рік

На рисунку 4.10 відображено таку інформацію для одного з районів, що був виділений як район з підвищеним рівнем злочинності.

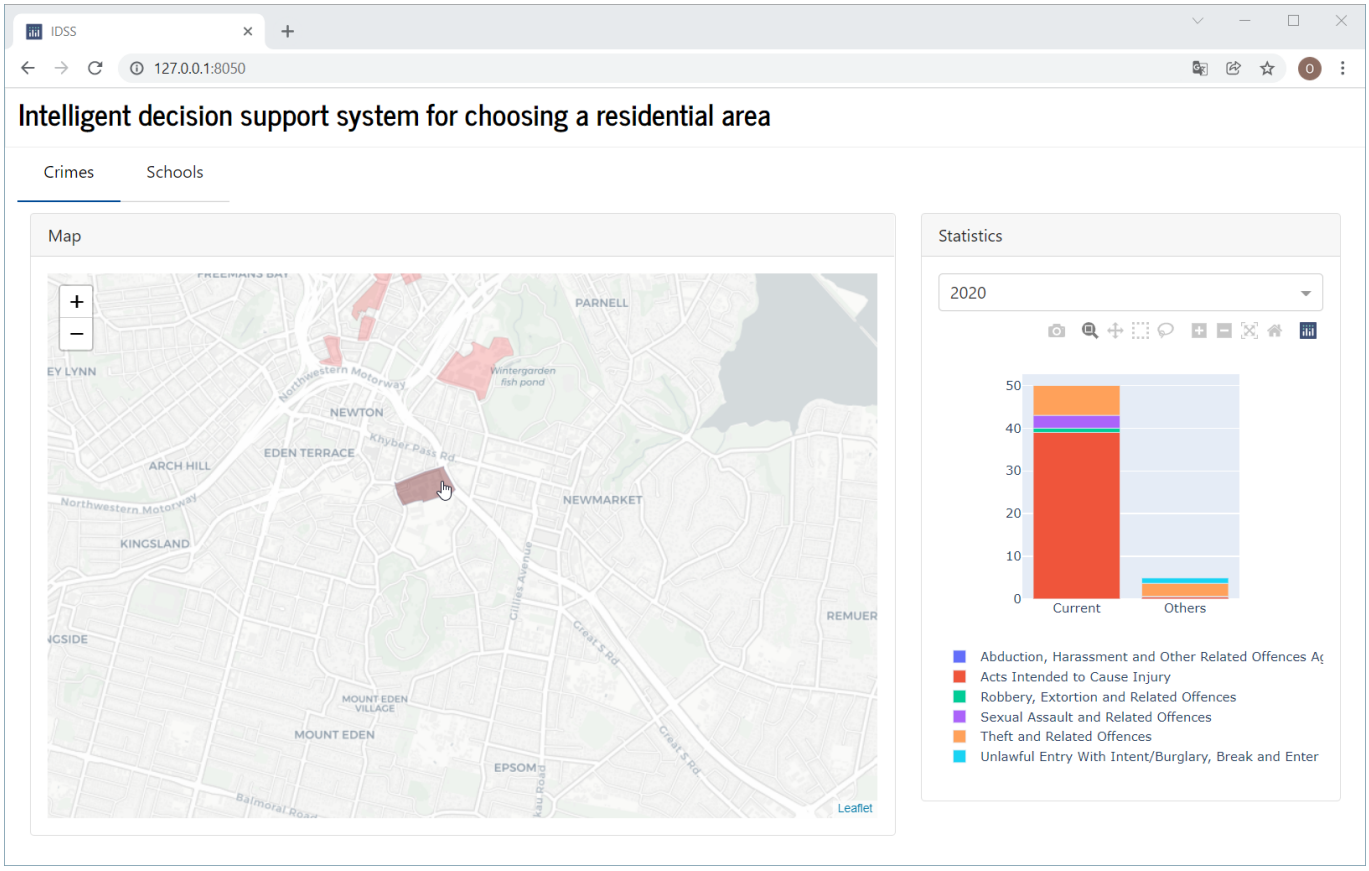

Рисунок 4.10 – Статистична інформація для району з підвищеним рівнем злочинності

Бачимо, що кількість постраждалих у цьому районі протягом 2020 року сильно відрізняється від середнього значення по інших районах міста. Зокрема, кількість постраждалих для категорії “Acts Intended to Cause Injury” (дії спрямовані на заподіяння шкоди) становить 39, коли середнє значення для інших районів є меншим за 0,5 (див. рис. 4.11).

