Analysis of Venues Around MRT Stations in Singapore Using Foursquare Location Data

By: Choy Siew Fong

(Applied Data Science Capstone Project Submitted as part of the Requirements for IBM Professional Certificate in Data Science)

26 March 2020

1. Introduction

Singapore's rail network system consists of the Mass Rapid Transit (MRT), which is a heavy rail rapid transit system and the subsidiary light rail transit (LRT) system which serves the non-mature estates of Sengkang, Punggol and Bukit Panjang and function as a feeder network to the MRT. Figure 1 shows an example of an MRT train while Figure 2 shows a typical LRT train.

Figure 1: Photograph of an MRT train at Mandai Depot (Source: Mass Rapid Transit (Singapore). 18 March 2020. In *Wikipedia.* Retrieved March 19, 2020, https://en.wikipedia.org/wiki/Mass_Rapid_Transit_(Singapore)

Figure 2: An LRT train (Source: Light Rail Transit (Singapore). 18 March 2020. In *Wikipedia.* Retrieved March 19, 2020, https://en.wikipedia.org/wiki/Light_Rail_Transit_(Singapore)

Due to the astronomical prices of cars in Singapore, most of the island's population gets around by public transport. In 2019, the daily ridership of the MRT network was 3.4 million while the annual ridership was 1.2 billion. Since the MRT and LRT is such an integral part of people's daily lives in Singapore, I was interested to find out what kind of venues (such as restaurants, cafes, or supermarkets et cetera) existed around the MRT and LRT stations and if they have any trends or commonalities which may be analysed using clustering methods such as KMeans. This information may be obtained by combining a dataset of the stations' latitude and longitude coordinates with Foursquare location data.

1.1 Potential Stakeholders

This information about the type and number of venues available around each MRT/LRT station would be of interest to various stakeholders. Firstly, passengers or commuters taking the train would be keen to know what amenities are around each MRT or LRT station so that they can make informed decisions during their commute, such as which station to drop off if they wish to buy a loaf of bread from the supermarket for tomorrow's breakfast or which station would have an ATM from their bank.

Application developers may also have an interest in this information as it will enable them to develop mobile applications which can locate a certain venue (e.g. ATMs or cafes) which would be very useful to potential users.

Potential business owners who are contemplating starting a business around MRT stations due to their high footfall would also be interest in this information since it would allow them to know the number and type of competitors already existing in the area. For instance, a businessman interested in opening a Japanese restaurant in the Jurong East MRT area can use this information to find out how many *other* Japanese restaurants already exist in the area, and what are their specialities, if any. This information would allow prospective business owners to make informed decisions on which niche area they wish to target or to decide to move to a different location altogether.

Similarly, current business owners would be interested in knowing which venues are trending in the locale of their business. This would highlight potential competitors or trends which would enable them to adjust their business model and menu accordingly.

2. Description of the Dataset

The dataset 'Singapore Train Station Coordinates' was obtained from Kaggle (see https://www.kaggle.com/yxlee245/singapore-train-station-coordinates). It is in comma separated values (.csv) format and consists of four columns listing the station name, type (i.e. whether it is an MRT or an LRT station) and positional coordinates (latitude and longitude) of each MRT and LRT station in Singapore at the time of its upload, which was around eight months ago. Figure 3 lists the top 5 rows of the dataset.

1	station_name	type	lat	lng
2	Jurong East	MRT	1.333207	103.742308
3	Bukit Batok	MRT	1.349069	103.749596
4	Bukit Gombak	MRT	1.359043	103.751863
5	Choa Chu Kang	MRT	1.385417	103.744316

Figure 3: Screenshot from Excel showing the top 5 rows of the dataset

Preliminary inspection of the dataset found no missing or null values. However, it was later discovered during Folium map plotting that the coordinates for Admiralty Station were identical to Woodlands Station, probably due to a typographical error by the author of the dataset. To resolve this issue, I searched online for the coordinates of Admiralty station and replaced the erroneous longitude and latitude values for Admiralty station in the dataset with those from this site* (latitude = 1.44069˚, longitude = 103.8009˚).

*(https://www.findlatitudeandlongitude.com/l/Admiralty+mrt/1214628/).

The Foursquare information about each station would be obtained by defining a search URL (limiting the number of venues returned for each station to 100 within a radius of 500 m from each station) and sending the ***GET*** request to Foursquare API.

3. Methodology section

3.1 Exploratory Data Analysis

Using the ***pd.read_csv*** function, the dataset was first read into a Pandas dataframe, named df, as follows. It was noted that the column names were 'station_name', 'type', 'lat' and 'lng' (for latitude and longitude) as shown in Figure 4 below.

```
In [2]: # read data file
        df = pd.read_csv('mrt_lrt_data2.csv')
        df.head()
```

Out[2]:

	station_name	type	lat	lng
0	Jurong East	MRT	1.333207	103.742308
1	Bukit Batok	MRT	1.349069	103.749596
2	Bukit Gombak	MRT	1.359043	103.751863
3	Choa Chu Kang	MRT	1.385417	103.744316
4	Yew Tee	MRT	1.397383	103.747523

Figure 4: Top 5 rows of the dataset when read into a dataframe df

To understand more about the dataset, the ***dtypes*** command was used to find out the data type of each column. This enables us to determine which functions can be applied and to do any conversions to other data types, if needed. As shown in Figure 5, the 'station_name' and 'type' fields were found to be of type *Object* while the 'lat' and 'lng' fields were of type *Float*.

```
# Basic type information about data
df.dtypes

station_name     object
type             object
lat             float64
lng             float64
dtype: object
```

Figure 5: Basic data type information of the dataset obtained with dtypes command

The data was then grouped by type of stations using the ***groupby*** function and the number of each station type obtained with the ***count*** function. There are 119 MRT stations and 38 LRT stations in the dataset.

```
#check number of LRT and MRT stations
df_group = df.groupby('type').count()
df_group
```

	station_name	lat	lng
type			
LRT	38	38	38
MRT	119	119	119

Figure 6: Output showing the number of each type of stations

It was decided at this point to confine the subsequent analysis to MRT stations only as these were the main stations while LRT stations are in the suburbs and may have few interesting venues around them if any. Moreover, the free Foursquare developer sandbox account allows only 950 regular calls a day. Using the full dataset may lead to issues with hitting the call limit later, given that each station is expected to have a substantial number of venues.

A new dataframe containing only the MRT stations was then defined as follows.

```
#Get MRT data into new dataframe - confine analysis to MRT stations
df_MRT = df.loc[df['type'] == 'MRT']
df_MRT.head()
```

Figure 7: Defining new dataframe df_MRT containing only MRT station data.

