**Bushfire Brigade**

DATA MANAGEMENT PLAN 

**Group TA30 - Next Gen Innovators:**

*Chandara Khvan (33527245)*

*Maherali R Vijapura (33633215)*

*Manya Bhatia (30488575)*

*Siramade Apivirasirikul (33037078)*

*Thinithi Bulathsinghala (27523306)*

**

**Table of Contents**

1. Dataset Overview.............................................................................2

2. Data Usage......................................................................................3

3. Data Preparation..............................................................................4

4. Data Storage....................................................................................5

5. Database Design..............................................................................6

6. Data Analytics..................................................................................7

7. Ethical, Legal & Privacy Issues.......................................................12

1



**1. Dataset Overview**

| ***Dataset*** | ***Source Physical Access*** | ***Granula***  ***rity*** | ***Licence*** | ***User***  ***Story*** |
| --- | --- | --- | --- | --- |
| ***Dataset 01: School***  ***locations in Victoria*** | ***https://discover.data.***  ***CSV***  ***vic.gov.au/dataset/sc***  ***hool-locations-2023/***  ***resource/92fdd072-4***  ***666-4cc6-a28a-749c***  ***826297a7*** | ***High*** | ***Open source***  ***data*** | ***U4.1,***  ***U4.2*** |
| ***Dataset 02: Enrollments in schools in Victoria***  ***Dataset 03: Bushfire risk school registry*** | ***CSV***  ***https://discover.data.vi***  ***c.gov.au/dataset/all-sc***  ***hools-fte-enrolments-f***  ***eb-2023-victoria/resou***  ***rce/64c14b8a-dbe9-4c***  ***28-9f75-c5689a727b8***  ***0***  ***https://www.vic.gov.a***  ***CSV***  ***u/bushfire-risk-regis***  ***ter-barr*** | ***High***  ***High*** | ***Open source***  ***data***  ***Open source***  ***data*** | ***U1.9***  ***U4.1,***  ***U4.2*** |
| ***Dataset 04: Bushfire***  ***scar history*** | ***https://discover.data.vi***  ***shapefile***  ***c.gov.au/dataset/fire-h***  ***istory-records-of-fires***  ***across-victoria-showin***  ***g-the-fire-scars1*** | ***High*** | ***Open source***  ***spatial data*** | ***U4.1,***  ***U4.2*** |

2



**2. Data Usage**

We plan to use the above mentioned datasets as below:

**● School locations in Victoria:** The school locations dataset provides detailed

information about schools across Victoria, including their coordinates, address, and

contact details. This data can be used to create interactive maps that visualise each

school's proximity to bushfire-prone areas. By overlaying bushfire risk zones on these

maps, students can better understand the specific risks their school faces and level of

alertness they need to maintain. Additionally, the dataset can inform the creation of

tailored bushfire preparedness plans and evacuation routes for each school, ensuring

that educational content is relevant and practical for the students' local environment.

**● Enrollments in schools in Victoria:** The enrollment data for schools in Victoria

provides a detailed breakdown of student numbers across different grade levels within

each school. This data will be used to assess the potential impact of bushfire education

programs on the student population. By analysing enrollment figures, the program can

tailor educational content to specific age groups and ensure that resources are

allocated effectively. Additionally, understanding the distribution of students across

grades helps in effective targeting, designing age-appropriate materials and activities,

ensuring that the bushfire education program is both engaging and relevant for

students at different stages of their schooling.

**● Bushfire risk school registry:** The bushfire risk school registry, which includes the

Fire Risk Category for 2023-24, will be used to identify and prioritise schools and

educational facilities that are at higher risk of bushfire incidents. By categorising

schools according to their fire risk, the bushfire education program can focus on

delivering targeted content to those in high-risk areas, ensuring that students and staff

are well-prepared for potential emergencies. An interactive map will be developed for

the first time with the schools overlaid on bushfire scar areas with kid friendly insights

highlighting the level of alert required.

3



● **Bushfire scar history:** This dataset includes information on areas that have been

affected by bushfires in the past. By analysing historical bushfire scars, we can

identify patterns and trends in bushfire activity over time. This information will be

used to create interactive child-friendly visualisations that show how bushfire scars

have changed over time. These visualisations can help students understand the impact

of bushfires on the environment and recognize areas that have experienced multiple

incidents.

**3. Data Preparation**

The data preparation process began by examining the entities and attributes contained within each CSV and shapefile file. We have 5 datasets that we need to clean and combine in order to establish the database for the project. Upon examination, we found that the datasets needed to be filtered based on the target audience, which are kids ages between 8 to 12 years old, living in the bushfire prone area. This filtering process will make the database lighter and improve querying efficiency.

To clean and wrangle the data, we used Python and the pandas library. One significant benefit of using pandas is its efficiency in handling tabular data, which is the format we are working with. We used pandas to clean column names, transform enrollment data into a long format, and perform initial data wrangling. Additionally, QGIS was employed to read and convert shapefile data into GeoJSON format for mapping historical bushfire risks that will be later used for mapping the fire-scaring history.

A crucial part of the data preparation involved geocoding school addresses to obtain latitude and longitude coordinates. If the latitude and longitude were not provided in the data, we utilised a geocoding function to acquire these coordinates. Schools from the datasets were matched based on their names and addresses, and latitude and longitude mappings were validated through name-address similarity mapping. Schools with matching identifiers were cross-referenced for accuracy, and discrepancies were addressed by checking coordinates against Google Maps. The cleaning is mainly to connect all datasets, those including School,

4



Risk Category, School\_BARR, Enrollment, the School\_No (school number). For schools without existing identifiers, new keys were created to ensure a seamless integration of data.

