

**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

SC4020 Data Analytics & Mining Group Project 2

Name	Matriculation Number
Brandon Jang Jin Tian	U2220936G
Chung Zhi Xuan	U2220300H
Ting Ruo Chee	U2220572C
Yau Jun Hao	U2220332F

Abstract:

This report presents our experiments and findings from three data mining tasks conducted on human mobility data in Japan. Our objectives were to uncover patterns and insights using different algorithmic approaches:

- 1) **Frequent POI Identification:** *Implementing the Apriori algorithm to identify popular Points-of-Interest (POIs) within the dataset.*
- 2) **Mobility Pattern Mining:** *Applying the Generalised Sequential Pattern (GSP) algorithm to extract common movement sequences among residents.*
- 3) **Advanced Prediction Task:** *Using a Long Short-Term Memory (LSTM) model to predict a resident's next location based on their past mobility history.*

For each task, we provide a comprehensive analysis of our approach and detailed problem-solving methods, integrating these strategies into our code to optimise accuracy and efficiency. Through this project, we gained valuable insights into the effectiveness of sequential data mining methods and the potential of predictive modelling in forecasting movement behaviours.

Task 1: Analysis of Co-occurrence Patterns of Points of Interest (POI)

1.1: Apriori Algorithm

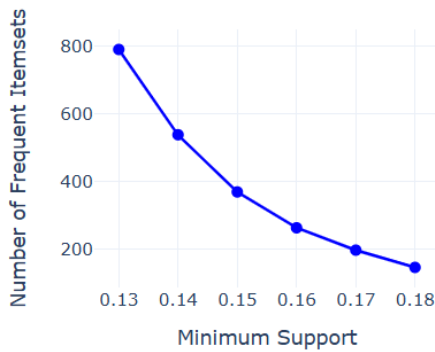


Figure 1: No. of frequent itemsets against minimum support threshold

density calculation. The average support is then used as a reference to determine the minimum support threshold. Based on Figure 1, a minimum support threshold of 0.15 appears to be appropriate, as it provides a moderate number of itemsets without capturing excessive noise.

The Apriori algorithm leverages the principle that if an itemset is frequent, then all of its subsets must also be frequent. Conversely, if an itemset is not frequent, any superset of that itemset cannot be frequent either. This property helps to eliminate branches of the search tree and reduces computational resources.

1.2: Determining minimum support threshold

Determining the minimum support threshold for Apriori is an iterative process that relies on domain knowledge and statistical analysis.

A method proposed by Hikmawati.E, et al (2021), utilises an utility function to rank each item. However, due to a lack of domain knowledge, we assigned equal importance to all POIs, setting their importance to 1. As a result, the function essentially converts to a

1.3: Analysis of Co-occurrence Patterns of Points of Interest (POI)

x	y	category
1	1	[74, 48, 79, 69, 73, 59, 58]
1	2	[61, 80, 48, 76, 60]
1	3	[74, 81, 48, 82, 54, 56, 58, 73, 36, 60]
1	4	[73, 81, 63, 48, 79, 74, 69]
1	5	[74, 60, 48, 69, 62, 76, 79]

Figure 2: Transactions Table

We created baskets for each combination of x and y. Based on the determined support threshold, we applied the Apriori algorithm to generate frequent itemsets.

The algorithm identifies frequent itemsets by first calculating the support of individual items. Then, it iteratively combines these frequent items into larger itemsets, while pruning any itemsets that do not meet the min_support threshold, until no more frequent itemsets can be identified.

To analyse co-occurrence patterns, our primary focus is on frequent itemsets that contain more than one item. Since we are not generating any association rules, and because the support of a subset is always higher than that of its superset, we removed all frequent itemsets that are subsets of larger frequent itemsets to obtain more meaningful patterns.

1.3.1: Itemsets with largest number of items

All four cities exhibit similar patterns, with heavy industry, building materials, hair salons, and transit stations being the most common POIs. Some counterintuitive insights include the frequent co-occurrence of hair salons with heavy industry and building materials. This could be due to the high number of these POIs, which increases the chances of matching. On the other hand, real estate does not co-occur with other popular POIs in City D. City C, being a denser city, shows larger itemsets compared to the other cities.