The shape of the new dataframe was checked using the ***shape*** attribute as below.

```
In [6]:  # check size of MRT data
         df_MRT.shape

Out[6]:  (119, 4)
```

Figure 8: Checking shape of the resulting dataframe

This output confirms that the new **df_MRT** dataframe did contain information from all 119 MRT stations that were in the original dataset. The data was then visualised as a map using **Folium.**

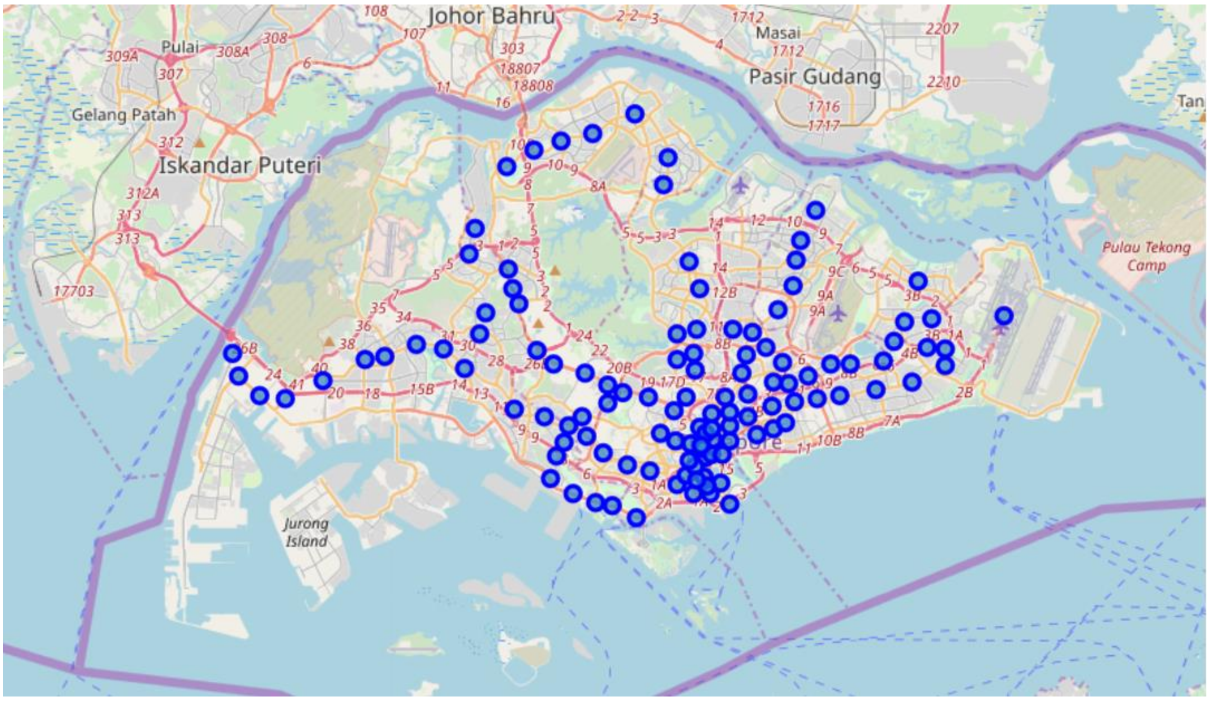


Figure 9: Folium map showing the locations of 119 MRT stations

3.2 Exploring the first MRT station

It is time to start using Foursquare API to get location data about the stations. I began the analysis by defining my Foursquare credentials (client ID, client secret and version). The first station in the dataframe was then identified using ***.loc*** command, with zero as the first row index.

```
#first station locaton
df_MRT.loc[0, 'station_name']

'Jurong East'
```

Figure 10: Finding the first station name from df_MRT

The latitude and longitude of Jurong East station was defined and a ***GET*** URL request sent to get information of 100 venues located within 500 m from the station.

```
In [12]:  results = requests.get(url).json() #get results
          results

Out[12]:  {'meta': {'code': 200, 'requestId': '5e74dc66660a9f001ea09003'},
           'response': {'suggestedFilters': {'header': 'Tap to show:',
             'filters': [{'name': 'Open now', 'key': 'openNow'}]},
            'headerLocation': 'Jurong East',
            'headerFullLocation': 'Jurong East, Singapore',
            'headerLocationGranularity': 'neighborhood',
            'totalResults': 76,
            'suggestedBounds': {'ne': {'lat': 1.3377070045000046,
              'lng': 103.74680081871648},
             'sw': {'lat': 1.3287069954999955, 'lng': 103.73781518128351}},
            'groups': [{'type': 'Recommended Places',
              'name': 'recommended',
              'items': [{'reasons': {'count': 0,
```

Figure 11: Json file output by the GET request

A get category type function was defined and the json output normalised. The resulting dataframe was then filtered to display only location related data such as venue name, venue category and venue coordinates (as latitude and longitude) (Figure 12). The output was then checked by calling the ***head*** function (Figure 13).

```
In [13]: #define get_category_type function
         def get_category_type(row):
             try:
                 categories_list = row['categories']
             except:
                 categories_list = row['venue.categories']
             if len(categories_list) == 0:
                 return None
             else:
                 return categories_list[0]['name']

In [14]: venues = results['response']['groups'][0]['items']
         nearby_venues = json_normalize(venues)

In [15]: # filter columns
         filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
         nearby_venues =nearby_venues.loc[:, filtered_columns]
         # filter the category for each row
         nearby_venues['venue.categories'] = nearby_venues.apply(get_category_type, axis=1)
         # clean columns
         nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]
         nearby_venues.head()
```

Figure 12: Defining the get category type function and filtering and cleaning the columns

Out[15]:

	name	categories	lat	lng
0	UNIQLO	Clothing Store	1.333175	103.743160
1	MUJI 無印良品	Furniture / Home Store	1.333187	103.743064
2	Song Fa Bak Kut Teh 松發肉骨茶	Chinese Restaurant	1.333394	103.743420
3	Johan Paris	Bakery	1.334083	103.742384
4	The Rink	Skating Rink	1.333424	103.740345

