TORONTO AND VANCOUVER PROJECT

CONTENTS

- Introduction
- Data
- Methodology
- Result
- Discussion
- Conclusion

TORONTO AND VANCOUVER PROJECT

Introduction/Business Problem

One friend currently lives in Toronto intends to move to Vancouver with an offer of new job. He is very happy with the current place he lives now - postcode area M6G (neighbourhood Christine) in Downtown Toronto, with many grocery stores, parks, and coffee shops around.

He wants to know in Vancouver what are the post code areas have similar surroundings, so that he will be able to choose one of these post code areas where he can live with the similar comforts as he has in Downtown Toronto M6G, and at the same time, among those choices, to pick one post code area near to his new workplace.

In modern society, quite often people need to move from one place to another, could be to different cities, or different states or even different countries where they may not be familiar with. If they can see the comparability of the old place and potential new places in terms of surrounding venues, it will be very helpful for them to find the suitable new place similar to their current living place, so that they will be more comfortable to move.

Data

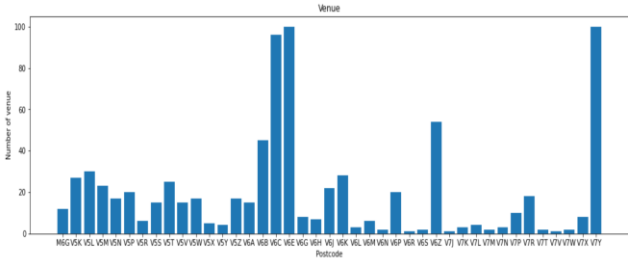
Data Source

1. Postcode Areas – https://en.wikipedia.org/.
2. Latitude and Longitude – pgeocode
3. Surrounding Venue and Venue Category - ***Foursquare*** database

	Postcode	Postcode Latitude	Postcode Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	M6G	43.6683	-79.4205	Fiesta Farms	43.668471	-79.420485	Grocery Store
1	M6G	43.6683	-79.4205	Scout and Cash Caffe	43.667360	-79.419938	Café
2	M6G	43.6683	-79.4205	Contra Cafe	43.669107	-79.426105	Café
3	M6G	43.6683	-79.4205	Christie Pits Park	43.664177	-79.420466	Park
4	M6G	43.6683	-79.4205	Karma Co-operative	43.668185	-79.414504	Grocery Store

Data Processing

1. BeautifulSoup to extract table
2. Visualization to review the venue data from Foursquare

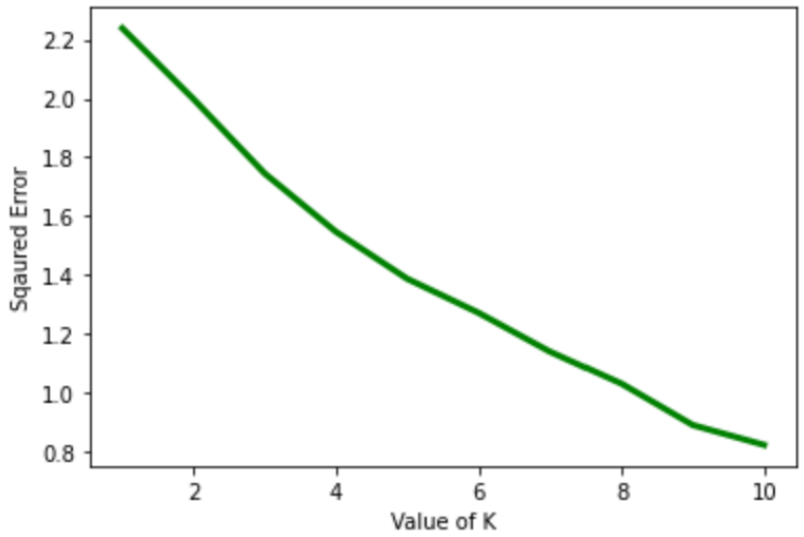


3. Use group by and means to get frequency of venue occurrence for each postcode area

Clustering - Unsupervised Leaning

- Kmeans Clustering
- Hierarchical Clustering
- Dbscan Clustering

Methodology




```
# run k-means clustering
kmeans = KMeans(n_clusters=6, random_state=0).fit(tandvclustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:33]
```

```
array([4, 0, 3, 3, 0, 3, 2, 1, 3, 4, 1, 5, 4, 3, 3, 3, 3, 0, 0, 3, 3, 1,
       1, 3, 4, 0, 0, 3])
```