City A	City B
Hair Salon, Building Material Heavy Industry, Real Estate, Home Appliances ----- Heavy Industry, Real Estate Hair Salon, Laundry ----- Real Estate, Hair Salon Laundry , Building Material ----- Heavy Industry, Real Estate Laundry , Building Material -----	Heavy Industry, Home Appliances Building Material, Transit Station ----- Home Appliances, Hair Salon Building Material, Transit Station ----- Real Estate, Home Appliances Building Material, Transit Station ----- Heavy Industry, Hair Salon, Transit Station ----- Accountant Office, Home Appliances, Building Material -----
City C	City D

Hair Salon, Building Material, Park Transit Station, Real Estate, Home Appliances ----- Hair Salon, Accountant Office, Building Material Park, Transit Station, Real Estate ----- Accountant Office, Building Material, Park Transit Station, Real Estate, Home Appliances ----- Park, Building Material Transit Station, Elderly Care Home, Real Estate -----	Heavy Industry, Hair Salon Building Material, Transit Station ----- Heavy Industry, Hair Salon Laundry , Transit Station ----- Hair Salon, Laundry Building Material, Transit Station ----- Heavy Industry, Hair Salon Laundry , Building Material -----
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1.3.2: Itemsets with highest support (After Filtering)

After removing subsets of larger frequent itemsets, we obtained unique patterns. We identified some key differences between the cities. City A and City C both have higher support, with City A being larger in size and City C being denser. POIs in City A are still strongly correlated with heavy industry, while City B shows more frequent combinations involving churches.

City A	City B
Community Center, Heavy Industry : 0.1914 ----- Heavy Industry, Church, Home Appliances : 0.1862 ----- Heavy Industry, Elderly Care Home : 0.1858 ----- Heavy Industry, Church, Building Material : 0.1844 ----- Heavy Industry, Port : 0.1822 -----	Port : 0.128 ----- Heavy Industry, Church : 0.1215 ----- Church, Home Appliances : 0.1179 ----- Community Center, Church : 0.1178 ----- NPO : 0.114 -----
City C	City D
Park, Building Material Transit Station, Elderly Care Home, Real Estate : 0.1944 ----- Café : 0.1938 ----- Heavy Industry, Driving School, Transit Station : 0.1938 ----- Retail Store, Park, Transit Station : 0.1926 ----- Heavy Industry, Driving School, Building Material : 0.1916 -----	Grocery Store, Church : 0.099 ----- Post Office : 0.0957 ----- Heavy Industry, Retail Store : 0.0924 ----- Church, Driving School : 0.0918 ----- Japanese restaurant, Transit Station : 0.0911 -----

1.3.3: POIs with lowest degree

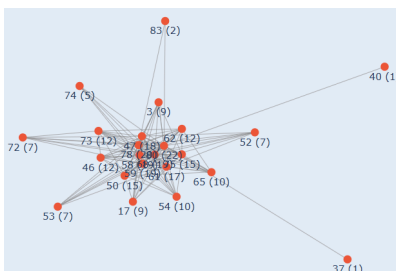


Figure 3: Network Graph City A

We used a network graph to identify POIs with the lowest degree. It's important to note that a low degree does not necessarily imply low support.

These POIs often co-occur with more popular ones, such as heavy industry, building materials, and transit stations. Combinations like (port, heavy industry), (kindergarten, park), and (post office, transit station) align with our intuition.

City A	City B
Heavy Industry, Port : 0.1822 ----- Heavy Industry, Interior Shop : 0.1563 ----- Grocery Store, Heavy Industry : 0.1532 ----- Port, Building Material : 0.1525 -----	Heavy Industry, Retail Store : 0.0843 ----- Interior Shop, Transit Station : 0.0826 ----- Retail Store, Transit Station : 0.0819 ----- Post Office, Transit Station : 0.0804 -----
City C	City D

Kindergarten, Transit Station : 0.1913 ----- Post Office, Transit Station : 0.1861 ----- Church, Transit Station : 0.1799 ----- Kindergarten, Real Estate : 0.179 ----- Kindergarten, Park : 0.1787 ----- Cram School, Transit Station : 0.1704 -----	Heavy Industry, Port : 0.0765 ----- Kindergarten, Building Material : 0.0703 -----
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------

1.3.4: 1-Itemsets

Some POIs, such as cafes, grocery stores, and convenience stores, occur frequently, but the co-occurrence patterns between them and other POIs appear to be random. As a result, no meaningful patterns are captured.