Figure 13: First 5 rows of nearby_venues dataframe

The venues found around Jurong East station was visualised using ***Folium.Map*** function. A red marker was added to indicate the location of the MRT station while blue markers were used to denote the venues.

```
In [16]: #Map for regions
         venues_map = folium.Map(location=[station_latitude, station_longitude], zoom_start=20)
         # add a red circle marker to represent the Jurong East station
         folium.features.CircleMarker(
             [station_latitude, station_longitude],
             radius=10,
             color='red',
             popup='Jurong East',
             fill = True,
             fill_color = 'red',
             fill_opacity = 0.6
         ).add_to(venues_map)
         # add all venues as blue circle markers
         for lat, lng, label in zip(nearby_venues.lat, nearby_venues.lng, nearby_venues.categories):
             folium.features.CircleMarker(
                 [lat, lng],
                 radius=5,
                 color='blue',
                 popup=label,
                 fill = True,
                 fill_color='blue',
                 fill_opacity=0.6
             ).add_to(venues_map)

         venues_map
```

Out[16]:

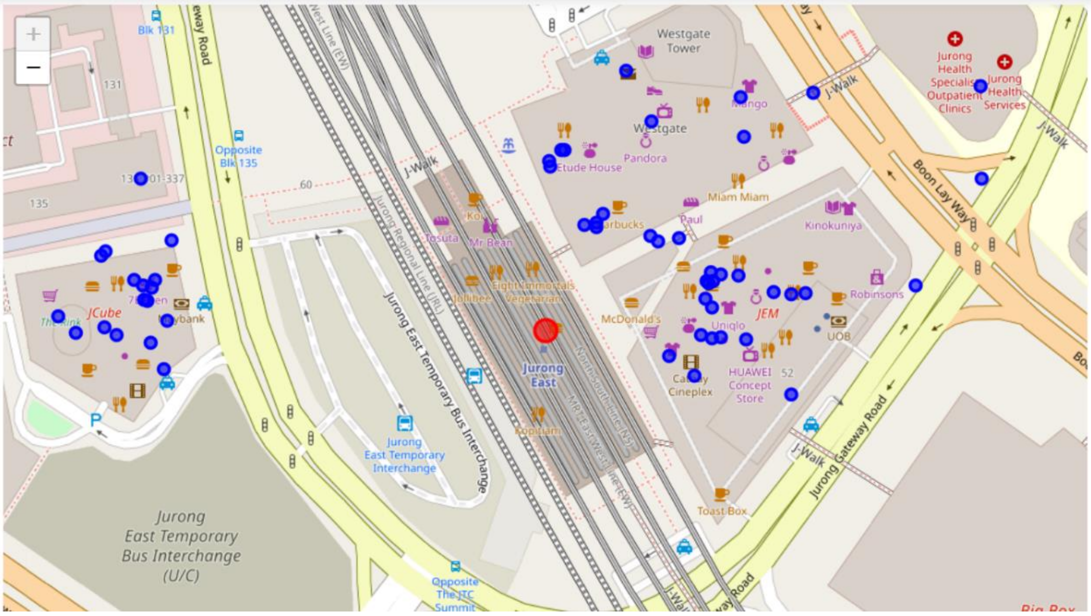


Figure 14: Map of venues found from Foursquare located within 500 m around the first MRT station, Jurong East.

Using the ***shape*** function on nearby_venues dataframe, it was found that a total of 73 venues were returned around Jurong East station by Foursquare.

```
In [17]: print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0])) # number of venues returned

         73 venues were returned by Foursquare.
```

Figure 15: Total number of venues returned by Foursquare

3.3 Expanding the analysis to the full dataset

Now the same analysis can be performed on all the MRT stations contained in df_MRT. First, a getNearbyVenues function was defined. It was then applied to df_MRT and the results were output to a new dataframe called station_venues.

```
station_venues = getNearbyVenues(names=df_MRT['station_name'],
                                 latitudes=df_MRT['lat'],
                                 longitudes=df_MRT['lng'])
```

Figure 16: Applying the defined function getNearbyVenues to df_MRT

Once again, the shape function was used to check the size of the resultant dataframe. It was found that the station_venues dataframe consisted of 4712 rows and 7 columns.

```
In [20]: #check size of station_venues
         print(station_venues.shape)
         station_venues.sample(10)

         (4712, 7)
```

Figure 17: Shape of station_venues dataframe

To check the output of the dataframe station_venues, the ***sample*** function was used to return a random sample of 10 stations' data. Using the ***head*** function will return only the first station's data and is not as representative of the full dataframe as the sample function.

	Station	Station_Latitude	Station_Longitude	Venue	Venue_Latitude	Venue_Longitude	Venue_Category
3728	Telok Blangah	1.270769	103.809878	Telok Blangah Drive Blk 82 Market	1.273392	103.807595	Market
83	Bukit Batok	1.349069	103.749596	NTUC FairPrice	1.348814	103.749248	Grocery Store
3130	Esplanade	1.293995	103.855396	Narrative Coffee Stand	1.297118	103.854371	Coffee Shop
89	Bukit Batok	1.349069	103.749596	Bee Cheng Hiang	1.349794	103.748235	Asian Restaurant
1947	Eunos	1.319809	103.902888	Benji Pet Kennel	1.318012	103.906195	Pet Store
2343	HarbourFront	1.265453	103.820514	Play Court | Vivocity	1.263891	103.822152	Playground
4606	Geylang Bahru	1.321479	103.871457	Beng Soon Seafood 明顺海鲜	1.323005	103.868693	Seafood Restaurant
2070	Simei	1.343237	103.953343	Each-A-Cup	1.342812	103.953248	Juice Bar
1562	Tanjong Pagar	1.276385	103.846771	Quan Ji @ Amoy Street Food Market	1.279100	103.847392	Chinese Restaurant
3665	Haw Par Villa	1.283149	103.781991	Haw Par Villa	1.283855	103.781421	Sculpture Garden

Figure 18: Sample containing 10 rows data from station_venues dataframe.

3.4 Analysing the station_venues dataframe

The station_venues dataframe would be the main dataframe being analysed from this point onwards as it contains the combined data of both MRT station names and location, as well as nearby venues name, category and location.

