MSA Project Week – Spring 2022

Analysis of FBC and Network Hierarchy Data

Team – Bhediyon ka Jhund

Akshay Jadiya

Saurabh Aggarwal

Yashwant Singh

Executive Summary

Problem Statement

Cox Communications wants to utilize FBC data to its full potential in order to reduce operational costs

Key Achievements

1. Clustered FBC data for 40 nodes and identified major impairments types
2. Visualized most common ancestors of *address* network elements
3. Compressed FBC data for faster access and efficient storage in Hive

Business Impact

1. Quicker impairment identification and resolution
2. Faster turnaround times on customer problems
3. Reduced Operational Cost
4. Efficient utilization technicians' time

Clustering Methodology

Extract relevant features that capture characteristics of FBC string using ***tsfresh***

Feature Extraction

Further reduce dimensions using **UMAP** to boost clustering performance

Dimensionality Reduction

Use **HDBSCAN** to find clusters in the FBC data to identify common impairment patterns

Density Based Clustering

Visually inspect different clusters and identify commonly occuring impairments

Impairment Identification

Clustering Results

Reasons for algorithm choices

Why UMAP?

- No linearity assumption (unlike PCA)
- Faster than other similar techniques
- Better ability to capture global structure

Why Density Based Clustering?

- Can identify non-spherical clusters
- Outlier independent

Why Density Based Cluster Validation?

- Most suitable for arbitrarily shaped clusters
- Capable of handling noise objects