City A	City B	City C	City D
Post Office Convenience Store Cram School Kindergarten NPO Drug Store	Port NPO Kindergarten Fishing Chiropractic Convenience Store Cafe	Cafe Grocery Store Sweets Lawyer Office NPO	Post Office Convenience Store Gardening Clothes Store School Sweets Cafe Drug Store

Task 2: Mining Sequential Patterns

2.1: Implementing the Generalised Sequential Pattern (GSP) Algorithm

The GSP algorithm is a method used to mine sequential patterns. Given a database of sequences and a minimum support threshold (`min_support`), the goal is to aggregate all frequent subsequences, or patterns, that appear at least `min_support` times. In this project, we were tasked with mining frequent patterns in the mobility data as sequences of coordinate pairs. We defined a GSP class and compiled the required attributes and functions.

```
class GSP:
    def __init__(self, sequences, min_support):
        self.sequences = sequences          # List of sequences with coordinate pairs
        self.min_support = min_support      # Minimum support threshold
        self.frequent_patterns = []        # Store all found frequent patterns
```

Figure 4: Initialisation of the GSP Class

The functions defined in the GSP class are summarised below:

- `is_subsequence(self, candidate, sequence)`: Returns True if candidate is a subsequence of sequence.
- `count_support(self, candidates)`: Counts the support of each sequence in candidates by iterating through the `self.sequences` attribute in the GSP class. Candidates which do not meet `self.min_support` are removed.
- `generate_candidates(self, prev_frequent_patterns, k)`: Generates candidates of `k` length by merging `prev_frequent_patterns` with common prefixes and suffixes. Candidates with infrequent subsequences are pruned.
- `run(self)`: Runs the GSP algorithm.

2.2: Data Preprocessing

As per the project description, we limit the frequency mining to only the first month's data (30 days) to manage computational demands effectively. Here, we define an arbitrary start date of the data as '2024-01-01'. We then preprocess the data by combining the date and time into a single datetime column to pass through the trackintel functions.

Due to the size of the data, we will chunk the code into size of 100000 to perform the GSP algorithm. For each chunk we pass the data through the `read_positionfixes_gpd` function provided by trackintel to generate position fixes.

2.2.1: Staypoints Generation

Staypoints represent points where a user remains stationary for a certain period. We generated staypoints from positionfixes using the generate_staypoints function from the trackintel library with the following parameters:

Parameters	Description	Value
gap_threshold	Time (in minutes) required to start a new staypoint	45
dist_threshold	Distance threshold (in units of 500 metres) for defining a staypoint	10

2.2.2: Triplegs Generation

Triplegs represent the movement between staypoints and were generated by combining positionfixes between consecutive staypoints. The function generate_triplegs from the trackintel library were used with the following parameters:

Parameters	Description	Value
gap_threshold	Maximum time gap (in minutes) allowed between consecutive positionfixes within a tripleg. If exceeded, a new tripleg begins.	45
method	Method used to associate triplegs with staypoints, here set to 'overlap_staypoints' to capture overlapping segments.	overlap_staypoints