**4. Data Storage**

Creating a robust data storage plan for a project involving NEXT.js, Prisma, and PostgreSQL requires a detailed approach to ensure efficient data management, security, and scalability. The first step is to define the data requirements and design the database schema based on the provided ERD. This schema includes attributes such as `School\_No`, `School\_Name`, `Risk\_Cat\_No`, and `Enroll\_Code`, among others, with well-defined relationships like the connection between `School\_Barr` and `Risk\_Category`, ensuring data integrity.

The PostgreSQL database setup is facilitated using Prisma, where models corresponding to the tables in the ERD are defined in the Prisma schema. Each model includes attributes relevant to its entity, such as `School\_No` and `Risk\_Cat\_No` in `School\_Barr`, and these models are then used to generate the necessary tables in the PostgreSQL database. The Prisma schema's attributes ensure that all necessary data points are captured and maintained accurately.

Integration with NEXT.js involves creating a `prismaClient` within the application, which interacts with the database to perform CRUD operations. API routes are developed in NEXT.js to handle operations for each entity, such as `School`, `School\_Barr`, `Enrollment`, and `Risk\_Category`, with these routes utilizing the specific attributes defined in the schema, like `School\_No` for identifying schools and `Risk\_Cat\_No` for categorizing risk levels.

Security measures are critical in this plan, where environment variables securely store database credentials, and API routes are protected with authentication and authorization strategies like JWT or OAuth. Each operation within these routes ensures the security of the attributes and the overall data integrity.

Finally, deployment is carried out on platforms such as Vercel or AWS, with the PostgreSQL database hosted securely, possibly on AWS RDS. Performance is optimized using Prisma’s

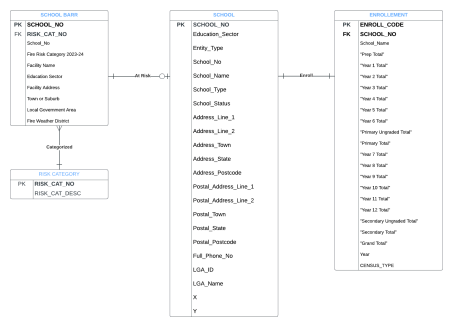
5



query optimization features and indexing the database on crucial attributes like `School\_No` and `Risk\_Cat\_No` to enhance retrieval times. Regular monitoring of both the application and database ensures that the attributes and data remain secure, accurate, and efficiently managed, fulfilling the goals of scalability and robust performance in the data storage plan.

**5. Database Design**

Identifying the primary and foreign keys, the team utilised found in the metadata available on the website Victorian Government (please refer to the link in the Data Overview section) to guide the establishment of our Entity-Relationship (ER) schema. However, we refined some attributes that were not relevant to the project to ensure efficiency and relevance. The high level schema is firstly focused on the school and risk category.

6

**6. Data Analytics**

The data science team utilised the cleaned dataset to conduct an exploratory data analysis (EDA) to uncover critical insights related to bushfire risk and its impact on schools across Victoria. Our primary analysis focuses on understanding the potential bushfire risks that pose a threat to educational institutions and the number of schools and children affected by these risks. We aim to answer the following key questions:

● **Which** schools are located in the highest bushfire risk zones?

● **Who** is affected in these risk areas?

● **What** are the distributions of schools across various fire risk categories?

● **Where** does the bushfire risk vary in different local government areas and weather

districts?

By addressing these questions, we aim to provide valuable insights into the vulnerability of schools to bushfires. This information is essential for structuring the storytelling on the homepage of the website, allowing us to present data-driven narratives effectively. The following section highlights and reveals the insights extracted from the datasets, offering a comprehensive overview of the bushfire risks facing schools in Victoria.

7



| **Insight** |
| --- |
| The historical data shows that bushfire activity in Victoria has evolved significantly over time and has scattered across the state. In the early 1900s, bushfires were relatively occasional. Over the years, there has been a marked **increase in both the frequency and intensity of these fires**. Initially, fire activity was more concentrated around the Australian Capital Territory (ACT), but **recent data indicates a significant shift, with fires now becoming more concentrated in the central regions of Victoria**. This trend highlights the need for targeted fire management and increased preparedness in these newly affected areas. |

8



| The bar chart displays **a significant upward trend in the yearly number of bushfires in Victoria**, starting from the year 2000. Notably, there are dramatic peaks in 2003, 2009, 2014, and 2019, indicating **a pattern of increased sudden occurrences**, particularly in the past two decades. According to the Australian Bureau of Meteorology, **Australia is warming faster than the global average**, a factor that likely contributes to the growing severity of fire seasons. The spike in 2019 is especially alarming, with nearly 1,000 bushfires recorded, signifying the **growing severity of fire seasons in the 21st century**. This trend suggests an escalating risk of bushfires, emphasising the urgent need for enhanced fire management strategies, preparedness measures, and climate resilience efforts to mitigate the impact of these increasingly frequent and intense bushfire events. |
| --- |