Feature Reduction

8,704

Original FBC vector

782

After feature extraction

5

After dimensionality reduction

13 clusters were obtained for each node. Below figures illustrate 3 impairments

DBCV Score (HDBSCAN) : 0.73

DBCV Score (KMeans) : 0.08

Fig 1 – Standing Waves

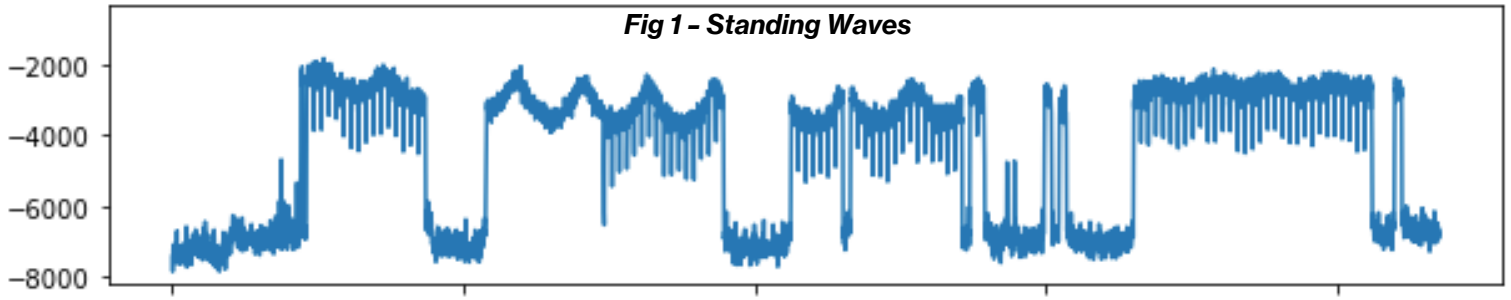


Fig 2– Negative Tilt

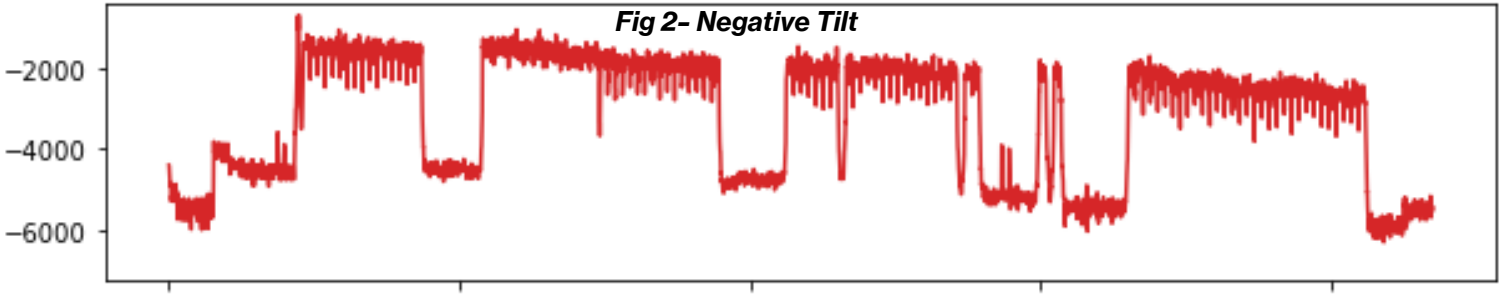
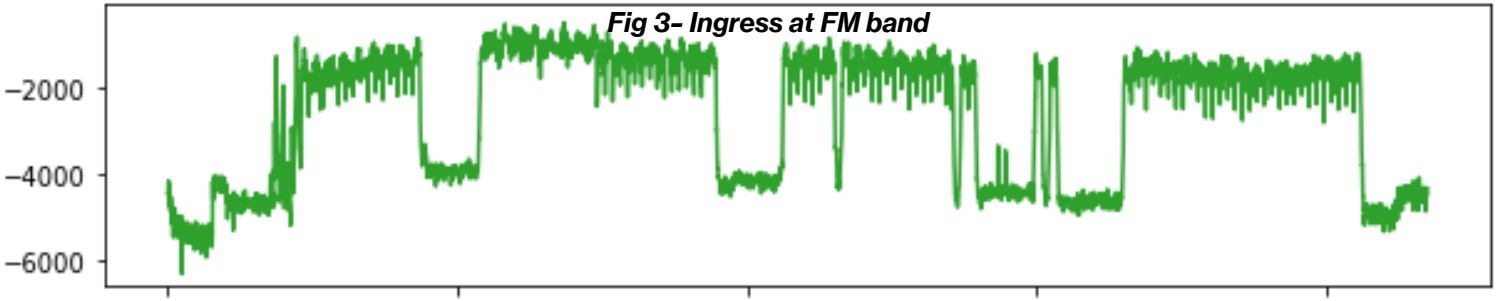


Fig 3– Ingress at FM band



Hierarchy of Devices and Visualization

Objective

Use network parent to child relationships, discover a way to algorithmically find the most common ancestor of any collection of *address* network elements and visualize this.

Data Structure Used

- Network graph created using ***NetworkX***
- Ancestor list for all nodes and stored it in a dictionary