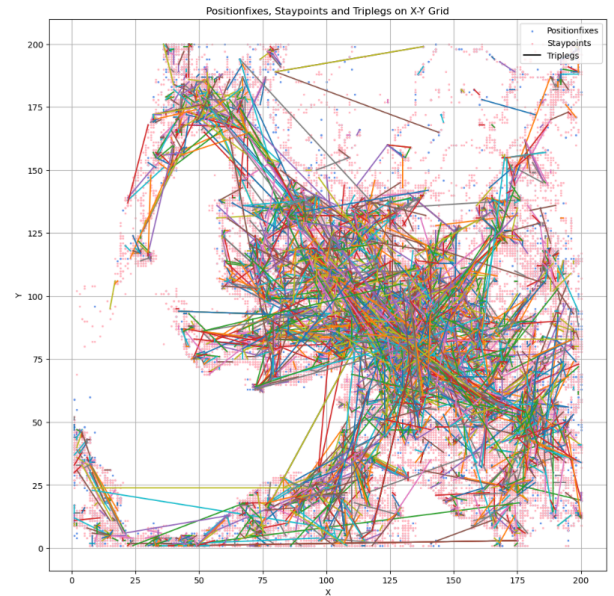


Figure 5: Illustration of Positionfixes, Staypoints and Triplegs for City A

2.3: Challenges

The challenges and limitations we faced while completing this task and how we addressed them are explained below.

- Tripleg preprocessing:** Many of the triplegs contained duplicate coordinates which appeared consecutively, leading to longer triplegs and redundant calculations while running the GSP algorithm. Hence, we created utility functions to remove consecutive duplicates and split up triplegs with more than 10 elements.
- Optimised support counting:** Support counting is the most time-consuming step as it involves passing through the dataset for every candidate sequence to find whether a subsequence exists. To streamline the process, we modified our algorithm to skip the counting step for a candidate sequence if it is longer than the sequence it is being compared to or if it has reached min_support counts, as it is already frequent.
- Chunk processing:** Performing the GSP algorithm on individual chunks significantly lowered the computational demands required for this task. However, one limitation is that we are unable to discover sequences that are infrequent in separate chunks but are frequent in the entire dataset. Nevertheless, it still allows us to aggregate some common patterns across different residents for each city.

2.4: Experimental Results

We ran the GSP algorithm with different numbers of min_support for each city. This is because relatively infrequent patterns can still appear a substantial number of times in cities with more mobility data. Therefore, it is necessary to adjust min_support in proportion to the dataset size to extract sufficient frequent patterns whilst minimising noise.