First, preliminary analysis was performed using the ***unique*** function to find out the number of unique venues and venue category being returned by Foursquare.

```
In [21]: #find unique categories number
         print('There are {} unique venue categories.'.format(len(station_venues['Venue_Category'].unique())))

         There are 327 unique venue categories.

In [22]: #find unique venue number
         print('There are {} uniques venues.'.format(len(station_venues['Venue'].unique())))#number of unique venues

         There are 3226 uniques venues.
```

Figure 19: Number of unique venue categories and unique venues

A total of *327 unique venue categories* and *3226 unique venues* were found around 119 MRT stations in the dataset. The total number of venues returned for each MRT station was determined using the ***groupby*** and ***count*** functions as follows.

```
station_venues.groupby('Station').count()
```

Figure 20: Using groupby and count functions to find the total number of venues for each MRT station

Out[23]:

Station	Station_Latitude	Station_Longitude	Venue	Venue_Latitude	Venue_Longitude	Venue_Category
Admiralty	8	8	8	8	8	8
Aljunied	47	47	47	47	47	47
Ang Mo Kio	41	41	41	41	41	41
Bartley	12	12	12	12	12	12
Bayfront	50	50	50	50	50	50
Beauty World	76	76	76	76	76	76
Bedok	56	56	56	56	56	56
Bedok North	17	17	17	17	17	17
Bedok Reservoir	5	5	5	5	5	5
Bencoolen	100	100	100	100	100	100

Figure 21: Screenshot showing the first 10 rows of the station_venues dataframe

To perform further analysis on the station_venues dataframe, it is necessary to convert the categorical values for Venue_Category to numerical values (i.e. one hot vectors). This conversion (one hot encoding) was done using the ***get_dummies*** function (see Jupyter notebook for details). The rows in the resulting dataframe were then grouped by station and taking the mean of the frequency of occurrence of each category. The resultant dataframe, called station_grouped, had 119 rows and 328 columns.

	Station	ATM	Accessories Store	Airport	Airport Lounge	American Restaurant	Arcade	Art Gallery	Art Museum	Arts & Crafts Store	...	Water Park	Waterfall	Waterfront	Whisky Bar
0	Admiralty	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000
1	Aljunied	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000
2	Ang Mo Kio	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000
3	Bartley	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000
4	Bayfront	0.0	0.020000	0.000000	0.000000	0.000000	0.000000	0.020000	0.00	0.000000	...	0.000000	0.000000	0.040000	0.000000
5	Beauty World	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000
6	Bedok	0.0	0.000000	0.000000	0.000000	0.017857	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000
7	Bedok North	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000
8	Bedok Reservoir	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000
9	Bencoolen	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.030000	0.02	0.010000	...	0.000000	0.000000	0.000000	0.020000
10	Bendemeer	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	...	0.000000	0.000000	0.000000	0.000000

Figure 22: Screenshot showing station_grouped dataframe

The top 5 venue category around each station were printed to get an idea of which venues were common as shown below.

```
In [28]: #print each station along with the top 5 most common venues
         num_top_venues = 5
         for stn in station_grouped['Station']:
             print("----"+stn+"----")
             temp = station_grouped[station_grouped['Station'] == stn].T.reset_index()
             temp.columns = ['venue','freq']
             temp = temp.iloc[1:]
             temp['freq'] = temp['freq'].astype(float)
             temp = temp.round({'freq': 2})
             print(temp.sort_values('freq', ascending=False).reset_index(drop=True).head(num_top_venues))
             print('\n')
```

```
----Admiralty----
             venue  freq
0      Supermarket  0.25
1     Optical Shop  0.12
2  Sushi Restaurant  0.12
3       Food Court  0.12
4      Snack Place  0.12


----Aljunied----
                          venue  freq
0            Chinese Restaurant  0.11
1              Asian Restaurant  0.09
2                  Noodle House  0.09
3                   Coffee Shop  0.06
4  Vegetarian / Vegan Restaurant  0.06
```

Figure 23: Printing the top 5 venue categories for each station.

The venue categories were then sorted in descending order and a new dataframe, station_venues_sorted, containing the top ten venue categories for each MRT station created as shown below.

Out[30]:

	Station	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Admiralty	Supermarket	Coffee Shop	Park	Sushi Restaurant	Optical Shop	Snack Place	Food Court	Food & Drink Shop	Filipino Restaurant	Fish & Chips Shop
1	Aljunied	Chinese Restaurant	Noodle House	Asian Restaurant	Coffee Shop	Vegetarian / Vegan Restaurant	Bus Station	Indian Restaurant	Dim Sum Restaurant	Food Court	Seafood Restaurant
2	Ang Mo Kio	Coffee Shop	Dessert Shop	Food Court	Supermarket	Japanese Restaurant	Bubble Tea Shop	Ramen Restaurant	Snack Place	Sushi Restaurant	Modern European Restaurant
3	Bartley	Bus Station	Noodle House	Pet Store	Café	Bus Stop	Food Truck	Indian Restaurant	Metro Station	Frozen Yogurt Shop	Fried Chicken Joint
4	Bayfront	Hotel	Boutique	Bar	Theater	Bridge	Lounge	Japanese Restaurant	Tea Room	Italian Restaurant	Waterfront

Figure 24: First 5 rows of the station_venues_sorted dataframe showing top 10 venue category for each station

3.5 KMeans Clustering

The station_grouped dataframe was then analysed using KMeans clustering. The number of clusters was estimated using the Elbow method, which found an optimised cluster size of 6.

```
In [32]: station_grouped_clustering = station_grouped.drop('Station', 1)
         Error =[]
         for i in range(2, 8):
             kmeans = KMeans(n_clusters = i).fit(station_grouped_clustering)
             kmeans.fit(station_grouped_clustering)
             Error.append(kmeans.inertia_)
         import matplotlib.pyplot as plt
         plt.plot(range(2, 8), Error)
         plt.title('Elbow method')
         plt.xlabel('No of clusters')
         plt.ylabel('Error')
         plt.show()
```

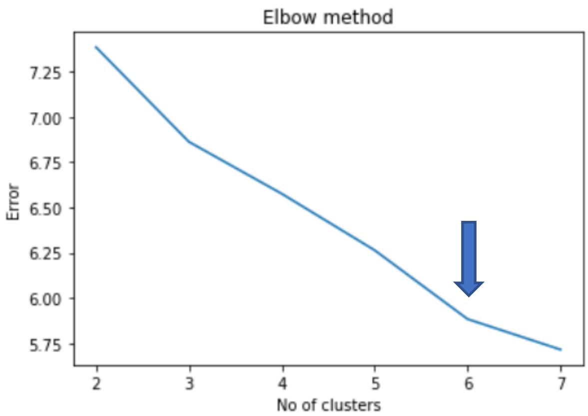


Figure 25: Finding the optimum number of clusters

KMeans clustering was run after dropping the first column ‘Station’ as it had a non-numerical value. The resulting dataframe was then merged with the original df_MRT to include the latitude and longitude data. Figure 26 below shows the relevant part of the code.

```
In [33]: # set number of clusters
         kclusters = 6
         station_grouped_clustering = station_grouped.drop('Station', 1)
         # run k-means clustering
         kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(station_grouped_clustering)
         # check cluster labels generated for each row in the dataframe
         kmeans.labels_[0:10]

Out[33]: array([1, 3, 1, 4, 4, 4, 4, 4, 3, 4])

In [34]: # add clustering labels
         station_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
         station_merged = df_MRT
         # merge station_grouped with station_data to add latitude/longitude for each station
         station_merged = station_merged.join(station_venues_sorted.set_index('Station'), on='station_name')
         station_merged.head() # check dataframe
```

Figure 26: Clustering using the station_grouped dataframe

The clusters were then visualised in a Folium map as shown below.