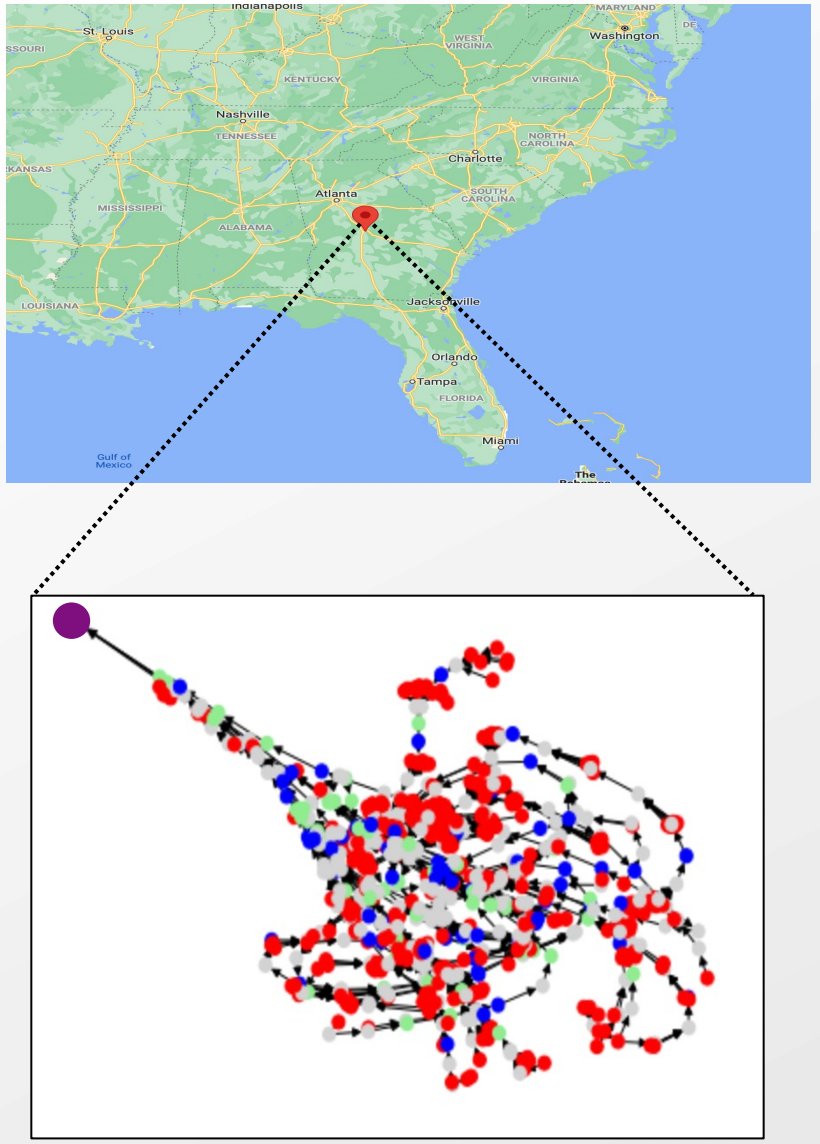
Algorithm

- Common ancestor is found using set intersection on ancestor lists

Why is our approach faster?

- Database structure allows constant time access to ancestor lists
- Traditional LCA algorithms traverse the whole graph. In contrast, our approach traverses only a fraction of the dataset
- We have reduced the time complexity by a factor of 10^4 at the expense of increasing space complexity

Location of the node MM106 and the hierarchical structure of the devices in this node

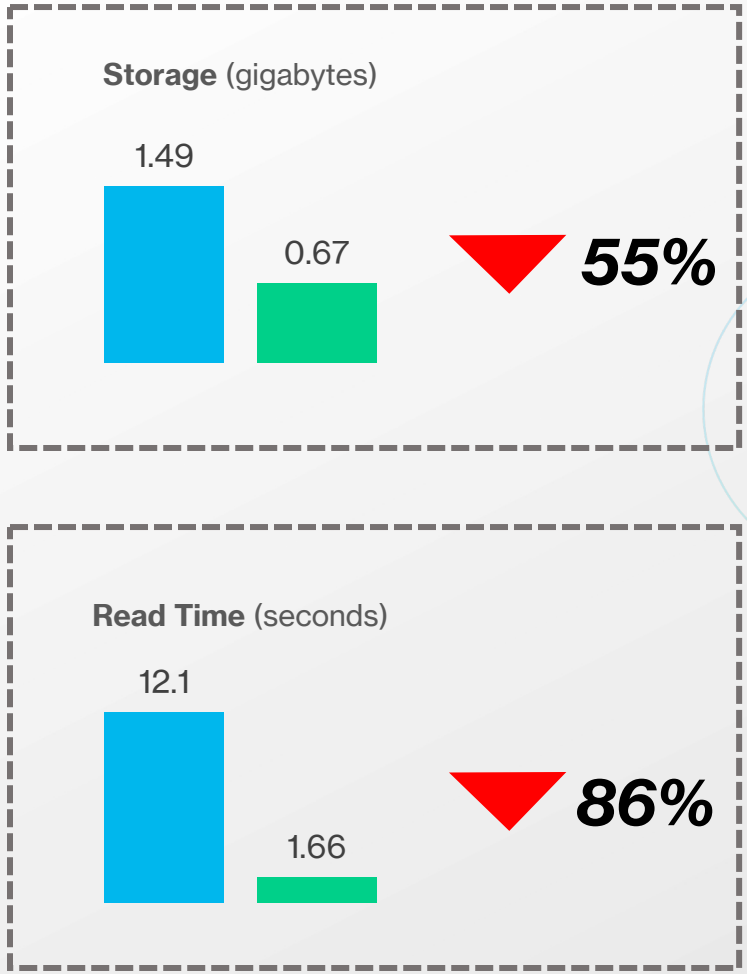


Data Compression

What is parquet file format?