9



| A **total of 839 schools have been identified as being in bushfire risk zones** in Victoria, highlighting the critical need for strategic intervention. The distribution of these schools across various risk levels, with Category 1 representing the highest risk, highlights the urgency of the situation. While most Government schools are situated in lower-risk categories, particularly CAT 3 and CAT 4, with over 270 and 300 schools respectively, the presence of schools in higher-risk zones cannot be overlooked. Catholic and Independent schools also predominantly fall into CAT 3 and CAT 4, though in smaller numbers. |
| --- |

10



| However, the few schools that are classified under the highest-risk category, **CAT 1, require immediate attention.** Specifically the **Central district appears to be a hotspot** for bushfires. This distribution indicates a **pressing need for targeted preparedness and risk mitigation strategies**, particularly for those schools in the highest-risk zones, to ensure the safety of students and staff. Proactive measures must be prioritised to address vulnerabilities and enhance bushfire resilience across all education sectors. |
| --- |

11



| The graph indicates a **stable enrollment of around 80,000 students** across various years, with a clear **drop in enrollments from year 10 to year 12, suggesting a potential issue with student retention**. A study by Towers in 2015 **identified gaps and misconceptions in bushfire education among children aged 8-12**. Furthermore, **nearly 27,000 of these**  **students are located in high-risk bushfire areas**, representing about 8% of the total. This overlap suggests that addressing educational gaps and improving bushfire preparedness in these high-risk zones could be crucial in enhancing student retention and safety. |
| --- |

12

**7. Ethical, Legal & Privacy Issues**

Some important aspects to consider when designing the ‘Bushfire Brigade’ web application are the ethical, legal and privacy issues associated with the team’s use of open data. We understand the importance of identifying and mitigating these issues and have incorporated this aspect into every step of the data management process for this project.

**Ethical Considerations**

Our team is determined to develop a project with ethical considerations at its core, with every effort taken to adhere to guidelines governing privacy and data protection:

● Transparency: We make every effort to be transparent about the way in which data is

collected, stored and used for the purposes of this project through extensive

documentation regarding the data sources used as well as the decisions made

regarding the data.

● Personally Identifiable Information (PII): We are committed to ensuring that no PII is

collected or stored, including information that may enable anonymous individuals

from the datasets to be identified.

**Legal Considerations**

The Data management approach for the project has been developed in line with all relevant laws and regulations:

● Licences: All the datasets used by the team are governed by the Creative Commons

Attribution 4.0 International Licence (Creative Commons, n.d.), and we are

committed to adhering to all the terms outlined in the Licence including giving

appropriate credit.

13



● Acts/Principles: We are determined to uphold each of the Australian Privacy

Principles (Office of the Australian Information Commissioner, n.d.) and protect

individuals’ privacy when working with open data.

**Privacy Considerations**

Our team is dedicated to protecting individuals’ privacy and managing data in a way that prioritises the protection of sensitive information:

● Necessary data collection: To avoid engaging in privacy violations, we only focus on

collecting data that is directly relevant to the project.

● Security: We have put in place various security measures including data access

controls to ensure that data is only accessible to authorised personnel, and encryption

to ensure that the data cannot be read by any unauthorised parties even if it is

accessed.

The ‘Bushfire Brigade’ project is determined to abide by all guidelines, laws and regulations governing ethics, legality and privacy. Through incorporating these principles into our approach to data management for this project, we hope to protect individuals’ privacy while simultaneously boosting the ‘Bushfire Brigade’ web application's legitimacy and dependability. Through upholding legal obligations, prioritising privacy, and preserving transparency, our goal is to construct a reliable and efficient solution that facilitates more child-friendly and impactful bushfire safety education in Victoria.

14

8/22/24, 11:22 PM Iteration1\_Datacleaning

**Load Packages**

In [42]:**import** time

**from** geopy.exc **import** GeocoderTimedOut

**from** geopy.geocoders **import** Nominatim

**import** plotly.express **as** px

**import** pandas **as** pd

**import** os

**!**pip install geopy

Requirement already satisfied: geopy in c:\users\thinithi\anaconda3\lib\site-packa ges (2.4.1)

Requirement already satisfied: geographiclib<3,>=1.52 in c:\users\thinithi\anacond a3\lib\site-packages (from geopy) (2.0)

In [6]:*# Get the current working directory*

current\_directory **=** os**.**getcwd()

current\_directory

Out[6]:

'C:\\Users\\Thinithi\\Monash\\Sem4\_2023\\FIT5120 INDUSTRY EXP\\onboarding\\ITERATI ON\_DATA'

**Data Exploration**

In [31]:file\_path **=** 'dv355-VIC All Schools Enrolments 2023.csv'

*# Read the CSV file into a DataFrame with a different encoding*

df **=** pd**.**read\_csv(file\_path, encoding**=**'ISO-8859-1')

*# describe data*

print(df**.**info())

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 1/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2290 entries, 0 to 2289

Data columns (total 26 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Education\_Sector 2290 non-null object