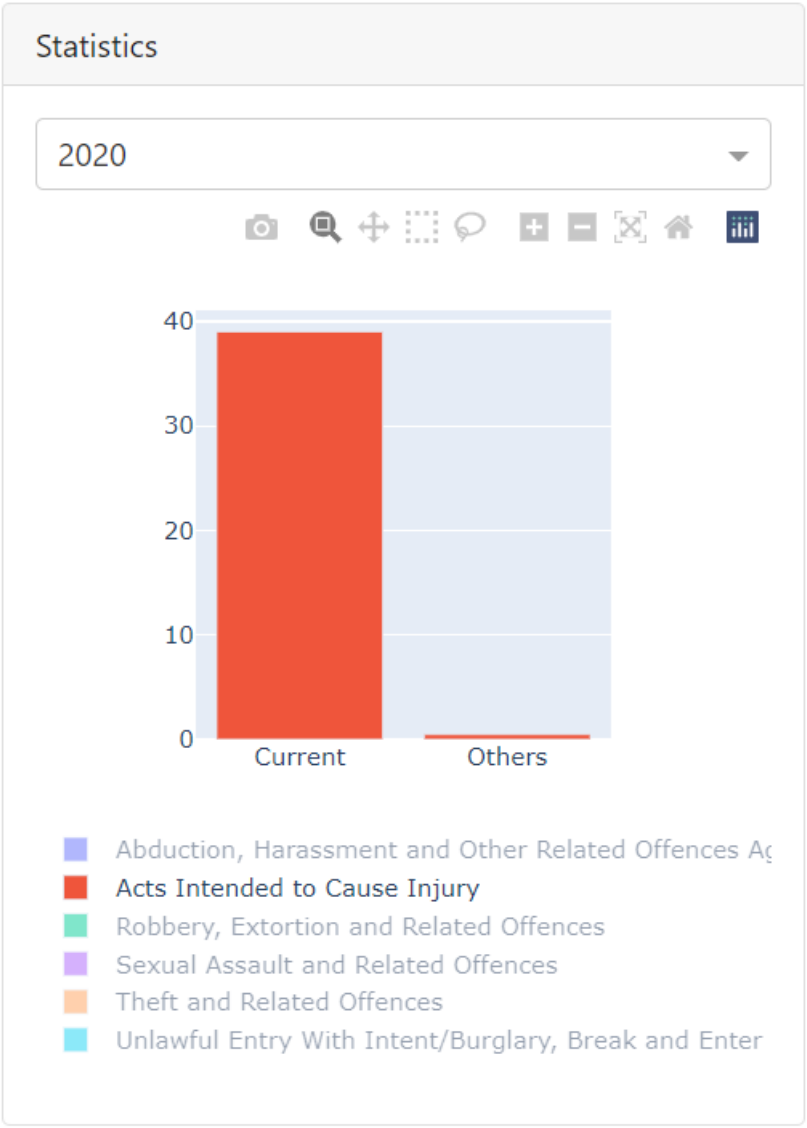

Рисунок 4.11 – Кількість постраждалих для категорії “Acts Intended to Cause Injury” у порівнянні з середнім показником по всіх інших (2020 рік)

В той же час, для сусіднього району значення відрізняються не так суттєво (див. рис. 4.12).

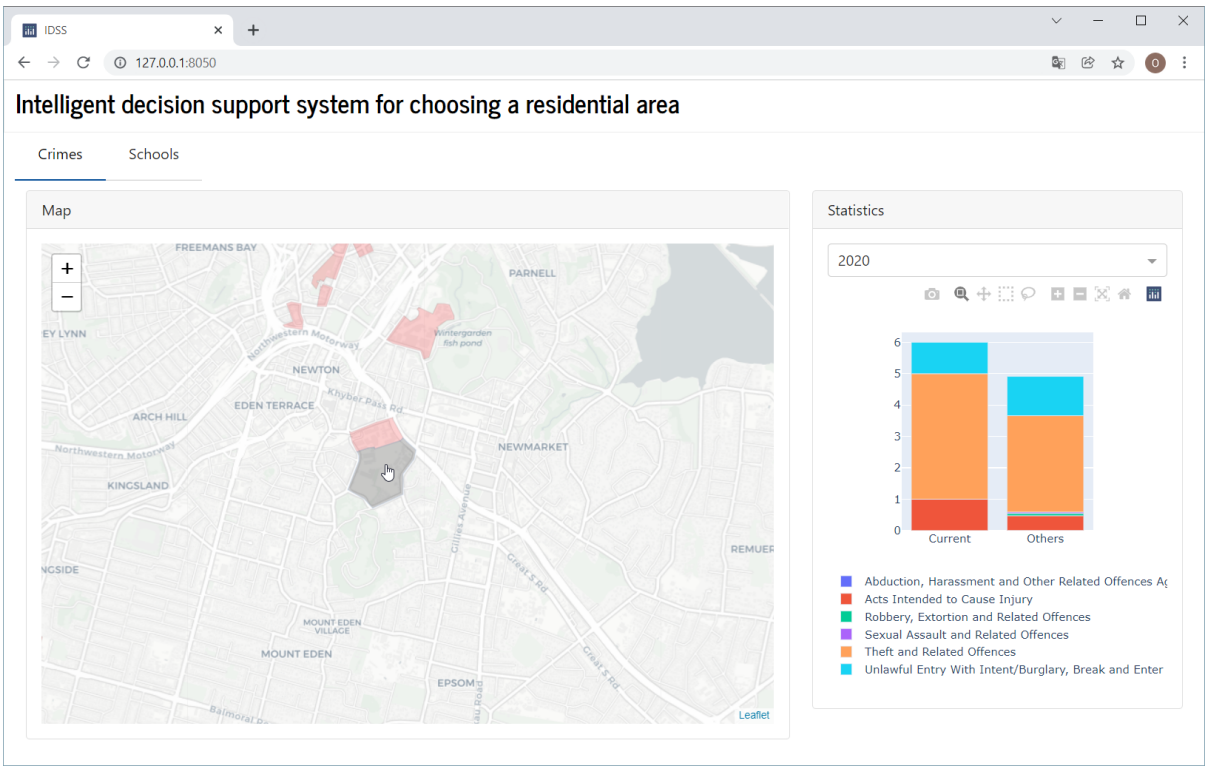

Рисунок 4.12 – Статистична інформація для району, що межує з районом з підвищеним рівнем злочинності

Вкладка “Schools” містить карту шкільних закладів міста Окленда з інформацією щодо контингенту школярів, а саме відсотку європейців, азійців, корінного населення та ін. для кожного закладу (див. рис. 4.13).