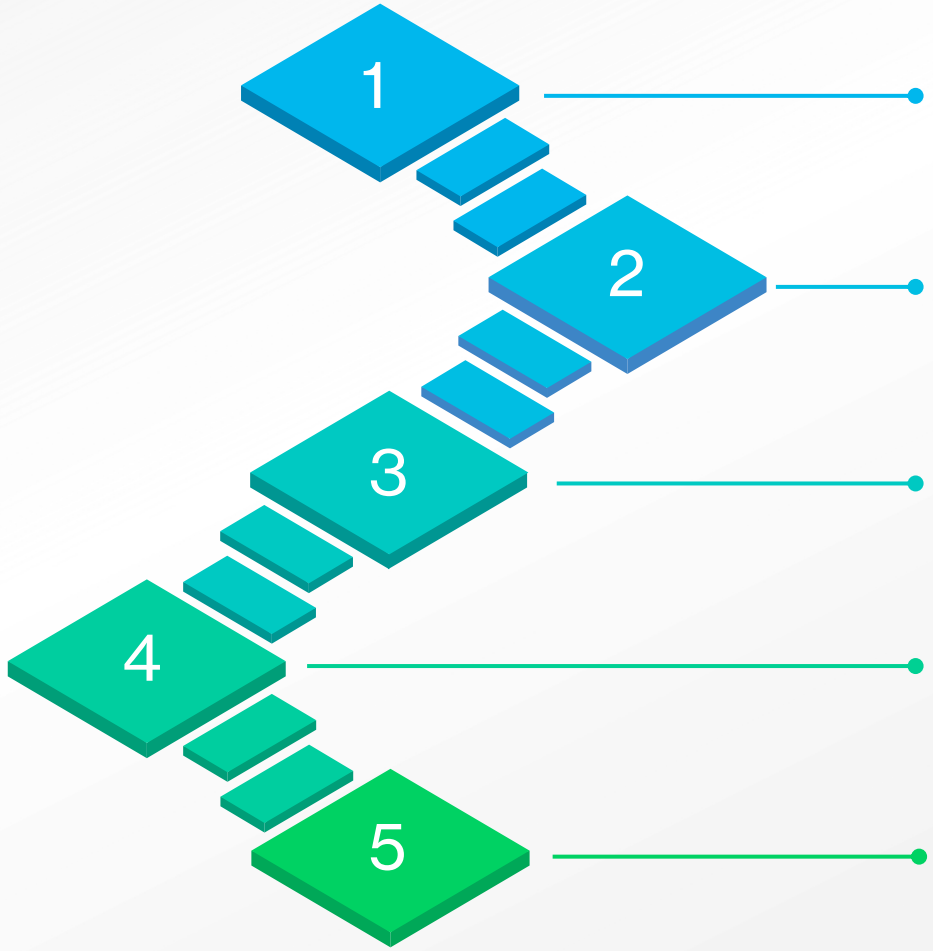
1. Open source file format available to any project in the Hadoop ecosystem
2. Optimized to work with complex data in bulk and features different ways for efficient data compression and encoding types
3. Columnar storage which is designed to bring efficiency compared to row-based files like CSV
4. Parquet is built to support flexible compression options and efficient encoding schemes

Below statistics are for 40 node files in the given dataset



Using .parquet file format instead of .csv can reduce the storage by **55%** and file read time by **86%**

Future Work



1. Based on the domain knowledge, make custom features for the FBC data. ***tsfresh*** provides the flexibility to define custom features
2. Use clustering results to facilitate labeling of the impairments. Based on the labels, train a CV model to classify impairments based on input FBC vector / image
3. A classification problem would allow us to use the ***tsfresh*** package for feature selection by identifying all strongly and weakly relevant attributes based on hypothesis tests
4. Create a dynamic visualization of address hierarchy with “zoom-in” and “zoom-out” features, showing labels for each device type
5. Convert the FBC vector to time domain using Inverse Fourier Transform and then perform the proposed steps and compare results