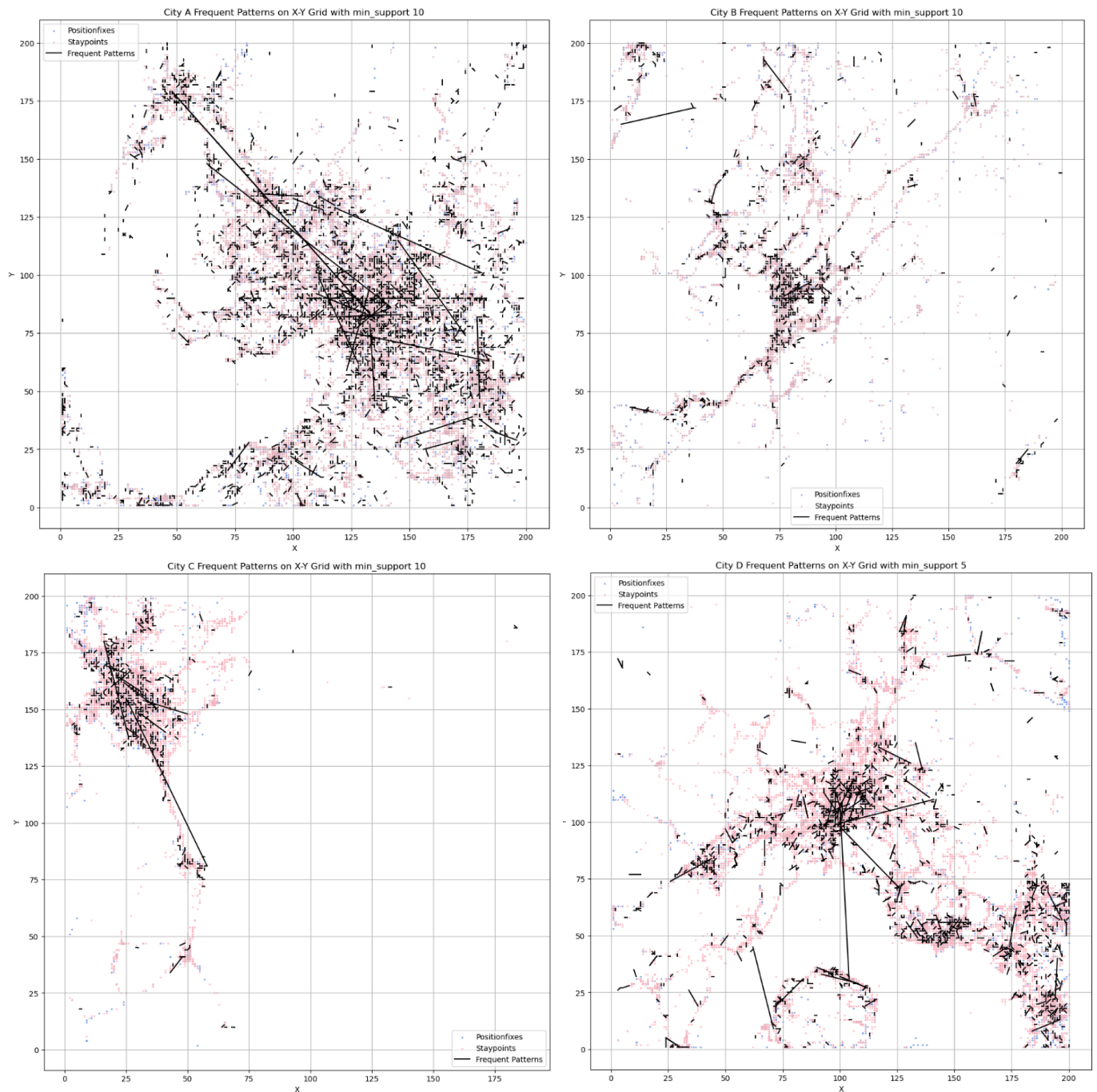


Figure 6: Frequent Patterns Extracted from Cities A, B, C and D

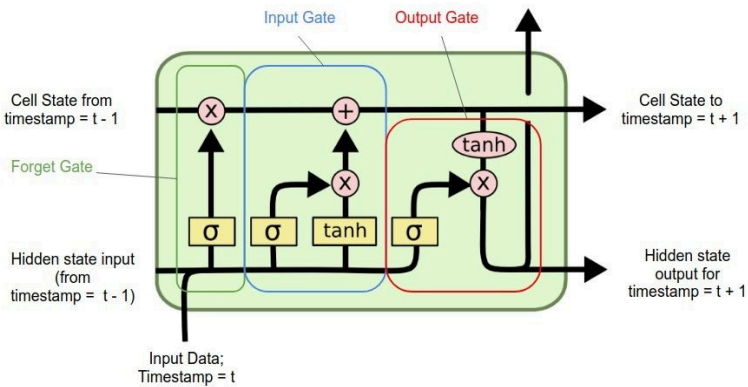
City A (min_support = 10)	This city has a dense central cluster with frequent patterns extending outward in various directions. There are distinct, lengthy movement lines, suggesting long-distance routes that connect further parts of the city to the central hub. Regular commuting within outer regions is also indicated by dispersed patterns, pointing to a structured urban area surrounded by downtown areas.
City B (min_support = 10)	City B shows a more fragmented pattern with smaller clusters and shorter frequent paths compared to City A. The lack of a clear centralised path or cluster, with its spread of staypoints across the city, suggests a decentralised activity pattern. This might indicate a city where residents are not concentrated in one area but instead distribute their activities across several locations.
City C (min_support = 10)	The frequent patterns tend to be concentrated in a narrow, elongated region along the vertical axis from the upper to lower sections of the city. This suggests a major route or corridor of activity, possibly indicating a primary commuting or transport path. We can also see that the staypoints are clustered around this central path, implying areas where people frequently stop or spend time along this main route.

City D (min_support = 5)	We used a lower min_support for City D since we have less data as compared to other cities. From the plots, we see that there are more distinct clusters in different areas, which might indicate multiple centres of activity rather than a single dominant path. The frequent patterns show connectivity among several clusters, suggesting that people are moving between different points across the city more evenly compared to City C.
------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Task 3: Open Advanced Tasks (Prediction of Subsequent Location)

3.1: Long Short-Term Memory (LSTM) Model Training

LSTM network is a type of Recurrent Neural Network (RNN) that is effective for modelling sequences over a period of time. Unlike traditional RNNs, LSTM can capture long-term dependencies in sequential data due to its unique architecture, which includes mechanisms to manage and retain information across timesteps. Therefore, we decided to deploy LSTMs given that this task requires past observations to impact future predictions, such as predicting a particular user's next location in Hiroshima based on historical movement patterns.