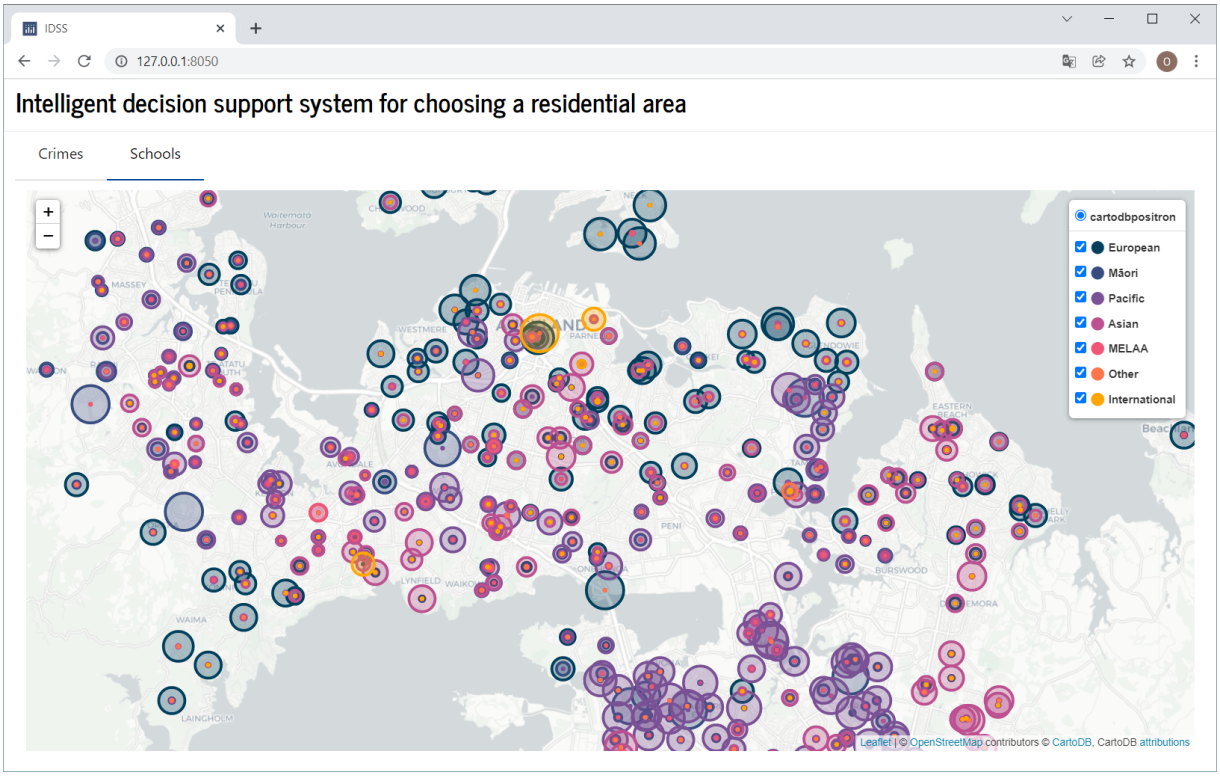

Рисунок 4.13 – Карта шкільних закладів міста Окленда

Карта містить декілька шарів, що накладаються. В кожному шарі школа позначається маркером у вигляді круга заданого кольору, що має радіус, пропорційний відсотку відповідного контингенту. Так, на рисунку 4.14 відображено школи, в яких навчаються європейці (всі інші шари відключені).