1 Entity\_Type 2290 non-null int64

2 School\_No 2290 non-null int64

3 School\_Name 2290 non-null object

4 School\_Type 2290 non-null object

5 School\_Status 2290 non-null object

6 "Prep Total" 2290 non-null float64

7 "Year 1 Total" 2290 non-null float64

8 "Year 2 Total" 2290 non-null float64

9 "Year 3 Total" 2290 non-null float64

10 "Year 4 Total" 2290 non-null float64

11 "Year 5 Total" 2290 non-null float64

12 "Year 6 Total" 2290 non-null float64

13 "Primary Ungraded Total" 2290 non-null float64

14 "Primary Total" 2290 non-null float64

15 "Year 7 Total" 2290 non-null float64

16 "Year 8 Total" 2290 non-null float64

17 "Year 9 Total" 2290 non-null float64

18 "Year 10 Total" 2290 non-null float64

19 "Year 11 Total" 2290 non-null float64

20 "Year 12 Total" 2290 non-null float64

21 "Secondary Ungraded Total" 2290 non-null float64

22 "Secondary Total" 2290 non-null float64

23 "Grand Total" 2290 non-null float64

24 Year 2290 non-null int64

25 CENSUS\_TYPE 2290 non-null object

dtypes: float64(18), int64(3), object(5)

memory usage: 465.3+ KB

None

In [18]:*# Check for nulls in columns*

df**.**isnull()**.**sum()

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 2/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

Out[18]:

Education\_Sector 0 Entity\_Type 0 School\_No 0 School\_Name 0 School\_Type 0 School\_Status 0 "Prep Total" 0 "Year 1 Total" 0 "Year 2 Total" 0 "Year 3 Total" 0 "Year 4 Total" 0 "Year 5 Total" 0 "Year 6 Total" 0 "Primary Ungraded Total" 0 "Primary Total" 0 "Year 7 Total" 0 "Year 8 Total" 0 "Year 9 Total" 0 "Year 10 Total" 0 "Year 11 Total" 0 "Year 12 Total" 0 "Secondary Ungraded Total" 0 "Secondary Total" 0 "Grand Total" 0 Year 0 CENSUS\_TYPE 0 dtype: int64

In [33]:*# describe data*

print(df**.**describe())

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 3/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

Entity\_Type School\_No "Prep Total" "Year 1 Total" "Year 2 Total" \ count 2290.000000 2290.000000 2290.000000 2290.000000 2290.000000 mean 1.316157 3531.783843 34.494236 35.045066 35.277773 std 0.465077 2518.851970 37.715966 38.869626 38.632585 min 1.000000 1.000000 0.000000 0.000000 0.000000 25% 1.000000 1554.000000 3.000000 3.000000 3.000000 50% 1.000000 2602.000000 24.000000 24.000000 25.000000 75% 2.000000 5239.750000 54.000000 54.000000 54.000000 max 2.000000 8917.000000 316.000000 355.000000 357.000000

"Year 3 Total" "Year 4 Total" "Year 5 Total" "Year 6 Total" \

count 2290.000000 2290.000000 2290.000000 2290.000000

mean 35.431354 35.308734 35.812533 34.705153

std 38.808744 38.645317 39.338791 38.477236

min 0.000000 0.000000 0.000000 0.000000

25% 4.000000 3.125000 3.000000 3.000000

50% 25.000000 25.000000 26.000000 25.000000

75% 54.000000 54.000000 54.000000 52.000000

max 383.000000 394.000000 425.000000 385.000000

"Primary Ungraded Total" ... "Year 7 Total" "Year 8 Total" \

count 2290.000000 ... 2290.000000 2290.000000

mean 2.793100 ... 34.554891 34.626463

std 18.821981 ... 77.200670 77.210002

min 0.000000 ... 0.000000 0.000000

25% 0.000000 ... 0.000000 0.000000

50% 0.000000 ... 0.000000 0.000000

75% 0.000000 ... 8.000000 8.000000

max 305.200000 ... 600.000000 561.000000

"Year 9 Total" "Year 10 Total" "Year 11 Total" "Year 12 Total" \

count 2290.000000 2290.000000 2290.000000 2290.000000

mean 34.373275 34.910480 33.142620 27.776638

std 76.201101 78.469754 79.575126 68.478003

min 0.000000 0.000000 0.000000 0.000000

25% 0.000000 0.000000 0.000000 0.000000

50% 0.000000 0.000000 0.000000 0.000000

75% 8.000000 5.000000 2.000000 0.000000

max 511.000000 600.000000 968.200000 776.100000

"Secondary Ungraded Total" "Secondary Total" "Grand Total" Year

count 2290.000000 2290.000000 2290.000000 2290.0

mean 2.481310 201.865677 450.733624 2023.0

std 16.295483 437.426883 481.084346 0.0

min 0.000000 0.000000 0.000000 2023.0

25% 0.000000 0.000000 132.650000 2023.0

50% 0.000000 0.000000 305.400000 2023.0

75% 0.000000 98.400000 584.250000 2023.0

max 253.000000 3317.000000 4610.000000 2023.0

[8 rows x 21 columns]

**Data Cleaning &Transformation**

**Student Enrollment Data**

In [35]:*# Clean column names by stripping extra characters*

df**.**columns **=** df**.**columns**.**str**.**strip()**.**str**.**replace('"', '')

*# Clean column names*

df**.**columns **=** df**.**columns**.**str**.**strip()**.**str**.**replace('"', '', regex**=False**)

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 4/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

In [36]:*# Convert the DataFrame from wide format to long format using the melt() function* long\_df **=** df**.**melt(

id\_vars**=**['Education\_Sector', 'Entity\_Type', 'School\_No', 'School\_Name',

'School\_Type', 'School\_Status', 'Year', 'CENSUS\_TYPE'],

value\_vars**=**['Prep Total', 'Year 1 Total', 'Year 2 Total', 'Year 3 Total', 'Year 'Year 6 Total', 'Primary Ungraded Total', 'Primary Total', 'Year 7 'Year 9 Total', 'Year 10 Total', 'Year 11 Total', 'Year 12 Total', 'Secondary Total', 'Grand Total'],

var\_name**=**'Year\_Level',

value\_name**=**'Enrollment'