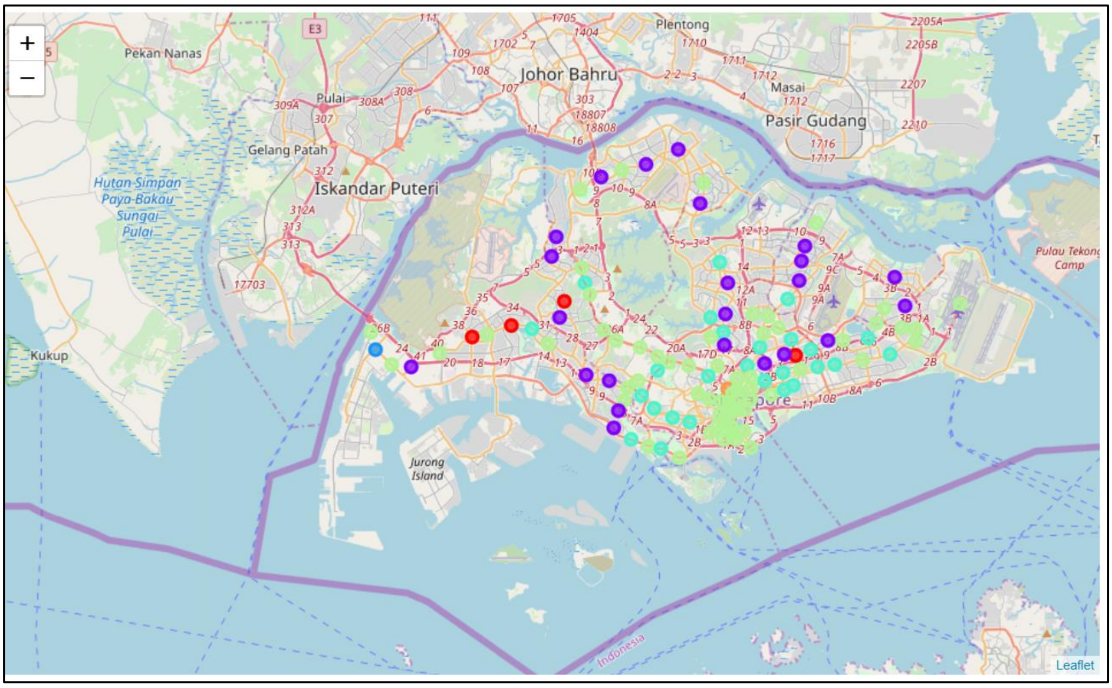


Figure 27: Folium map showing the 6 clusters

Each of the clusters were then examined in turn to check for any commonalities among the stations in the same cluster.

In [56]:
```
#Examine the clusters - cluster 1
station_merged.loc[station_merged['Cluster Labels'] == 0, station_merged.columns[[0] + list(range(5, station_merged.shape[1]))]]
```

Out[56]:

	station_name	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
2	Bukit Gombak	Food Court	Vegetarian / Vegan Restaurant	Stadium	Ice Cream Shop	Steakhouse	Supermarket	Flea Market	Fast Food Restaurant	Chinese Restaurant	Sandwich Place
31	Pioneer	Gym / Fitness Center	Bus Station	Stadium	Convenience Store	Pool	Fast Food Restaurant	Shopping Mall	Food Court	Snack Place	Bus Line
33	Lakeside	Food Court	Convenience Store	Trail	Snack Place	Food & Drink Shop	Field	Filipino Restaurant	Fish & Chips Shop	Flea Market	Flower Shop
79	MacPherson	Food Court	Climbing Gym	Basketball Court	Hobby Shop	BBQ Joint	Thai Restaurant	Food Truck	Asian Restaurant	Bakery	Office

Figure 28: Cluster 1 - ten most common venues

In [58]:
```
#Examine the clusters - cluster 2
station_merged.loc[station_merged['Cluster Labels'] == 1, station_merged.columns[[0] + list(range(5, station_merged.shape[1]))]]
```

Out[58]:

	station_name	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
1	Bukit Batok	Coffee Shop	Fast Food Restaurant	Food Court	Chinese Restaurant	Frozen Yogurt Shop	Bus Station	Bowling Alley	Multiplex	Café	Shopping Mall
3	Choa Chu Kang	Coffee Shop	Fast Food Restaurant	Ice Cream Shop	Smoke Shop	Gym	Thai Restaurant	Sushi Restaurant	Supermarket	Bubble Tea Shop	Food Court
4	Yew Tee	Fast Food Restaurant	Japanese Restaurant	Pool	Electronics Store	Café	Sandwich Place	Coffee Shop	Food Court	Diner	Shopping Mall
6	Marsiling	Coffee Shop	Flea Market	Steakhouse	Grocery Store	BBQ Joint	Asian Restaurant	Hainan Restaurant	Paintball Field	Track	Trail
8	Admiralty	Supermarket	Coffee Shop	Park	Sushi Restaurant	Optical Shop	Snack Place	Food Court	Food & Drink Shop	Filipino Restaurant	Fish & Chips Shop
9	Sembawang	Coffee Shop	Japanese Restaurant	Asian Restaurant	Bus Station	Supermarket	Bistro	Shopping Mall	Chinese Restaurant	Train Station	BBQ Joint
11	Khatib	Coffee Shop	Asian Restaurant	Supermarket	Food Court	Grocery Store	Shopping Mall	Bakery	Train Station	Park	Seafood Restaurant

Figure 29: Cluster 2 - ten most common venues

```
In [59]: #Examine the clusters - cluster 3
station_merged.loc[station_merged['Cluster Labels'] == 2, station_merged.columns[[0] + list(range(5, station_merged.shape[1]))]]
```

Out[59]:

	station_name	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
27	Tuas West Road	Bus Station	Fast Food Restaurant	Filipino Restaurant	Fish & Chips Shop	Flea Market	Flower Shop	Food	Food & Drink Shop	Food Court	Food Stand