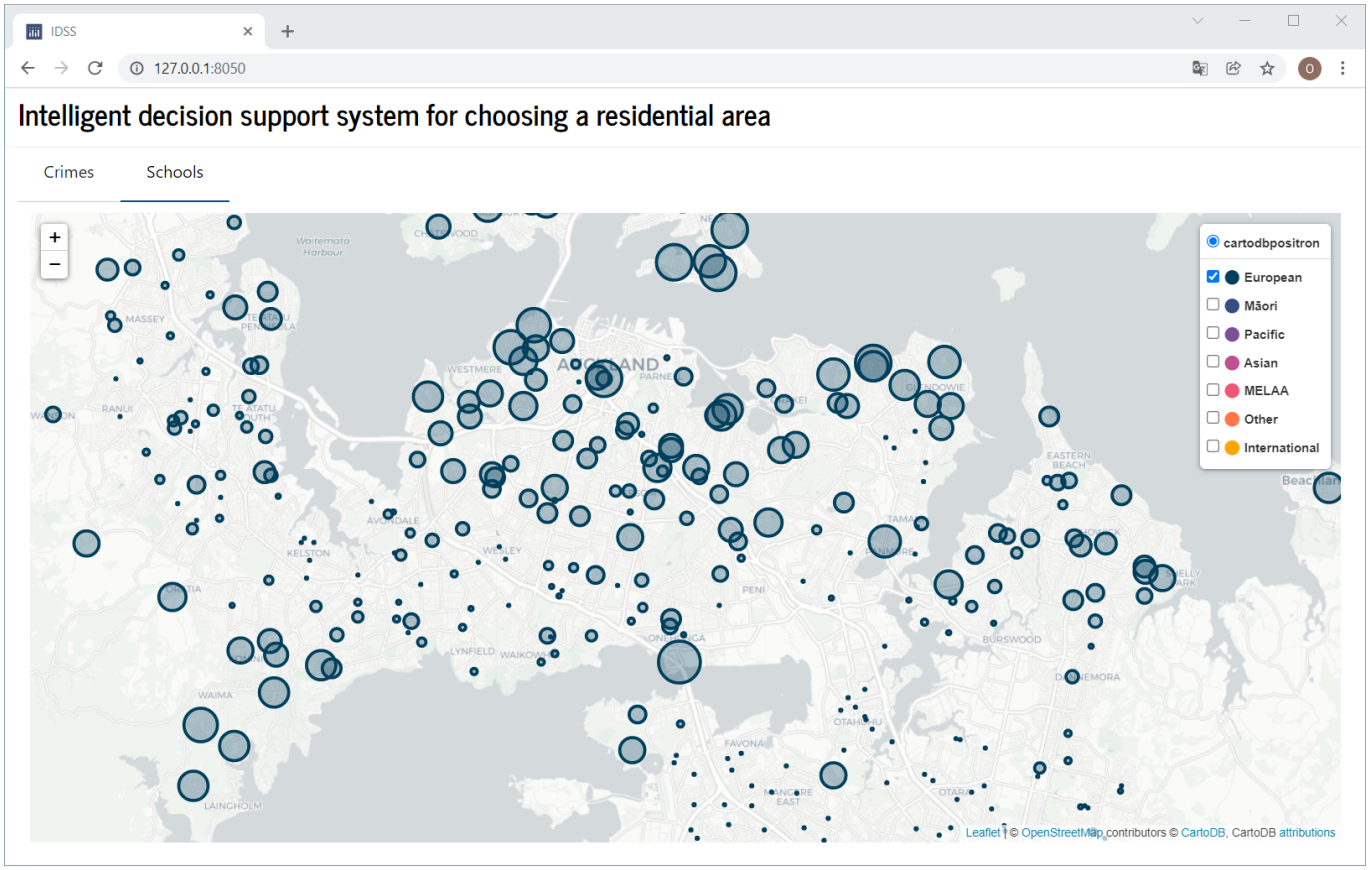

Рисунок 4.14 – Шкільні заклади міста, де навчаються європейці

Райони з кругами більшого радіусу говорять про більший відсоток таких школярів. Якщо натиснути ліву клавішу на маркері, то буде виведено назву школи та відповідний відсоток (див. рис. 4.15). Таку карту можна сприймати як карту розподілу європейців на території міста, якщо припустити, що відсоток школярів в закладі корелюється з відсотком жителів у цьому та близьких до нього районів для заданого контингенту.

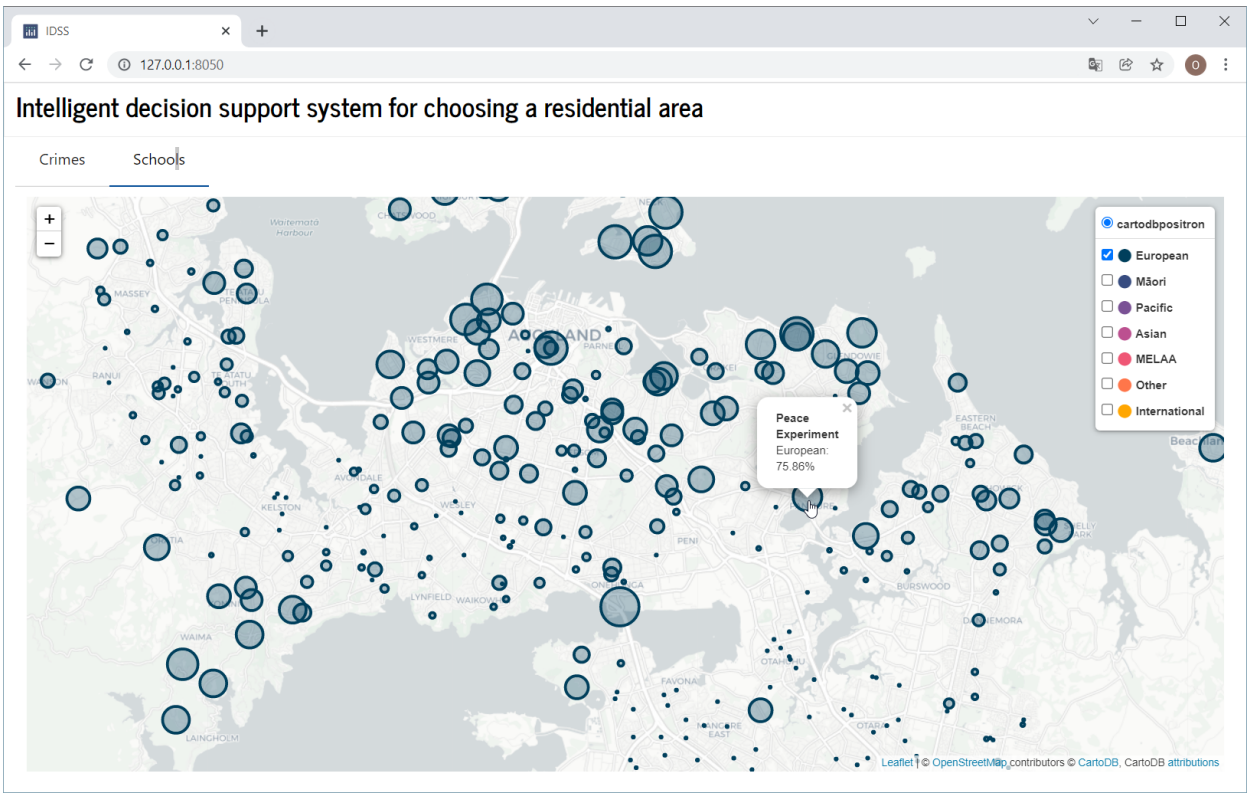

Рисунок 4.15 – Виведення додаткової інформації на карті шкільних закладів

Можна включати відображення інших шарів, тоді кожна школа на карті буде позначатися набором кругів різного радіусу (див. рис. 4.16). Це дозволяє візуально оцінити контингент школярів.