Memory Cell	The core of the LSTM is its memory cell, which allows it to remember values over long sequences.
Gates	<p>Utilises 3 Gates to control information flow</p>  <p><i>Figure 7: LSTM Architecture</i> (Abdullah Al Rahman, 2023)</p> <p>Forget Gate: Decides what information from previous timesteps should be forgotten. Input Gate: Determines what new information should be added to the memory cell. Output Gate: Controls the information to output based on the cell state.</p>
Cell State	The cell state flows across all timesteps, allowing LSTMs to retain long-term dependencies more effectively.

LSTM Model Architecture	
Input Layer	<ul style="list-style-type: none"> x and y coordinates (location data) d (relative date) t (time within a day) category (type of nearby establishment or POI)
LSTM Layers	<ul style="list-style-type: none"> First LSTM Layer with 64 units, configured to return sequences (return_sequences=True) to pass information to the next LSTM layer. Second LSTM Layer with 32 units, which outputs the processed information for predicting the next location.
Dense Output Layer	A fully connected layer with 2 units to predict the x and y coordinates of the next location.

```

1 # Train the model
2 history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

1 # Evaluate the model
2 loss, mae = model.evaluate(X_test, y_test)
3 print(f"Test Loss: {loss}, Test MAE: {mae}")

663/663 [=====] - 2s 3ms/step - loss: 9.8024 - mae: 0.8231
Test Loss: 9.802356719970703, Test MAE: 0.8231422305107117

```

Figure 8: Code Retrieved from Jupyter Notebook

By running 10 epochs with a batch size of 32, the Mean Absolute Error (MAE) is low and shows signs of overfitting.

3.2: Model Refinement & Parameter Tuning

```
# Early Stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

# Train the model
history = model.fit(
    X_train, y_train, epochs=20, batch_size=32, validation_split=0.2,
    callbacks=[early_stopping]
)
```

Figure 9: Incorporating Early Stopping

```
#Experiment with different sequence lengths, e.g., 3, 5, 7
sequence_length = 5

# Modified Model with adjusted sequence length, adding more units for tuning
model = Sequential()
model.add(LSTM(128, input_shape=(sequence_length, X_sequences.shape[2]), return_sequences=True))
model.add(LSTM(64))
model.add(Dense(2))
```

Figure 10: Parameter Tuning

to unseen data. In addition, we opted for **sequenced length (5)** as **shorter sequences (3)** are faster to train and less prone to overfitting, but may miss critical long-term patterns. **Longer sequences(7)** will improve performance in capturing complex movement patterns but risk overfitting and require more memory. We also increase the units to two LSTM layers with 128 and 64 units, balancing complexity and generalisability. We considered **lower units (32)** as having faster training but less capacity to capture complex movement dynamics while **higher units (128)** are better in pattern recognition at the cost of increased memory usage and potential overfitting.

Now, we proceed to refine our model. Our refinement involves experimenting different hyperparameters, architectures, and feature engineering techniques to optimise predictive accuracy.

To prevent overfitting, early stopping is used to monitor the validation loss per epoch. Early stopping halts training if the validation loss stops improving over several epochs, ensuring the model generalises well