Figure 30: Cluster 3 - ten most common venues

```
In [60]: #Examine the clusters - cluster 4
station_merged.loc[station_merged['Cluster Labels'] == 3, station_merged.columns[[0] + list(range(5, station_merged.shape[1]))]]
```

Out[60]:

	station_name	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
12	Yio Chu Kang	Food Court	Chinese Restaurant	Gym	Dance Studio	College Auditorium	Bus Stop	Cafeteria	Fast Food Restaurant	Tennis Court	Seafood Restaurant
15	Braddell	Noodle House	Chinese Restaurant	Food Court	Café	Hakka Restaurant	Asian Restaurant	Seafood Restaurant	Fast Food Restaurant	Bakery	Thai Restaurant
18	Newton	Chinese Restaurant	Seafood Restaurant	Hotel Bar	Italian Restaurant	Gym / Fitness Center	Convenience Store	Pool	Pizza Place	Café	Food Court
34	Chinese Garden	Chinese Restaurant	Coffee Shop	Bus Station	Train Station	Café	Pizza Place	Asian Restaurant	Food Court	Indian Restaurant	Food Truck
38	Commonwealth	Chinese Restaurant	Indian Restaurant	Asian Restaurant	Coffee Shop	Vegetarian / Vegan Restaurant	Noodle House	Diner	Fast Food Restaurant	Paper / Office Supplies Store	Food Court
39	Queenstown	Noodle House	Food Court	Chinese Restaurant	BBQ Joint	Stadium	Spa	Food & Drink Shop	Café	Seafood Restaurant	Train Station

Figure 31: Cluster 4 - ten most common venues

```
In [61]: #Examine the clusters - cluster 5
station_merged.loc[station_merged['Cluster Labels'] == 4, station_merged.columns[[0] + list(range(5, station_merged.shape[1]))]]
```

Out[61]:

	station_name	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Jurong East	Japanese Restaurant	Coffee Shop	Chinese Restaurant	Food Court	Shopping Mall	Steakhouse	Korean Restaurant	Multiplex	Department Store	Café
5	Kranji	Bus Line	Noodle House	Go Kart Track	Dessert Shop	Stadium	Racetrack	Food Stand	Fish & Chips Shop	Flea Market	Flower Shop
7	Woodlands	Japanese Restaurant	Café	Coffee Shop	Clothing Store	Shopping Mall	Chinese Restaurant	Asian Restaurant	Frozen Yogurt Shop	Fast Food Restaurant	Electronics Store
10	Yishun	Chinese Restaurant	Food Court	Supermarket	Park	Coffee Shop	Italian Restaurant	Hainan Restaurant	Fried Chicken Joint	Bubble Tea Shop	Café
17	Novena	Café	Coffee Shop	Japanese Restaurant	Hotel	Chinese Restaurant	Italian Restaurant	Ramen Restaurant	Dessert Shop	Sandwich Place	Restaurant
19	Orchard	Boutique	Sushi Restaurant	Bakery	Hotel	Shopping Mall	Chinese Restaurant	Bubble Tea Shop	Café	Cosmetics Shop	Coffee Shop

Figure 32: Cluster 5 – ten most common venues

```
In [62]: #Examine the clusters - cluster 6
station_merged.loc[station_merged['Cluster Labels'] == 5, station_merged.columns[[0] + list(range(5, station_merged.shape[1]))]]
```

Out[62]:

	station_name	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
61	Little India	Indian Restaurant	Vegetarian / Vegan Restaurant	Coffee Shop	General College & University	Bakery	Playground	Music Venue	Restaurant	Hotel	Motel

Figure 33: Cluster 6 - – ten most common venues

3.6 Determining the Station with the Highest Venue Category

Since each station can have a variable number of venue categories and a certain number of each category, I would like to find out which station has the maximum number of a venue category, and what category it is. This was done using the **groupby** and **size** functions.

```
In [63]: #group by station and find number of venue category of each type
         stn_grp = station_venues.groupby(['Station', 'Venue_Category']).size()
         stn_grp

Out[63]: Station    Venue_Category
         Admiralty  Coffee Shop                        1
                    Food Court                         1
                    Optical Shop                       1
                    Park                               1
                    Snack Place                        1
                    Supermarket                        2
                    Sushi Restaurant                   1
         Aljunied   Asian Restaurant                   4
                    BBQ Joint                          1
                    Badminton Court                    1
                    Basketball Court                   1
                    Boarding House                     1
                    Breakfast Spot                     1
                    Bus Station                        2
                    Café                               1
                    Chinese Restaurant                 5
                    Coffee Shop                        3
                    Convenience Store                  1
                    Dim Sum Restaurant                 2
                    Farmers Market                     1
                    Food Court                         2
                    Food Truck                         1
                    Hotel                              1
                    Indian Restaurant                  2
```

Figure 34: Grouping stations by station and Venue Category with resultant output

The station with the maximum number of venue category was then listed using the **idxmax** function.

```
In [64]: #find station with max number of venue category
         stn_grp.idxmax()

Out[64]: ('Jalan Besar', 'Indian Restaurant')
```

Figure 35: Station with maximum number of a venue category

Surprisingly, Jalan Besar returned the greatest number of venue category (Indian restaurant). The Indian restaurants in Jalan Besar were called from the station_venues dataframe. A total of 21 indian restaurants were returned (see Figure 36).