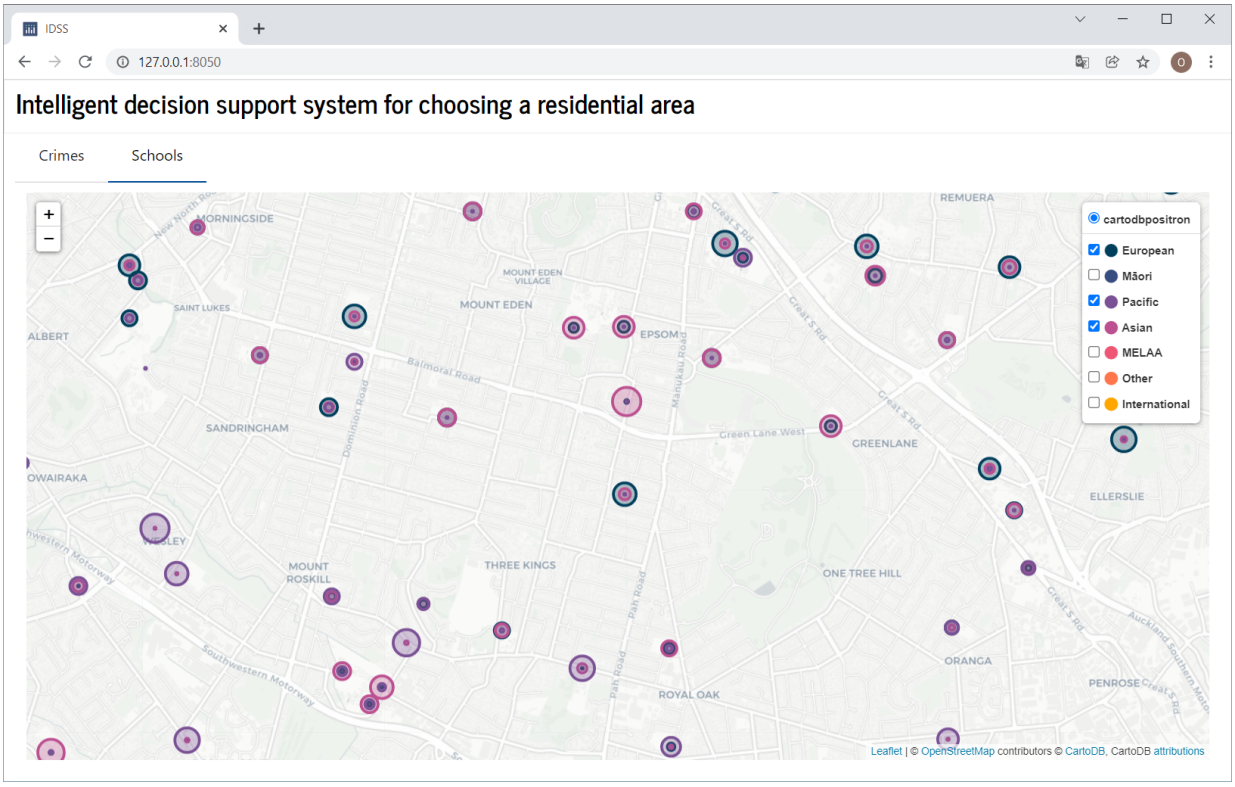

Рисунок 4.16 – Накладання декількох шарів

Висновки до розділу

Програмне забезпечення інтелектуальної системи підтримки прийняття рішень вибору району проживання реалізоване з використанням вебфреймворку Dash, що є інструментом для інтерактивної візуалізації з можливістю інтеграції у вебсистему.

Вибір району проживання здійснюється користувачем системи на основі наявної статистичної інформації щодо злочинів на території району за останніх 7 років у розрізі 6 категорій, порівняльного аналізу цих даних з аналогічними показниками по інших районах та контингенту жителів. Попередньо проведена кластеризація районів міста дозволила виділити райони, що суттєво відрізняються від інших за рівнем злочинності. Відображення їх на карті доповнює загальну картину і сприяє прийняттю виваженого та обґрунтованого рішення під час вибору.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

Таблиця 5.1 – Інформаційна карта проекту

Назва номінації	Інтелектуальна СППР
Назва проекту	Інтелектуальна система підтримки прийняття рішень вибору району проживання
Назва ВНЗ, кафедри, спеціальності	НЛТУ, кафедра інформаційних технологій, 122 Комп'ютерні науки
Прізвище, ім'я, по-батькові автора	Томчук Остап Євгенович
Географія проекту	Місто Окленд, Нова Зеландія
Цілі і задачі проекту	Розробити інтелектуальну СППР вибору району проживання на основі інформації щодо рівня злочинності та інших статистичних даних у розрізі районів міста. Реалізувати кластеризацію районів за наявними ознаками та візуальне представлення отриманих результатів на карті.
Короткий зміст проекту	Розроблена система дозволяє користувачу здійснювати вибір району на основі наявної статистичної інформації щодо злочинів на території району, порівняльного аналізу з аналогічними показниками по інших районах та контингенту жителів. Проведена кластеризація районів міста дозволила виділити ті, що суттєво відрізняються від інших за рівнем злочинності. Відображення їх на карті доповнює загальну картину та сприяє прийняттю виваженого обґрунтованого рішення під час вибору.

Продовження таблиці 5.1

Терміни виконання проекту	11 місяців
Бюджет проекту	Важливою умовою побудови інтелектуальної системи такого роду є наявність великих об’ємів статистичних даних у розрізі районів, а це потребує значних коштів.

5.2 Розроблення ринкової стратегії

Розроблення ринкової стратегії передбачає визначення стратегії охоплення ринку (табл. 5.2).

Таблиця 5.2 – Опис цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Особи, що бажають придбати житло, а також ріелтори
Інтенсивність конкуренції на ринку	Середня
Готовність споживачів сприйняти продукт	Середня
Орієнтовний попит в межах цільової групи	Високий
Простота входу на ринок	Успішному входу на регіональний ринок сприяють простота та зручність розробленої системи, використання інтерактивних засобів візуалізації на противагу табличному представленню інформації.

5.3 Розроблення маркетингової програми

Таблиця 5.3 – Визначення ключових переваг концепції потенційного продукту

Потреба	Вибір району проживання
Вигода, яку пропонує товар	Дозволяє швидко отримати доступну статистичну інформацію для вибраного на карті району та провести порівняльний аналіз з аналогічними показниками по інших районах. Не потребує встановлення на машину користувача. Для повноцінної роботи потрібні тільки доступ до інтернету та браузер.
Ключові переваги перед конкурентами	Інтуїтивно зрозумілий інтерфейс користувача, інтерактивні засоби візуалізації на противагу табличному представленню інформації, використання машинного навчання.