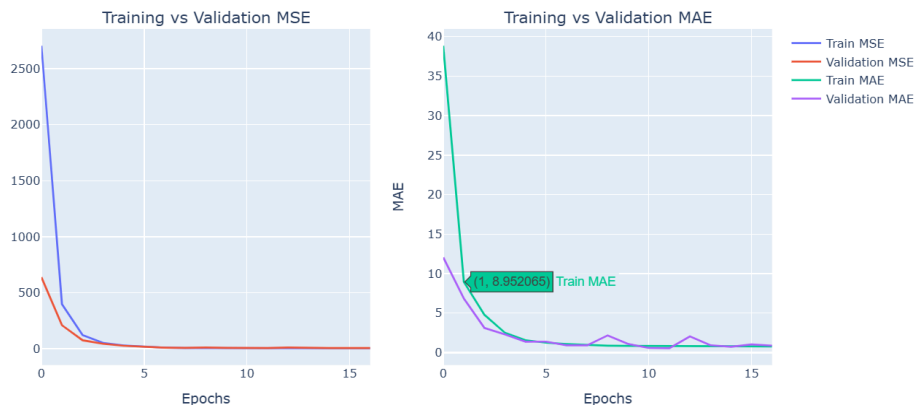


Figure 11: Mean Square Error (MSE) and Mean Absolute Error (MAE) Metrics against Epochs

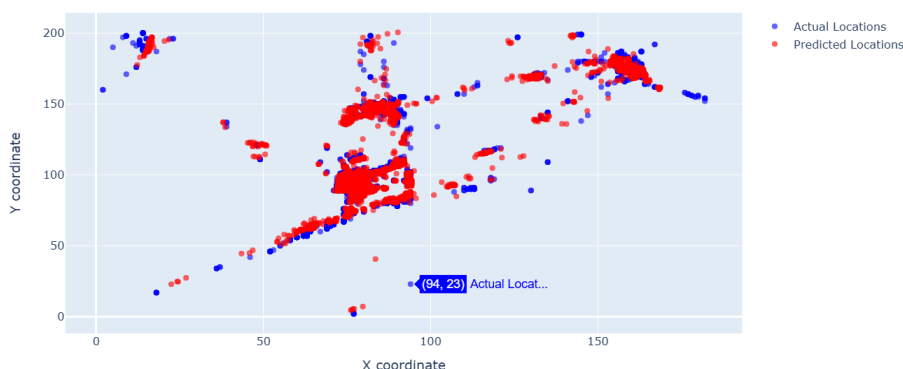


Figure 12: Actual vs. Predicted Locations (Scatter Grid Plot)

overfitting. The model performance on unseen data (validation set) is very close to the training set, which is desirable in predictive modelling.

Both the training and validation metrics (MSE and MAE) decrease rapidly in the first few epochs and then stabilise which indicates that the model is learning effectively during the initial stages and converges within a reasonable number of epochs. The validation error is low and remains stable without significant divergence from the training error, suggesting that the model is not overfitting. Since the validation curves (MSE and MAE) follow the training curves closely, there is no significant sign of

Insights Derived from Figure 12

Close Proximity of Points

The red and blue points are closely clustered in many areas, suggesting that our model is generally accurate in predicting the user's next location in Hiroshima.

Clusters and Movement Patterns	There are visible clusters in certain regions and these clusters can represent popular areas or frequent locations where users tend to move, such as high-traffic zones or popular POIs.
Areas of Deviation	In some cases, the red points (predictions) deviate noticeably from the blue points (actual locations). These deviations could indicate areas where our model struggles to predict accurately, possibly due to sparse data or complex movement patterns in these regions.

For further improvement, we can further analyse why certain regions show more deviations may help refine the model, especially by enhancing data representation or adjusting the model to handle areas with more complex patterns.

3.3: Further Analysis Geographic Clustering of POIs (K-Means)