)

*# Display the first few rows of the transformed DataFrame*

print(long\_df**.**head())

Education\_Sector Entity\_Type School\_No School\_Name \

0 Catholic 2 20 Parade College

1 Catholic 2 25 Simonds Catholic College

2 Catholic 2 26 St Marys College Melbourne

3 Catholic 2 28 St Patrick's College Ballarat

4 Catholic 2 29 St Patrick's School

School\_Type School\_Status Year CENSUS\_TYPE Year\_Level Enrollment

0 Secondary O 2023 F Prep Total 0.0

1 Secondary O 2023 F Prep Total 0.0

2 Secondary O 2023 F Prep Total 0.0

3 Secondary O 2023 F Prep Total 0.0

4 Primary O 2023 F Prep Total 28.0

In [38]:*# Verify the column names*

print(long\_df**.**info())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 41220 entries, 0 to 41219

Data columns (total 10 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Education\_Sector 41220 non-null object

1 Entity\_Type 41220 non-null int64

2 School\_No 41220 non-null int64

3 School\_Name 41220 non-null object

4 School\_Type 41220 non-null object

5 School\_Status 41220 non-null object

6 Year 41220 non-null int64

7 CENSUS\_TYPE 41220 non-null object

8 Year\_Level 41220 non-null object

9 Enrollment 41220 non-null float64

dtypes: float64(1), int64(3), object(6)

memory usage: 3.1+ MB

None

In [39]:*# Group by 'Year' and calculate the sum of 'Year 3 Total'*

yearly\_sum **=** long\_df**.**groupby('Year\_Level')['Enrollment']**.**sum()**.**reset\_index()

*# Filter to include only 'Year' levels*

yearly\_sum **=** yearly\_sum[yearly\_sum['Year\_Level']**.**str**.**contains('Year')]

*# Define the order of categories*

order **=** ['Year 1 Total', 'Year 2 Total', 'Year 3 Total', 'Year 4 Total', 'Year 5 To 'Year 7 Total', 'Year 8 Total', 'Year 9 Total', 'Year 10 Total',

'Year 11 Total', 'Year 12 Total']

*# Convert 'Year\_Level' to categorical with a specified order*

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 5/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

yearly\_sum['Year\_Level'] **=** pd**.**Categorical(

yearly\_sum['Year\_Level'], categories**=**order, ordered**=True**)

*# Sort the DataFrame by 'Year\_Level'*

yearly\_sum **=** yearly\_sum**.**sort\_values('Year\_Level')

*# Create the bar chart*

fig **=** px**.**bar(yearly\_sum, x**=**'Year\_Level', y**=**'Enrollment',

title**=**'Sum of Enrollments by student year level (In 2023)',

labels**=**{'Year\_Level': 'Year Level',

'Enrollment': 'Sum of Enrollment'},

text**=**'Enrollment')

*# Update the text formatting to include commas*

fig**.**update\_traces(texttemplate**=**'%{text:,.0f}', textposition**=**'outside')

*# Show the plot*

fig**.**show()

**School Bushfire risk Data**

In [48]:file\_path **=** 'Website-BARR-2023-24-updated.xlsx'

*# Read the Excel file into a DataFrame*

df2 **=** pd**.**read\_excel(file\_path)

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 6/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

In [49]:*# Display the first few rows of the DataFrame*

print(df2**.**info())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 868 entries, 0 to 867

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 row 868 non-null int64

1 SCHOOL\_NO2 868 non-null int64

2 flag 868 non-null int64

3 Fire Risk Category 2023-24 868 non-null object

4 Facility Name 868 non-null object

5 Education Sector 868 non-null object

6 Facility Address 868 non-null object

7 Town or Suburb 868 non-null object

8 Local Government Area 868 non-null object

9 Fire Weather District 867 non-null object

10 LATITUDE 868 non-null float64

11 LONGITUDE 868 non-null float64

dtypes: float64(2), int64(3), object(7)

memory usage: 81.5+ KB

None

**A) Geocoding the School address**

In [50]:*# Initialize geocoder*

geolocator **=** Nominatim(user\_agent**=**"myGeocoder")

*# Function to geocode addresses*

**def** geocode\_address(address):

**try**:

*# Attempt to get the geographic coordinates (latitude and longitude) of the # Timeout is set to handle cases where the service is slow*

location **=** geolocator**.**geocode(address, timeout**=**10)

**if** location:

*# If the location is found, return the latitude and longitude*

**return** location**.**latitude, location**.**longitude

**else**:

*# If no location is found, return (None, None)*

**return None**, **None**

**except** GeocoderTimedOut:

*# If the geocoding service times out, retry the geocoding request*

**return** geocode\_address(address) *# Recursive call to retry*

**except** Exception **as** e:

*# If any other exception occurs, print the error and return (None, None)* print(f"Error: {e}")

**return None**, **None**

In [51]:*# Concatenate 'Facility Address' with 'Town or Suburb'*

df2['Full Address'] **=** df2['Facility Address']**.**str**.**strip(

) **+** ', ' **+** df2['Town or Suburb'] **+** ', ' **+** df2['Local Government Area']**.**str**.**strip()

*# Display the DataFrame to check the Full Address*

df2**.**head(5)

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 7/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

Out[51]:

**row SCHOOL\_NO2 flag**

**Fire Risk Category 2023-24**

**Facility Name**

Advance

**Education Sector**

**Facility Address**

1973

**Town or Suburb**

**Loca**

**Governmen Are**

**0** 1 1098 2 CAT 3 **1** 2 5566 1 CAT 2

College of Education Incorporated - Ha...