5.4 Вимоги до апаратного та програмного забезпечення

Рекомендується використовувати комп’ютери з характеристиками не гіршими, ніж наведені нижче.

Апаратне забезпечення

- процесор Intel Core i5 з тактовою частотою 1.8 ГГц та вище;
- 4 ГБ оперативної пам’яті, рекомендовано 8 ГБ;
- графічна плата NVIDIA GeForce GT 750M з 2047MB відеопам’яті;
- SSD-накопичувач;
- монітор з мінімальною роздільною здатністю 1024x768.
- інтегрована мережева плата.

Програмне забезпечення

Будь-який браузер з доступом до мережі Інтернет.

Висновки до розділу

Проведений в цьому розділі аналіз дозволяє стверджувати про можливість комерціалізації проекту. Розроблена СППР, адаптована під конкретний регіон, користуватиметься попитом серед людей, що вирішили придбати житло, а також серед ріелторів. Успішному входу на регіональний ринок сприяють простота та зручність розробленої системи, використання інтерактивних засобів візуалізації на противагу табличному представленню інформації, а також використання технологій машинного навчання для знаходження закономірностей у великих об’ємах статистичної інформації.

ВИСНОВКИ

В процесі виконання дипломної роботи магістра було розроблено інтелектуальну систему підтримки прийняття рішень вибору району проживання на прикладі міста Окленда в Новій Зеландії.

Здійснено попередню обробку статистичної інформації по шкільних закладах та реалізовано її візуалізацію на карті з використанням бібліотеки Folium.

Проведено кластеризацію районів на основі інформації щодо злочинів на території міста за останні 7 років та, як результат, виділено райони, що характеризуються підвищеним рівнем злочинності.

Програмний продукт, що створено на базі вебфреймворку Dash, має інтуїтивно зрозумілий інтерфейс і дозволяє швидко отримати доступну статистичну інформацію для вибраного на карті району та провести порівняльний аналіз з аналогічними показниками по інших районах.

Основні результати роботи опубліковано в матеріалах III науково-практичної конференції студентів, аспірантів та молодих вчених «Комп'ютерне моделювання та інформаційні технології» [14].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація для бібліотеки Folium. – Режим доступу: http://python-visualization.github.io/folium/ (дата звернення: 30.11.2021).
2. Спецификация формата GeoJSON. URL: https://gis-lab.info/docs/geojson_ru.html (дата звернення: 30.11.2021).
3. Офіційна документація для бібліотеки GeoPandas. – Режим доступу: https://geopandas.org/en/stable/docs.html# (дата звернення 30.11.2021).
4. Навчальні посібники по Pandas, Numpy, Matplotlib і Scikit-learn. – Режим доступу: http://scipy-lectures.org/index.html (дата звернення 30.11.2021).
5. Офіційна документація для бібліотеки Pandas. – Режим доступу: https://pandas.pydata.org/pandas-docs/stable/index.html (дата звернення 30.11.2021).
6. Victimisation Time and Place [Електронний ресурс]. – Режим доступу: https://www.police.govt.nz/about-us/publications-statistics/data-and-statistics/policedatanz/victimisation-time-and-place (дата звернення 30.11.2021).
7. Метод главных компонент [Електронний ресурс]. – Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=Метод_главных_компонент (дата звернення 30.11.2021).
8. Principal Component Analysis. in 3 Simple Steps. – Режим доступу: https://sebastianraschka.com/Articles/2015_pca_in_3_steps.html (дата звернення 30.11.2021).
9. Overview of clustering methods [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/modules/clustering.html (дата звернення 30.11.2021).
10. Silhouette (clustering) [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Silhouette_(clustering) (дата звернення 30.11.2021).
11. Лутц, М. Изучаем Python, 4-е издание / М. Лутц. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
12. Плас, Дж. Вандер. Python для сложных задач: наука о данных и машинное обучение / Дж. Вандер Плас. – СПб.: Питер, 2018. – 576 с.

13. Офіційна документація для бібліотеки Dash [Електронний ресурс]. – Режим доступу: https://dash.plot.ly (дата звернення 30.11.2021).
14. Томчук О. Є. Інтелектуальна система підтримки прийняття рішень вибору району проживання / О. Є. Томчук, Ю. С. Процик // Комп'ютерне моделювання та інформаційні технології: III науково-практична конференція студентів, аспірантів та молодих вчених (Львів, 14-16 жовтня 2021р.). – Львів: кафедра інформаційних технологій НЛТУ України, 2021, с. 18-20.