	Station	Station_Latitude	Station_Longitude	Venue	Venue_Latitude	Venue_Longitude	Venue_Category
4489	Jalan Besar	1.305551	103.855443	Bismillah Biryani	1.304956	103.853602	Indian Restaurant
4495	Jalan Besar	1.305551	103.855443	Murugan Idli Shop	1.308842	103.856380	Indian Restaurant
4502	Jalan Besar	1.305551	103.855443	Azmi Restaurant	1.308256	103.853075	Indian Restaurant
4503	Jalan Besar	1.305551	103.855443	Sakunthala's Restaurant	1.306000	103.852169	Indian Restaurant
4509	Jalan Besar	1.305551	103.855443	Komala Vilas (Buffalo Rd)	1.306308	103.851158	Indian Restaurant
4512	Jalan Besar	1.305551	103.855443	Khansama Tandoori Restaurant	1.308251	103.853122	Indian Restaurant
4522	Jalan Besar	1.305551	103.855443	Kailash Parbat	1.308039	103.852660	Indian Restaurant
4523	Jalan Besar	1.305551	103.855443	Lagnaa Barefoot Dining	1.306472	103.852298	Indian Restaurant
4531	Jalan Besar	1.305551	103.855443	Veeras Curry Restaurant @ Hindoo Rd	1.308650	103.853515	Indian Restaurant
4535	Jalan Besar	1.305551	103.855443	Sakunthala's Restaurant	1.309475	103.855717	Indian Restaurant
4542	Jalan Besar	1.305551	103.855443	Premaas Cuisine	1.305094	103.851980	Indian Restaurant
4552	Jalan Besar	1.305551	103.855443	Taste Of India	1.307900	103.852089	Indian Restaurant
4555	Jalan Besar	1.305551	103.855443	The Banana Leaf Apolo	1.305348	103.851698	Indian Restaurant
4557	Jalan Besar	1.305551	103.855443	Islamic Restaurant	1.303075	103.859172	Indian Restaurant
4559	Jalan Besar	1.305551	103.855443	Kebabs 'n Curries	1.309157	103.856456	Indian Restaurant
4562	Jalan Besar	1.305551	103.855443	Allauddin's Bryiani	1.305799	103.851180	Indian Restaurant
4563	Jalan Besar	1.305551	103.855443	Ma Raj Restaurant	1.309817	103.855970	Indian Restaurant
4565	Jalan Besar	1.305551	103.855443	Saravanaa Bhavan	1.309272	103.855971	Indian Restaurant
4567	Jalan Besar	1.305551	103.855443	Copper Chimney	1.309712	103.855447	Indian Restaurant
4571	Jalan Besar	1.305551	103.855443	Ananda Bhavan Restaurant	1.309665	103.855614	Indian Restaurant
4576	Jalan Besar	1.305551	103.855443	Yakader	1.305794	103.851108	Indian Restaurant

Figure 36: Total number of Indian restaurants around Jalan Besar station.

3.7 Top 10 Venue Category for all Stations

To determine the top ten most common venue category for all MRT stations, the stn_grp object returned in Figure 27 was converted to a dataframe, which was grouped by Venue Category and the counts for each venue category summed up. The resultant output is as follows:

```
Out[67]: Venue_Category
         ATM                            1
         Accessories Store              4
         Airport                        2
         Airport Lounge                 5
         American Restaurant           21
         Arcade                         3
         Art Gallery                   21
         Art Museum                     2
         Arts & Crafts Store            9
         Arts & Entertainment           1
         Asian Restaurant             127
         Athletics & Sports             2
         Australian Restaurant          5
         BBQ Joint                     44
         Baby Store                     1
         Badminton Court                1
         Bagel Shop                     2
         Bakery                       118
         Bank                           2
         Bar                           43
         Basketball Court               9
```

Figure 37: Count for each venue category for all MRT stations

This list was then sorted in descending order and the top ten venue categories displayed using the **head(10)** command.

```
In [68]: Alist.sort_values(ascending = False, inplace = True) #sort venue category in descending order
         Top_10 = Alist.head(10) #find top 10 venue category
         Top_10

Out[68]: Venue_Category
         Coffee Shop              245
         Chinese Restaurant       236
         Café                     218
         Food Court               182
         Japanese Restaurant      175
         Hotel                    136
         Asian Restaurant         127
         Indian Restaurant        121
         Bakery                   118
         Fast Food Restaurant      98
         Name: Counts, dtype: int64
```

Figure 38: Top 10 list of venue category

To visualise the results, matplotlib bar chart was used to plot the top 10 categories.

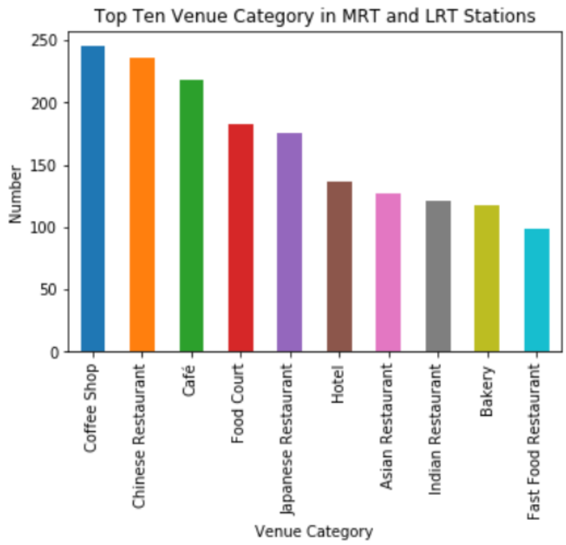


Figure 39: Bar chart showing top 10 venue categories for all MRT stations

3.8 Determining the Station with the Maximum Number of a Venue

A similar analysis was done to determine the station with the highest number of one particular venue. It turned out that Downtown station had the maximum number of Starbucks as shown below.

```
In [71]: stn_grp2.idxmax() #Find venue with max number and the corresponding station

Out[71]: ('Downtown', 'Starbucks')
```

Figure 40: Output showing Downtown station having the highest number of Starbucks outlets

How many Starbucks are there around Downtown station? It turned out that there are 4 Starbucks around this station alone.

Out[72]:

	Station	Station_Latitude	Station_Longitude	Venue	Venue_Latitude	Venue_Longitude
4110	Downtown	1.27949	103.852802	Starbucks	1.279335	103.854128
4142	Downtown	1.27949	103.852802	Starbucks	1.277949	103.850985
4161	Downtown	1.27949	103.852802	Starbucks	1.276988	103.852458
4176	Downtown	1.27949	103.852802	Starbucks	1.279422	103.854494

Figure 41: Output showing location coordinates of the 4 Starbucks around Downtown station

To find out how close these different Starbucks outlets are, I use Folium map to visualise their locations as follows.

Figure 41: Map showing location of the four Starbucks outlets around Downtown station.

3.9 ATMs Around MRT Stations

Lastly, I decided to determine which stations had automatic teller machines in their vicinity (ATM) by filtering by 'Venue Category' == 'ATM'. Surprisingly, the output returned only one station, Commonwealth.

Out[74]:

	Station	Station_Latitude	Station_Longitude	Venue_Category	Venue	Venue_Latitude	Venue_Longitude
1400	Commonwealth	1.302439	103.798326	ATM	POSB ATM	1.300509	103.801128

Figure 42: ATMs around MRT stations

4. Results

A total of 327 unique venue categories and 3226 unique venues were found around the 119 MRT stations. Analysis of the top 5 venue categories around each station found that eateries such as restaurants, coffee shops and cafes, featured prominently, proving that Singapore is indeed a foodie nation. Using the elbow method, an optimum cluster size of 6 was obtained and used for KMeans clustering.