Aireys Inlet Primary

School

Independent Government

Frankston Flinders Road

13

Anderson Street

Hastings Morningto Peninsul

Aireys

Inlet Surf Coas

**2** 3 2101 2 CAT 2 Alice Miller School Independent

110

Bailey Road

Macedon Macedo Range

**3** 4 366 1 CAT 3

Alice Miller School -

Candlebark

Independent 83 Kerrie

Road RomseyMacedo

Range

**4** 5 1906 1 CAT 3

Al-Taqwa College - Camp

Independent

10

Cranswick Road

Banksia Peninsula

Eas

Gippsland

In [ ]:*# Create a DataFrame to store the geocoded results*

results **=** pd**.**DataFrame(df2['Full Address'], columns**=**['Full Address'])

*# Apply geocoding with a delay to handle rate limits*

**def** apply\_geocoding(address):

time**.**sleep(1) *# Adding delay to handle rate limits*

**return** geocode\_address(address)

*# Apply geocoding to addresses and create Latitude and Longitude columns*

results[['Latitude', 'Longitude']] **=** results['Full Address']**.**apply(

**lambda** x: pd**.**Series(apply\_geocoding(x)))

*# Merge the geocoded results with the original DataFrame*

final\_df **=** pd**.**concat([df2, results[['Latitude', 'Longitude']]], axis**=**1)

*# Display the DataFrame with geocoded coordinates*

final\_df**.**head()

In [58]: *# Write the DataFrame to a CSV file*

final\_df**.**to\_csv('geocoded\_facilities.csv')

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 8/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

**School register and location data**

In [53]:file\_path **=** 'dv346-schoollocations2023.csv'

*# Read the CSV file into a DataFrame with a different encoding*

df3 **=** pd**.**read\_csv(file\_path, encoding**=**'ISO-8859-1')

*# Display the structure*

df3**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2302 entries, 0 to 2301

Data columns (total 25 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Education\_Sector 2302 non-null object

1 Entity\_Type 2302 non-null int64

2 School\_No 2302 non-null int64

3 count 2302 non-null int64

4 School\_Name 2302 non-null object

5 School\_Type 2302 non-null object

6 School\_Status 2302 non-null object

7 Address\_Line\_1 2302 non-null object

8 Address\_Line\_2 11 non-null object

9 Address\_Town 2302 non-null object

10 Address\_State 2302 non-null object

11 Address\_Postcode 2302 non-null int64

12 Postal\_Address\_Line\_1 2302 non-null object

13 Postal\_Address\_Line\_2 15 non-null object

14 Postal\_Town 2302 non-null object

15 Postal\_State 2302 non-null object

16 Postal\_Postcode 2302 non-null int64

17 Full\_Phone\_No 2302 non-null object

18 LGA\_ID 2302 non-null int64

19 LGA\_Name 2302 non-null object

20 X 2301 non-null float64

21 Y 2301 non-null float64

22 X.1 2302 non-null float64

23 Y.1 2302 non-null float64

24 lat-lon 2302 non-null object

dtypes: float64(4), int64(6), object(15)

memory usage: 449.7+ KB

In [54]:*# Display the first few rows of the DataFrame*

df3**.**head()

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.ip… 9/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

Out[54]:

**Education\_Sector Entity\_Type School\_No count School\_Name School\_Type School\_Status Ad** Alberton

**0** Government 1 1 2 **1** Government 1 3 1

Primary

School

Allansford and District Primary

School

Primary O Primary O

**2** Government 1 4 1Avoca Primary

School Primary O

Avenel

**3** Government 1 8 1

**4** Government 1 12 1 5 rows × 25 columns

In [55]:*# Load geocoded data*

file\_path **=** 'geocoded\_facilities.csv'

Primary

School

Warrandyte Primary

School

Primary O Primary O

*# Read the CSV file into a DataFrame with a different encoding*

df4 **=** pd**.**read\_csv(file\_path, encoding**=**'ISO-8859-1')

*# Display the first few rows of the DataFrame*

print(df4**.**info())

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 868 entries, 0 to 867

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Unnamed: 0 868 non-null int64

1 row 868 non-null int64

2 Fire Risk Category 2023-24 868 non-null object

3 Facility Name 868 non-null object

4 Education Sector 868 non-null object

5 Facility Address 868 non-null object

6 Town or Suburb 868 non-null object

7 Local Government Area 868 non-null object

8 Fire Weather District 867 non-null object

9 Full Address 868 non-null object

10 Latitude 818 non-null float64

11 Longitude 818 non-null float64

dtypes: float64(2), int64(2), object(8)

memory usage: 81.5+ KB

None

**B) Matching schools using longitude and latitude**

In [60]:*# Filter out rows with NaN in the Coordinates columns*

df4 **=** df4**.**dropna(subset**=**['Latitude', 'Longitude'])

df3 **=** df3**.**dropna(subset**=**['Y', 'X'])

*# Extract latitudes and longitudes*

df4['Coordinates'] **=** list(zip(df4['Latitude'], df4['Longitude']))

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.i… 10/14

8/22/24, 11:22 PM Iteration1\_Datacleaning df3['Coordinates'] **=** list(zip(df3['Y'], df3['X']))

df4**.**head()

Out[60]:

**Unnamed:**

**0row**

**Fire Risk Category 2023-24**

**Facility Name**

Advance

**Education Sector**

**Facility Address**

1973

**Town or Suburb**

**Local**

**Government Area**

**F**

**Weath Distr**

**0** 0 3 CAT 3 **1** 1 4 CAT 2

College of Education Incorporated - Ha...