ДОДАТКИ

ДОДАТОК А

Файл app.py

```
import numpy as np
import pandas as pd
import geopandas as gpd
from shapely import wkt
from shapely.geometry import Point
import json
from dash import Dash
from dash import html
from dash import dcc
import dash_bootstrap_components as dbc
import dash_leaflet as dl
import dash_leaflet.express as dlx
from dash.dependencies import Input, Output
from dash_extensions.javascript import assign
import plotly.express as px

mb_geometry = pd.read_csv("mb_geometry.csv")
mb_geometry['geometry'] = mb_geometry['geometry'].apply(wkt.loads)
gdf = gpd.GeoDataFrame(mb_geometry, crs = 'epsg:4326')

df_regions = pd.read_csv("region_data.csv", header=[0,1],
index_col=[0])
df_regions = df_regions.reset_index()

X_MAP, Y_MAP = 174.77602536007595, -36.877834116025944

def mesh_by_point(coords):
    coords_float = [float(i) for i in coords]
    p = Point(coords_float)
```

```
    filter_gdf = gdf[gdf.apply(lambda x: p.within(x['geometry']),
axis=1)]
    if (not filter_gdf.empty):
        return filter_gdf['meshblock'].iloc[0]

def default_figure(year):
    d = df_regions.loc[:, year].sum().reset_index()
    d.columns = ['Division', 'Number']
    fig = px.pie(d.reset_index(), values= 'Number', names =
'Division')
    fig.update_layout(
        legend=dict(orientation="h")
        )
    return fig

geojson = json.load(open("crimes_anomaly.json"))
geobuf = dlx.geojson_to_geobuf(geojson)
options_anomaly = dict(fillColor="#f51720", weight=0.1,
color='#000000', fillOpacity=0.2)
geojson = dl.GeoJSON(data=geobuf, format="geobuf", id="geojson",
options=options_anomaly)


base_map =
dl.TileLayer(url="https://{s}.basemaps.cartocdn.com/rastertiles/light_
all/{z}/{x}/{y}.png")

my_map = dl.Map(children=[base_map, geojson], center=(Y_MAP, X_MAP),
zoom=12,
             style={'width': '100%', 'height': '70vh', 'margin':
"auto", "display": "block"}, id="map")

YEARS = ['2014', '2015', '2016', '2017', '2018', '2019', '2020']

tools_card = dbc.Card([
    dbc.CardHeader("Statistics"),
```

```
    dbc.CardBody([
        html.Div(id='clickdata'),
        dcc.Dropdown(
        id="dropdown",
        options=[{"label": x, "value": x} for x in YEARS],
        value=YEARS[-1],
        clearable=False,
    ),
        dcc.Graph(id="bar-chart")
    ])
])

map_card = dbc.Card([
    dbc.CardHeader("Map"),
    dbc.CardBody([my_map])
])

header = dbc.Navbar(
    dbc.Container([
        html.H3("Intelligent decision support system for choosing a
residential area")
    ], fluid=True),
    dark=False, color="light", sticky="top"
)

tab_style = {
    'width': 'inherit',
    'border': 'none',
    'boxShadow': 'inset 0px -1px 0px 0px lightgrey',
    'background': 'white',
    'paddingTop': 0,
    'paddingBottom': 0,
    'height': '42px',
}

tab_selected_style = {
```

```
    'width': 'inherit',
    'boxShadow': 'none',
    'borderLeft': 'none',
    'borderRight': 'none',
    'borderTop': 'none',
    'borderBottom': '2px #004A96 solid',
    'background': 'white',
    'paddingTop': 0,
    'paddingBottom': 0,
    'height': '42px',
}

tab1_content = dbc.Container([
    dbc.Row([
            dbc.Col([map_card], width=8),
            dbc.Col([tools_card], width=4)
        ]),
    ], fluid=True, style={'padding-top': '10px'})

tab2_content = dbc.Container([
    html.Iframe(srcDoc = open('Auckland_schools.html', 'r').read(),
                width="100%", height='600')
    ], fluid=True, style={'padding-top': '10px'}),

tabs = dcc.Tabs(
    [
        dcc.Tab(tab1_content, label="Crimes", style=tab_style,
selected_style=tab_selected_style),
        dcc.Tab(tab2_content, label="Schools", style=tab_style,
selected_style=tab_selected_style),
    ]
)

app = Dash(external_stylesheets=[dbc.themes.JOURNAL,
dbc.icons.BOOTSTRAP])
app.title = "IDSS"
```

```
app.layout = html.Div([
    header,
    dbc.Container([
        tabs
    ], fluid=True, style={'padding-top': '10px'}),
])

@app.callback([Output("clickdata", "children"), Output("bar-chart",
"figure"), Output("map", "children")],
              [Input("map", "click_lat_lng"), Input("dropdown",
"value")])

def map_click(click_lat_lng, year):
    if click_lat_lng:
        MESHBLOCK = mesh_by_point([click_lat_lng[1],
click_lat_lng[0]])
        current_mb = gdf[gdf['meshblock'] == MESHBLOCK]
        current_geojson = eval(current_mb.to_json())
        current_geobuf = dlx.geojson_to_geobuf(current_geojson)
        options = dict(fillColor="#000000", weight=0.1,
fillOpacity=0.2)
        current_geojson = dl.GeoJSON(data=current_geobuf,
format="geobuf", id="current_geojson", options=options)

        if (MESHBLOCK):
            current_meshblock = np.where(df_regions['Meshblock'] ==
MESHBLOCK , "Current", 'Others')
            data_year_meshblock = df_regions.loc[:,
year].groupby(current_meshblock).mean()
            fig = px.bar(data_year_meshblock)
            fig.update_layout(
                legend_title_text='',
            legend=dict(orientation="h")
            )
            fig.update_yaxes(title='')
```

```
            fig.update_xaxes(title='')

            return "", fig, [base_map, geojson, current_geojson]
        else:
            return "No information", px.bar(), [base_map, geojson]
    else:
        return "", default_figure(year), [base_map, geojson]


if __name__ == '__main__':
    app.run_server(debug=False, use_reloader=False)
```