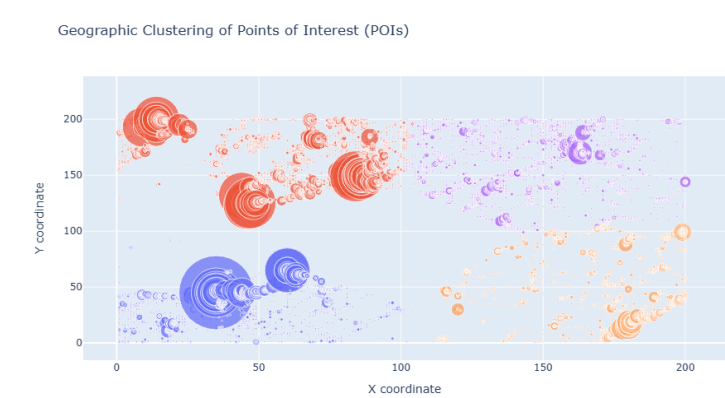


Figure 13: Geographic Clustering of Hiroshima’s POI

We employ K-Means Geographic Clustering to visualise Hiroshima POI’s density of the establishments as it offers an intuitive view of POI density and clustering patterns, valuable for identifying spatial patterns. Referring to Figure 13, the large circles indicate high-density POIs, suggesting areas with a high concentration of establishments or popular destinations. These regions could represent commercial or high-traffic zones. For instance, red and blue circles are key areas where users frequently move and the POI include shopping districts, business centres, or entertainment areas. The presence of distinct high-density clusters suggests that users will likely move between

or within these zones frequently and these insights are valuable for our understanding of potential movement patterns or common routes.

3.4: Prediction of User’s Subsequent Location

Results	User Characteristics
1/1 [=====] - 0s 52ms/step User 0, Current Location (Date 2, Time 37): Current Coordinates (x, y): (80, 101) Nearest Establishment: Bank at (80, 101) Predicted Next Location (x, y): (80.65, 100.63) Closest Establishment: Vet at (81, 101) -----	User (uid) 0 has been visiting various categories of establishments over time.
1/1 [=====] - 0s 44ms/step User 0, Current Location (Date 13, Time 44): Current Coordinates (x, y): (80, 101) Nearest Establishment: Bank at (80, 101) Predicted Next Location (x, y): (80.64, 100.60) Closest Establishment: Vet at (81, 101)	User (uid) 0 has been visiting categories of establishments in nearby locations over time.
1/1 [=====] - 0s 49ms/step User 82, Current Location (Date 70, Time 47): Current Coordinates (x, y): (80, 83) Nearest Establishment: School at (80, 83) Predicted Next Location (x, y): (80.55, 82.50) Closest Establishment: Church at (81, 82)	The user (uid) 82 has shown minimal to no movement over time.
1/1 [=====] - 0s 41ms/step User 108, Current Location (Date 69, Time 7): Current Coordinates (x, y): (93, 95) Nearest Establishment: Interior Shop at (93, 95) Predicted Next Location (x, y): (93.45, 94.76) Closest Establishment: Interior Shop at (93, 95)	The user (uid) 108 movement is concentrated only in early hours of the day.

References

- Abdullah Al Rahman. (2023, May 28). 1. What is LSTM? LSTM stands for Long Short-Term Memory. LinkedIn.com. <https://www.linkedin.com/pulse/lstm-networks-abdullah-al-rahman/>
- Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, 487–499.
- Antunes, C. & Oliveira, A. L. (2004). Sequential Pattern Mining Algorithms: Trade-offs Between Speed and Memory. <https://web.ist.utl.pt/claudia.antunes/artigos/antunes04mgts.pkdd.pdf>
- Döring, M., & Wagner, D. (2023). Trackintel: A Python package for human mobility analysis. *Trackintel Documentation*. Retrieved November 6, 2024, from <https://trackintel.readthedocs.io/en/latest/modules/model.html#trackintel.Positionfixes>
- Hartigan, J. A., & Wong, M. A. (1979). A K-means clustering algorithm. *Applied Statistics*, 28(1), 100–108. <https://doi.org/10.2307/2346830>
- Hikmawati, E., Maulidevi, N.U. & Surendro, K., (2021). Minimum threshold determination method based on dataset characteristics in association rule mining. *J Big Data* 8, 146. <https://doi.org/10.1186/s40537-021-00538-3>
- Srikant, R. & Agrawal, R. (1996, March). Mining Sequential Patterns: Generalizations and Performance Improvements. In *International conference on extending database technology* (pp. 1-17). Berlin, Heidelberg: Springer Berlin Heidelberg. <https://doi.org/10.1007/BFb0014140>

Contribution Summary

Brandon Jang Jin Tian: Task 3

Chung Zhi Xuan: Task 2

- Data preprocessing, experimental results for CityC and CityD

Ting Ruo Chee: Task 2

- Challenges of sequential pattern mining, experimental results for CityA and CityB

Yau Jun Hao: Task 1