Aireys Inlet Primary

School

Independent Government

Frankston Flinders Road

13

Anderson Street

Hastings Mornington

Peninsula Cen

Aireys

Inlet Surf Coast Cen

**2** 2 5 CAT 2 Alice Miller School Independent

110

Bailey Road

Macedon Macedon

Ranges Cen

**3** 3 6 CAT 3

Alice Miller School -

Candlebark

Independent 83 Kerrie

Road RomseyMacedon

Ranges Cen

**4** 4 9 CAT 3

Al-Taqwa College - Camp

Independent

10

Cranswick Road

Banksia Peninsula

East

Gippsland

E

Gippsla

Reference: https://geopy.readthedocs.io/en/stable/#geopy.distance.GeodesicDistance

In [61]:**from** geopy.distance **import** geodesic

*# Filter out rows with Null in the Coordinates columns*

df3 **=** df3**.**dropna(subset**=**['Y', 'X'])

df4 **=** df4**.**dropna(subset**=**['Latitude', 'Longitude'])

*# Extract latitudes and longitudes*

df4['Coordinates'] **=** list(zip(df4['Latitude'], df4['Longitude']))

df3['Coordinates'] **=** list(zip(df3['Y'], df3['X']))

*# Ensure no extra spaces in column names*

df3**.**columns **=** df3**.**columns**.**str**.**strip()

df4**.**columns **=** df4**.**columns**.**str**.**strip()

*# Function to find the closest school in dataset A for a given facility in dataset* **def** find\_closest\_school(facility\_coords, school\_coords):

closest\_school **= None**

min\_distance **=** float('inf')

**for** i, school\_coord **in** enumerate(school\_coords):

distance **=** geodesic(facility\_coords, school\_coord)**.**kilometers

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.i… 11/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

**if** distance **<** min\_distance:

min\_distance **=** distance

closest\_school **=** i

**return** closest\_school

*# Ensure 'School\_No' exists in df3*

**if** 'School\_No' **not in** df3**.**columns:

**raise** KeyError("The column 'School\_No' is not present in df3")

*# Find the closest school for each facility*

df4['Closest\_School\_No'] **=** df4['Coordinates']**.**apply(

**lambda** x: df3**.**iloc[find\_closest\_school(x, df3['Coordinates'])]['School\_No'] **if** find\_closest\_school(x, df3['Coordinates']) **is not None**

**else None**

)

*# Define the path and filename for the CSV file*

csv\_file\_path **=** 'output\_data\_with\_school\_no.csv'

*# Save the DataFrame to a CSV file*

df4[['row','Facility Name', 'Closest\_School\_No']]**.**to\_csv(csv\_file\_path, index**=False**

In [64]:*# reload dataset with all long and lat, remap schools*

file\_path **=** 'Website-BARR-2023-24-updated.xlsx'

*# Read the Excel file into a DataFrame*

df5 **=** pd**.**read\_excel(file\_path)

In [65]:*# Extract latitudes and longitudes*

df5['Coordinates'] **=** list(zip(df5['LATITUDE'], df5['LONGITUDE']))

*# Ensure no extra spaces in column names*

df5**.**columns **=** df5**.**columns**.**str**.**strip()

*# Find the closest school for each facility*

df5['Closest\_School\_No'] **=** df5['Coordinates']**.**apply(

**lambda** x: df3**.**iloc[find\_closest\_school(x, df3['Coordinates'])]['School\_No'] **if** find\_closest\_school(x, df3['Coordinates']) **is not None**

**else None**

)

*# Define the path and filename for the CSV file*

csv\_file\_path **=** 'output\_data\_with\_school\_no2.csv'

*# Save the DataFrame to a CSV file*

df5[['row','Facility Name', 'Closest\_School\_No']]**.**to\_csv(csv\_file\_path, index**=False**

**C) Matching schools between the datasets based on name and city**

In [69]:**!**pip install fuzzywuzzy

**!**pip install python-Levenshtein

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.i… 12/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

Collecting fuzzywuzzy

Obtaining dependency information for fuzzywuzzy from https://files.pythonhosted. org/packages/43/ff/74f23998ad2f93b945c0309f825be92e04e0348e062026998b5eefef4c33/fu zzywuzzy-0.18.0-py2.py3-none-any.whl.metadata

Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl.metadata (4.9 kB)

Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)

Installing collected packages: fuzzywuzzy

Successfully installed fuzzywuzzy-0.18.0

Collecting python-Levenshtein

Obtaining dependency information for python-Levenshtein from https://files.pytho nhosted.org/packages/72/8e/559c539e76bc0b1defec3da39a047fe151258efc9b215bf41db41e2 c7922/python\_Levenshtein-0.25.1-py3-none-any.whl.metadata

Downloading python\_Levenshtein-0.25.1-py3-none-any.whl.metadata (3.7 kB)