Попередня обробка даних (Файл Jupyter Notebook)

```
#!/usr/bin/env python
# coding: utf-8
# # Інтелектуальна система підтримки прийняття рішень вибору району проживання
# In[1]:
import numpy as np
import pandas as pd
import folium
from folium.plugins import MarkerCluster
from tqdm import notebook
from sklearn import metrics
from sklearn.cluster import MiniBatchKMeans, KMeans, AgglomerativeClustering, SpectralClustering, AffinityPropagation
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
RANDOM_STATE = 17
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
get_ipython().run_line_magic('config', "InlineBackend.figure_format = 'retina'")
```

```
# In[2]:
import warnings
warnings.filterwarnings('ignore')
import geopandas as gpd
from shapely import wkt
from shapely.geometry import Point
import folium

# ### Police Data
# In[3]:
df_raw = pd.read_csv("data/nz_police_data.csv", sep="\t",
engine="python", encoding = "utf-16", error_bad_lines=False)
df_raw.head()

# ### Попередня обробка даних
# In[4]:
df = df_raw.drop(['Number of Records', 'Table 1', ' Month Year', 'Year
Month (copy 2)'], axis=1)
df['Area Unit'] = df['Area Unit'].str.rstrip('.')
df['Territorial Authority'] = df['Territorial
Authority'].str.rstrip('.')
# set index
df['Year Month'] = pd.to_datetime(df['Year Month']).dt.to_period('M')
df = df.set_index('Year Month')
# filter Auckland
df = df[df['Territorial Authority'] == 'Auckland']
df.head(3) # 1294334 записів

# In[84]:
regions_data = df.pivot_table(values='Victimisations',
index='Meshblock', columns=[df.index.year, 'ANZSOC Division'],
aggfunc='sum', fill_value=0)
regions_data.sample(5)

# In[66]:
```

```
# Save data to csv-file
regions_data.to_csv("region_data.csv")

# In[149]:
# meshblock-2021
mb = pd.read_csv("data/statsnzmeshblock-2021/meshblock-2021-
generalised.csv",
                 sep=",", usecols=[0,1], names=['geometry',
'meshblock'], header=1)
mb = mb[~mb['geometry'].isna()]
mb['geometry'] = mb['geometry'].apply(wkt.loads)
mb.head(3)

# In[150]:
mb_geometry = regions_data.merge(mb, how='inner', left_on='Meshblock',
right_on='meshblock')
mb_geometry = mb_geometry.iloc[:,-2:]

# In[151]:
# Save geometry to csv-file
mb_geometry.to_csv("mb_geometry.csv", index=False)

# ### Задача кластеризації
# In[155]:
X = regions_data.copy()

# In[156]:
# Масштабування даних (стандартизація)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# In[157]:
# Зниження розмірності (90% дисперсії)
pca = PCA(n_components=0.9)
X_pca = pca.fit_transform(X_scaled)
X.shape, X_pca.shape
```

```
# In[9]:
# Вибір моделі кластеризації (розбиття) та кількості кластерів
max_num = 10
results = pd.DataFrame(index=['K-means', 'Agglomerative'],
columns=list(range(2, max_num+1)))
for k in notebook.tqdm(range(2, max_num+1)):
    algorithms = []
    algorithms.append(KMeans(n_clusters=k, random_state=RANDOM_STATE,
n_init=100))
    algorithms.append(AgglomerativeClustering(n_clusters=k,
linkage='ward'))
    data = []
    for algo in notebook.tqdm(algorithms):
        algo.fit(X_pca)
        data.append(metrics.silhouette_score(X, algo.labels_))
    results[k] = data
results.columns = pd.MultiIndex.from_product([['Silhouette'],
results.columns])
results

# In[158]:
N_CLUSTERS = 2
algorithms = KMeans(n_clusters=N_CLUSTERS, random_state=RANDOM_STATE,
n_init=100).fit(X_pca)
print('Silhouette: %.3f' % metrics.silhouette_score(X,
algorithms.labels_))

# In[165]:
cluster_labels = algorithms.labels_
unique, counts = np.unique(cluster_labels, return_counts=True)
dict(zip(unique, counts))

# In[172]:
regions_data['labels'] = cluster_labels
regions_data.head()
```

```
# In[100]:
regions_data.to_csv("region_data.csv")

# In[194]:
temp = regions_data[regions_data['labels'] ==
1].reset_index()[['Meshblock', 'labels']]
crimes_anomaly = temp.merge(mb, how='inner', left_on='Meshblock',
right_on='meshblock')
crimes_anomaly = crimes_anomaly.iloc[:,-2:]
crimes_anomaly = gpd.GeoDataFrame(crimes_anomaly, crs = 'epsg:4326')
# Save crimes_anomaly to json-file
crimes_anomaly.to_file("crimes_anomaly.json", driver="GeoJSON")

# ### schooldirectory-02-12-2020-083021
# In[4]:
df_school = pd.read_csv("data/schooldirectory-02-12-2020-083021.csv")
df_school = df_school[df_school['Education_Region'] == 'Auckland']
df_school.head()

# In[161]:
percent = 100 *
df_school.loc[:,'European':'International'].apply(lambda s: s /
df_school['Total'])
percent.fillna(0, inplace=True)
percent.head(3)

# In[162]:
data = pd.concat([df_school[['Org_Name','Latitude', 'Longitude']],
percent], axis=1)
data.dropna(inplace=True)
data.head(3)

# In[163]:
```

```
gdf2 = gpd.GeoDataFrame(data,
geometry=gpd.points_from_xy(data['Longitude'], data['Latitude']), crs
= 'epsg:4326')
gdf2.head()

# In[164]:
y_map, x_map = data[['Latitude', 'Longitude']].mean().values
mymap = folium.Map(location=[y_map, x_map], zoom_start=12,
tiles='cartodbpositron')
color_list =
['#003f5c','#374c80','#7a5195','#bc5090','#ef5675','#ff764a','#ffa600'
]
percent_columns = data.columns[3:].tolist()
for column, color in zip(percent_columns, color_list):
    lgd_text = '<span style="color: transparent;  text-shadow: 0 0 0
{}; ">&#9899</span> {}'.format(color, column)
    fg = folium.FeatureGroup(lgd_text)
    mymap.add_child(fg)
    df = data[data[column] > 0]
    for lat, lon, name, percent in zip(df['Latitude'],
df['Longitude'], df['Org_Name'], df[column]):
        popup_text = "<b>{}</b><br>{}: {}%".format(name, column,
round(percent,2))
        folium.CircleMarker(
            [lat, lon],
            radius=0.2*percent,
            color=color,
            fill_color=color,
            popup=popup_text,
            fill=True,
            fill_opacity=0.3
            ).add_to(fg)#.add_to(marker_cluster)
folium.LayerControl(collapsed=False).add_to(mymap)
mymap.save('Auckland_schools.html')
mymap
```