Elbow method is a popular method to determine optimal number of K. It is just to perform K-means clustering with all these different values of K and for each K, we calculate average distances to the centroid across all data points, as K increases, less points in cluster, average distance to centroid decreases.

Kmeans Clustering

- Efficient
- Needs to define number of K
- Tries to minimize the sum of distance in each cluster to its central (Euclidean distance)
- Each run of Kmeans may have different results.
- Assign all data points to one cluster even they do not belong to any.

```
#let's try another hierarchical
from sklearn.cluster import AgglomerativeClustering
```

```
Hierarchical= AgglomerativeClustering(n_clusters=6, affinity='euclidean')
Hierarchical.fit(tandvclustering)
```

```
AgglomerativeClustering(n_clusters=6)
```

```
Hierarchical.labels_
```

```
array([1, 0, 0, 0, 0, 0, 5, 0, 0, 1, 0, 3, 1, 0, 0, 0, 0, 4, 0, 0, 0, 2,
       0, 0, 1, 4, 0, 0], dtype=int64)
```

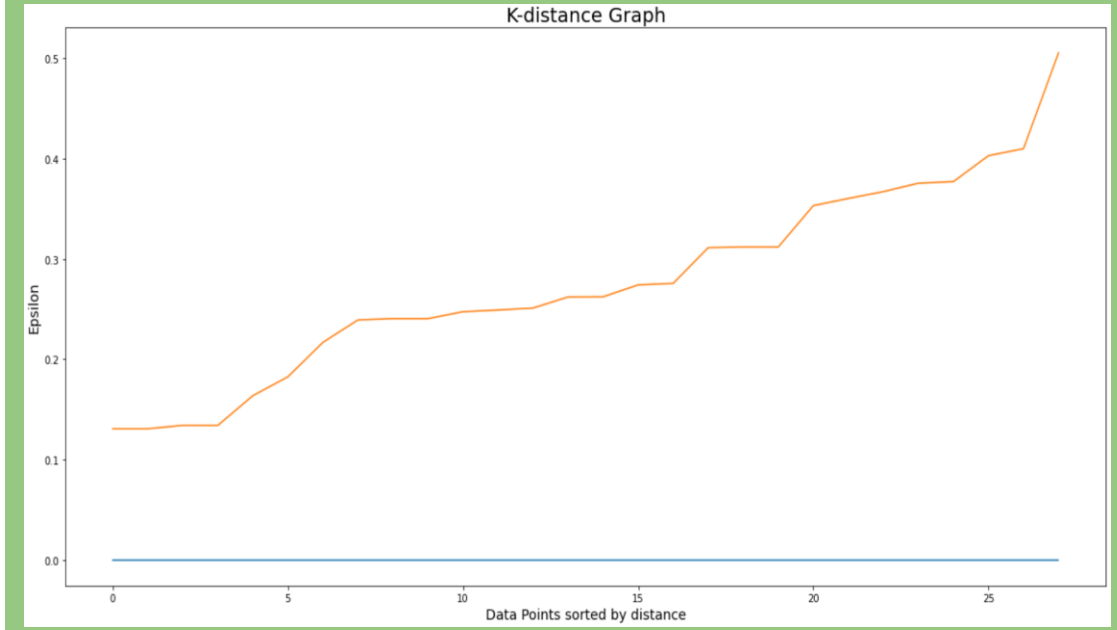
Agglomerative algorithm begins with ‘n’ clusters, and it merges the closet datapoints and keeps doing so untill the only single cluster remains.

Different criteria of distance between clusters. Here we use Euclidean distance.