Collecting Levenshtein==0.25.1 (from python-Levenshtein)

Obtaining dependency information for Levenshtein==0.25.1 from https://files.pyth onhosted.org/packages/47/19/4528246e25bb79fa8d4adae6640251c613f05eb310d79307d1ac53 c7bf28/Levenshtein-0.25.1-cp311-cp311-win\_amd64.whl.metadata

Downloading Levenshtein-0.25.1-cp311-cp311-win\_amd64.whl.metadata (3.4 kB)

Collecting rapidfuzz<4.0.0,>=3.8.0 (from Levenshtein==0.25.1->python-Levenshtein) Obtaining dependency information for rapidfuzz<4.0.0,>=3.8.0 from https://files. pythonhosted.org/packages/aa/bb/cdd512d40f8ea67692deee6b0da4f7235c6a0f9e126fdded32 b62c5d91fe/rapidfuzz-3.9.6-cp311-cp311-win\_amd64.whl.metadata

Downloading rapidfuzz-3.9.6-cp311-cp311-win\_amd64.whl.metadata (12 kB)

Downloading python\_Levenshtein-0.25.1-py3-none-any.whl (9.4 kB)

Downloading Levenshtein-0.25.1-cp311-cp311-win\_amd64.whl (98 kB)

---------------------------------------- 0.0/98.4 kB ? eta -:--:--

------------------------------------- -- 92.2/98.4 kB 2.6 MB/s eta 0:00:01 ---------------------------------------- 98.4/98.4 kB 1.9 MB/s eta 0:00:00 Downloading rapidfuzz-3.9.6-cp311-cp311-win\_amd64.whl (1.7 MB)

---------------------------------------- 0.0/1.7 MB ? eta -:--:--

--------- ------------------------------ 0.4/1.7 MB 11.9 MB/s eta 0:00:01

----------------- ---------------------- 0.7/1.7 MB 9.3 MB/s eta 0:00:01

------------------------------ --------- 1.2/1.7 MB 9.9 MB/s eta 0:00:01

---------------------------------------- 1.7/1.7 MB 10.5 MB/s eta 0:00:00

Installing collected packages: rapidfuzz, Levenshtein, python-Levenshtein

Successfully installed Levenshtein-0.25.1 python-Levenshtein-0.25.1 rapidfuzz-3.9. 6

Reference: https://stackoverflow.com/questions/32055817/python-fuzzy-matching

fuzzywuzzy-keep-only-best-match

In [80]:**from** fuzzywuzzy **import** fuzz

**from** fuzzywuzzy **import** process

**import** pandas **as** pd

*# Ensure no extra spaces in column names*

df2**.**columns **=** df2**.**columns**.**str**.**strip()

df3**.**columns **=** df3**.**columns**.**str**.**strip()

*# Combine 'Facility Name' and 'Town or Suburb' into a single string*

df2['Name\_City'] **=** df2['Facility Name'] **+** ", " **+** df2['Town or Suburb']

*# Combine 'School\_Name' and 'Address\_Town' into a single string*

df3['Name\_City'] **=** df3['School\_Name'] **+** ", " **+** df3['Address\_Town']

*# Function to find the best match for a facility in df2 with the school in df3*

**def** find\_best\_school\_match(facility\_name\_city, school\_name\_cities, threshold**=**80): best\_match **=** process**.**extractOne(

facility\_name\_city, school\_name\_cities, scorer**=**fuzz**.**ratio)

**if** best\_match **and** best\_match[1] **>=** threshold:

*# return the best matching school name if the match is above the threshold* localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.i… 13/14

8/22/24, 11:22 PM Iteration1\_Datacleaning

**return** best\_match[0]

**else**:

**return None**

*# Ensure 'School\_No' and 'Name\_City' exist in df3*

**if** 'School\_No' **not in** df3**.**columns **or** 'Name\_City' **not in** df3**.**columns:

**raise** KeyError(

"The column 'School\_No' or 'Name\_City' is not present in df3")

*# Create a dictionary to map Name\_City to their corresponding School\_No*

name\_city\_to\_no **=** df3**.**set\_index('Name\_City')['School\_No']**.**to\_dict()

*# Find the best matching school name for each facility and get its School\_No*

df2['Closest\_School\_Name\_City'] **=** df2['Name\_City']**.**apply(

**lambda** x: find\_best\_school\_match(x, df3['Name\_City'], threshold**=**80)

)

df2['Closest\_School\_No'] **=** df2['Closest\_School\_Name\_City']**.**map(name\_city\_to\_no)

*# Define the path and filename for the CSV file*

csv\_file\_path **=** 'output\_data\_with\_school\_no3.csv'

*# Save the DataFrame to a CSV file*

df2[['row', 'Facility Name', 'Town or Suburb', 'Closest\_School\_No']]**.**to\_csv(

csv\_file\_path, index**=False**)

**Longitude and latitude mappings were further validated through name-address**

**similarity mapping to identify whether the same school appears in both datasets. A unique identifier was assigned to each school to facilitate this process. This approach ensured that schools with matching geographic coordinates were accurately linked, while discrepancies were addressed by verifying names and addresses to confirm or correct the data..**

In [ ]:

localhost:8888/nbconvert/html/Thinithi/Monash/Sem4\_2023/FIT5120 INDUSTRY EXP/onboarding/ITERATION\_DATA/Iteration1\_Datacleaning.i… 14/14