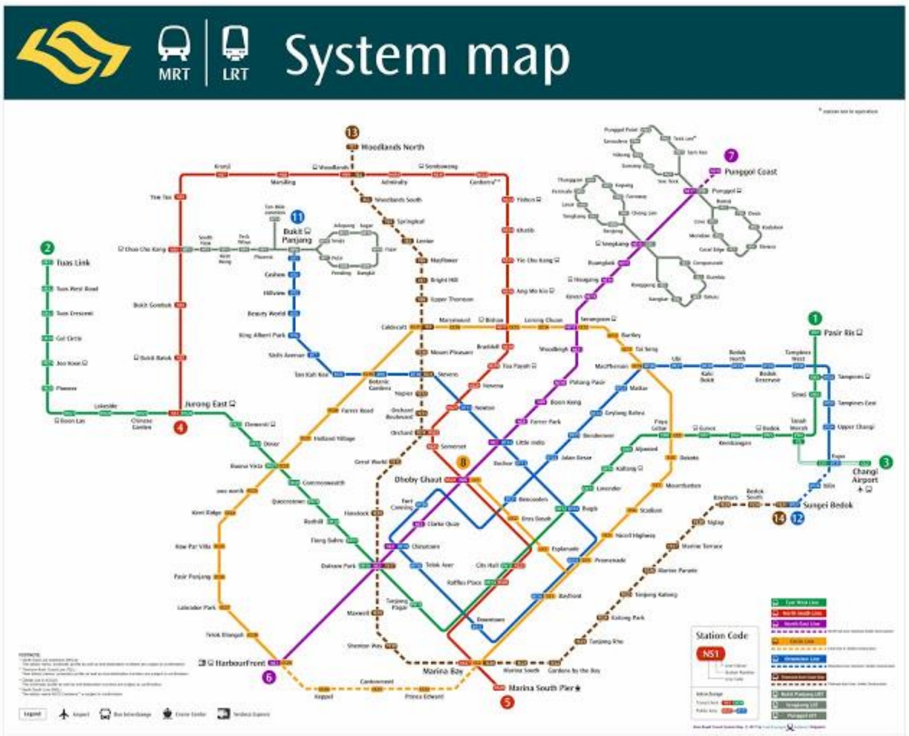
Analysis of the clustering results revealed cluster 1 (label 0) to be composed of 4 stations: Bukit Gombak, Pioneer, Lakeside and MacPherson. These were characterised by having a food court and a fitness related venue (gym or stadium) as their top venues. Cluster 2 consisted of 23 stations with a coffee shop as the top venue categories. Cluster 3 consisted of only one station – Tuas West Road, and it was distinguished by having a bus stop as the top venue category. Cluster 4 had 26 stations in total. This cluster had Chinese Restaurant as the top venue category in their vicinity. Cluster 5 proved to be the largest cluster, with 64 stations. This cluster had a more eclectic mix of categories, characterised by retail (shopping mall, supermarket or flea markets) or hotels. Cluster 6 consisted of only Little India station, with Indian restaurant and vegetarian restaurants as the top venue categories.

Jalan Besar had the maximum number (21) of one venue category, i.e. Indian restaurants. Overall, the top 10 venue categories around the MRT stations are coffee shops, Chinese restaurants, café, food court, Japanese restaurant, hotel, Asian restaurant, bakery, Indian restaurant and fast food restaurant in that order. An overwhelming 9 out of the 10 top venue categories were associated with food.

The most common venue in any MRT station turned out to be the American coffee chain Starbucks, which had 4 outlets around Downtown station alone. Only one ATM was found at Commonwealth station, an abnormality which would be discussed in the following section.

5. Discussion

The number of unique venue categories (327) was surprisingly large, for a small city state like Singapore. It could be that some of the venues were classified as different categories by different users (for example, Toast box could be classified as a breakfast place or a café. It is interesting to note that the clustering results found some clusters which appeared to follow certain sections of the MRT line (see below SMRT map).



For instance, cluster 4 was dominated by stations at the central town regions around Singapore river such as Outram, Raffles Place, Chinatown and Clarke Quay, as well as a section of the downtown line from

Stevens station all the way towards Bukit Panjang. Similarly, cluster 1 appeared to trace the north-south line including Choa Chu Kang, Yew Tee, Admiralty to Ang Mo Kio. It is possible that these sections of the MRT network, being built around the same timeframe or surrounded by housing estates of a certain age, had similar characteristics or amenities.

It was surprising that Jalan Besar had 21 indian restaurants, since it is located away from the Little India district, showing how the siting of restaurants may evolve organically, and differ from common expectation.

Workers in the downtown station area could be of an expatriate profile, since there are four Starbucks in that locale alone. A surprising number justifiable only by having copious numbers of coffee drinkers.

The unexpectedly small number of ATMs returned could be the result of several factors. First, ATM being a ubiquitous amenity, serves a rather mundane, though necessary function. Hence, Foursquare users are unlikely to check in and write a review since it is neither cool nor hip. The ATM at Commonwealth station could have been checked in because it is difficult to find one in that station, hence users are motivated to leave a tip for future visitors. Another reason could be that ATMs located in malls around MRTs are so common that people would not think of checking in at that location on Foursquare.

6. Conclusion

This analysis shows both the power and limitations of using Foursquare API location data to analyse venues around a location. While it can lead to interesting insights, as the correlation between certain sections of the MRT line with different clusters and the unexpectedly high concentration of Indian restaurants around Jalan Besar; for mundane utility-type venues such as ATMs, Foursquare location data may not be truly representative. This is due to the nature of Foursquare data, which is dependent on user check-ins. User check-ins are in general biased towards restaurants and hip cafes, that would be cool to share with their friends, not everyday venues such as ATMs, post offices or banks. For the latter type of venues, it would be better to use location data from the bank or post offices directly, since that would capture all the available locations for that facility. Nevertheless, Foursquare location data remains a powerful and invaluable tool to shed new insights on data and is likely to increase in importance, given an increasingly socially connected world.

References:

1. https://en.wikipedia.org/wiki/Mass_Rapid_Transit_(Singapore)
2. https://en.wikipedia.org/wiki/Light_Rail_Transit_(Singapore)
3. https://www.kaggle.com/yxlee245/singapore-train-station-coordinates
4. https://www.findlatitudeandlongitude.com/l/Admiralty+mrt/1214628/)
5. Foursquare API
6. SMRT Network Map