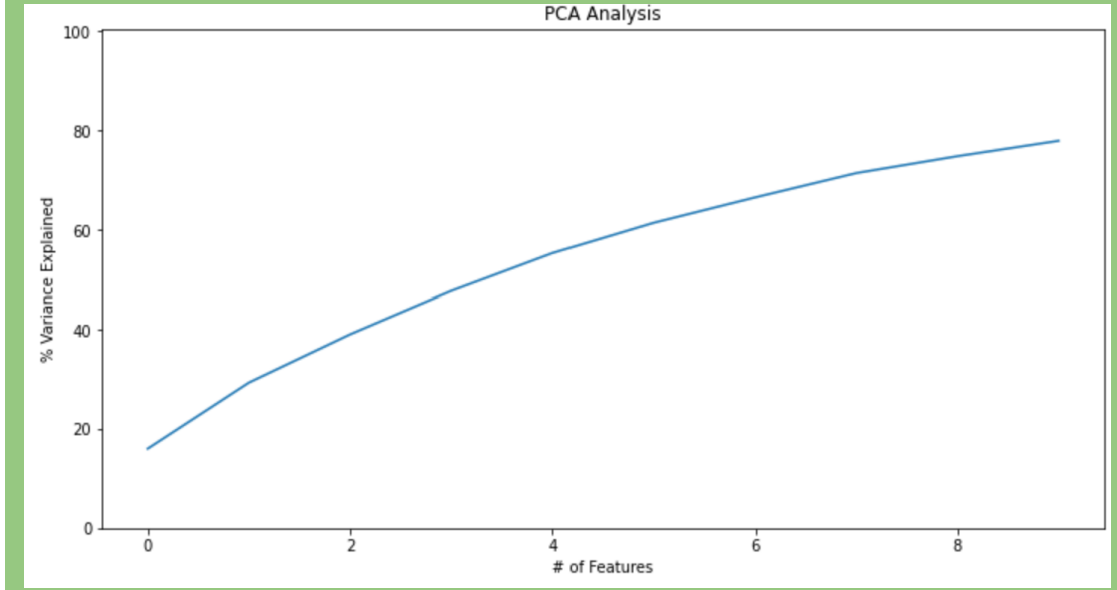
Default linkage is "ward" which minimizes the variance of the clusters being merged.

Hierarchical Clustering

- Does not require pre-defined number of clusters.
- Always generate the same clustering result.
- Less efficient, not suitable for large dataset.



K-distance to find Epsilon (elbow method)



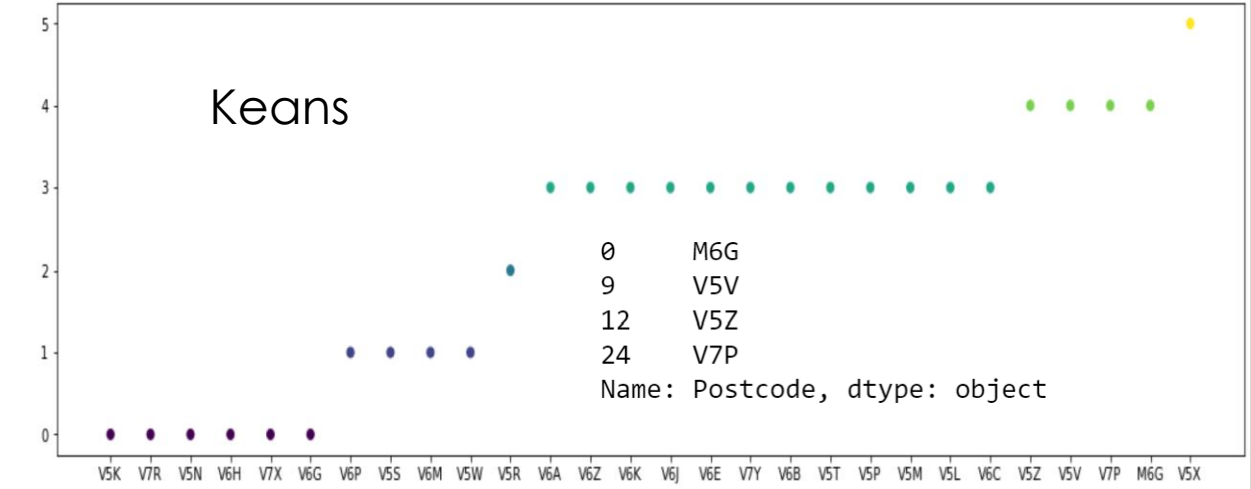
PCA to reduce dimension
- from 183 features to 8 features

Dbscan Clustering

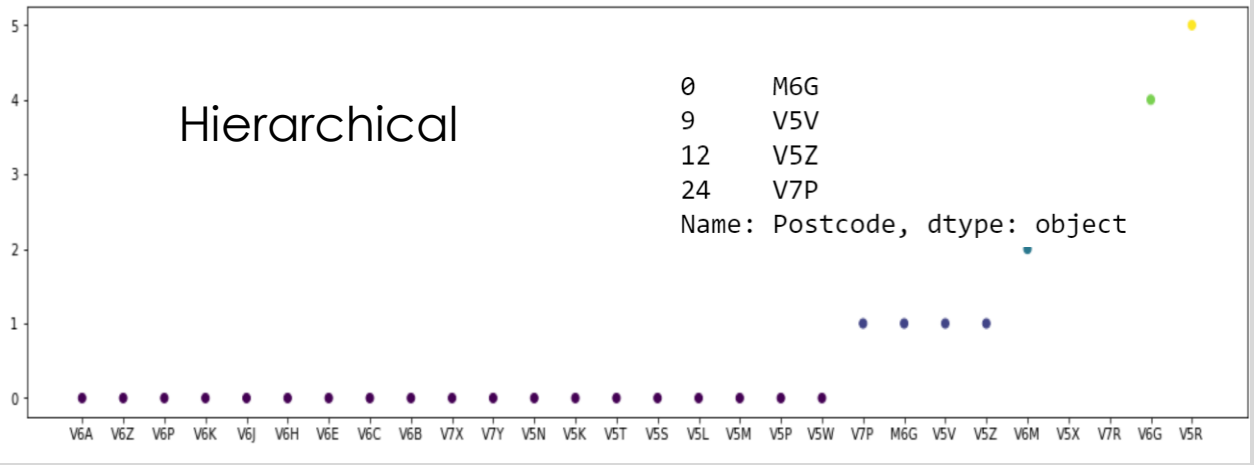
- Separates high density area from low density area
- Density here is defined by mainly two parameters, Epsilon and Minimum points
- Robust to find outliers

Results

Keans



Hierarchical



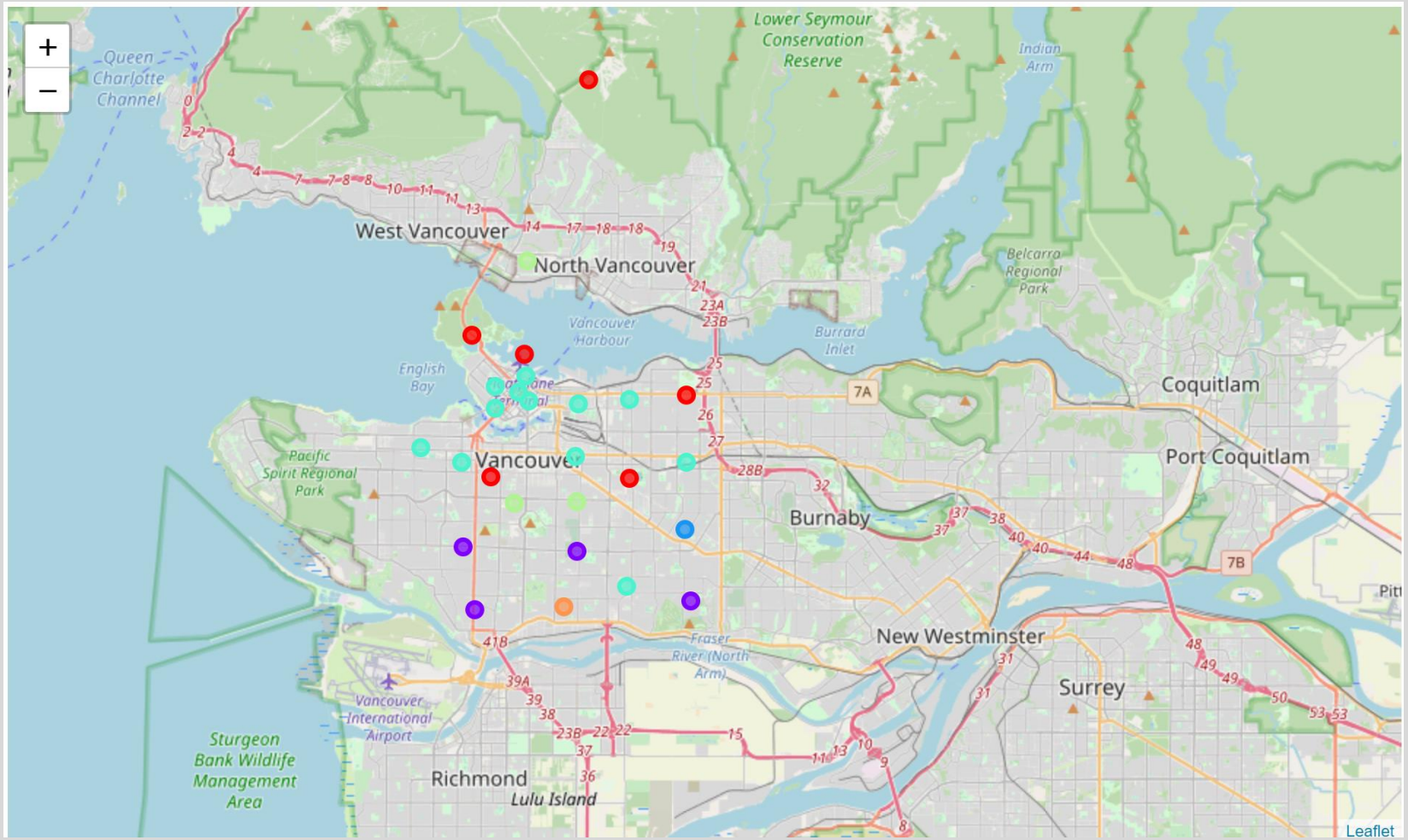

```
n_clusters_ = len(set(dbscan_opt.labels_)) - (1 if -1 in dbscan_opt.labels_ else 0)
n_noise_ = list(dbscan_opt.labels_).count(-1)
print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
```

```
Estimated number of clusters: 1
Estimated number of noise points: 3
```

Dbscan

Results

Map based on hierarchical



Discussion

Kmeans and Hierarchical clustering generate similar results to find the similar postcodes area in Vancouver as Toronto area M6G. Both of them work well for this particular datasets. But Kmeans and hierarchial clustering do not work well for non-globular or non-spherical structured clusters.

Dbscan here basically clusters majority of the postcodes area into one cluster and points out 3 outliers. Looks like Dbscan does not work well for high dimension data, but even we reduced the dimension, Dbscan still does not cluster properly, that is due to Dbscan has difficulty to cluster with similar densities.

Interestingly Dbscan is outliering V5R, V5X and V6M three areas as outliers (DBSCAN clustering is robust to outliers), and Hierachical and Kmeans clustering is actually put these three postcode areas as single culster for each, as these two algorithm assign all the data points to clusters.

CONCLUSION

Conclusion

Different clustering algorithm uses different methodology like Kmeans is a least-squares optimization, while DBSCAN finds density-connected regions. They all have pros and cons. Which algorithm to use depends on the nature, size and shapes of the datasets. Of course the domain knowledge also helps us to choose the right algorithm and choose the right parameters for those algorithms.