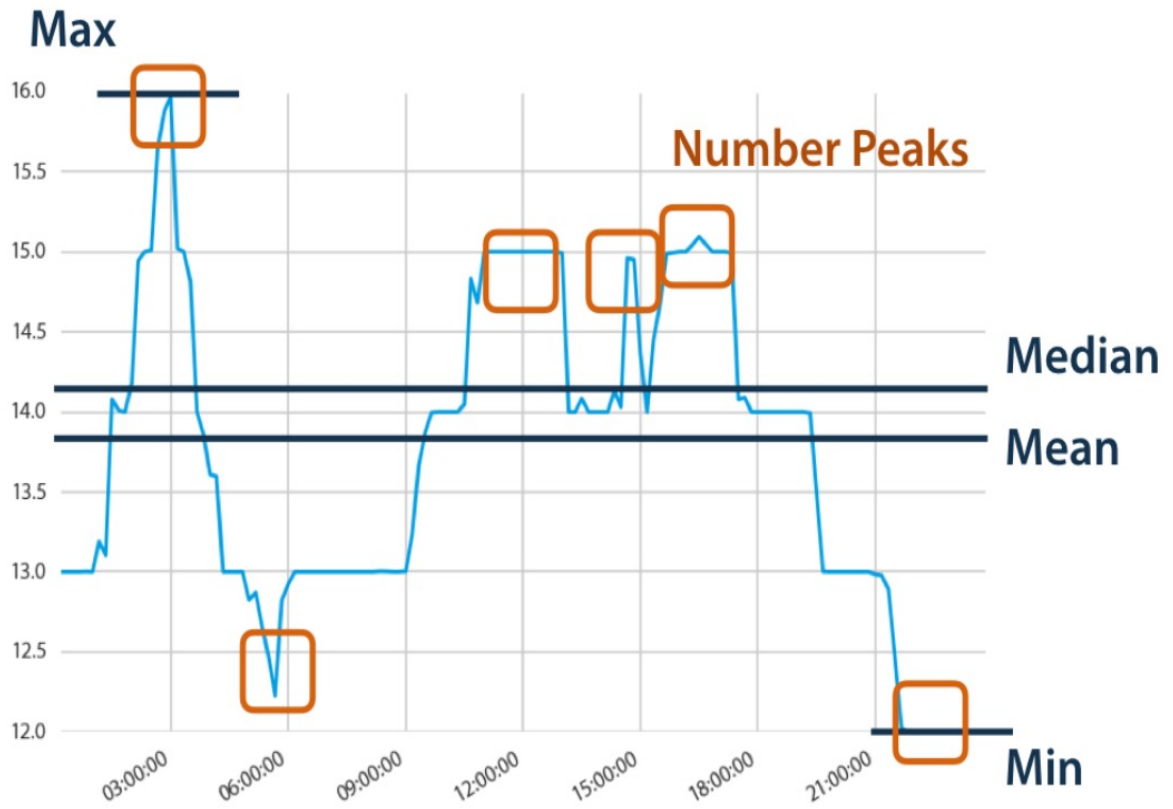
Questions?

Appendix

Feature Extraction for Sequential Data - *tsfresh*

Extracting features from a sequential data

- Feature extraction from a sequential data plays a major role during the early phases of data science projects in order to rapidly extract and explore different time series features
- In order to use a set of sequential data as input for supervised or unsupervised ML algorithms, the sequential data needs to be mapped into a well-defined feature space with problem specific dimensionality
- One might decide to map the sequential data into a design matrix of *N(# time series)* rows and *M* columns by choosing *M* data points.
- However, from the perspective of pattern recognition, it is much more efficient and effective to characterize the sequential data with respect to the distribution of data points, correlation properties, stationarity, entropy, and nonlinear analysis.



A few examples of the type of features extracted - source

References

1. Original tsfresh paper
2. tsfresh GitHub
3. tsfresh documentation

UMAP – Fast and Efficient Dimensionality Reduction

UMAP - Uniform Manifold Approximation and Projection

- Most dimensionality reduction algorithms fit into either one of two broad categories:
  1. Matrix factorization (such as PCA)
  2. Graph layout (such as t-SNE)

- UMAP is a graph layout algorithm, very similar to t-SNE, but with a number of key theoretical underpinnings that give the algorithm a more solid footing

- Two steps in UMAP algorithms:
  1. Construction of a graph in high dimensions with edge strength representing how “close” a given point is to another
  2. Optimization to find the most similar graph in lower dimensions

- To determine connectedness, UMAP extends a radius outwards from each point, connecting points when those radii overlap

- Choosing this radius is critical
  1. Too small → small, isolated clusters
  2. Too large → will connect everything together

- UMAP overcomes this challenge by choosing a radius locally, based on the distance to each point's *n*th nearest neighbor. UMAP then makes the graph "fuzzy" by decreasing the likelihood of connection as the radius grows. Finally, by stipulating that each point must be connected to at least its closest neighbor, UMAP ensures that local structure is preserved in balance with global structure.

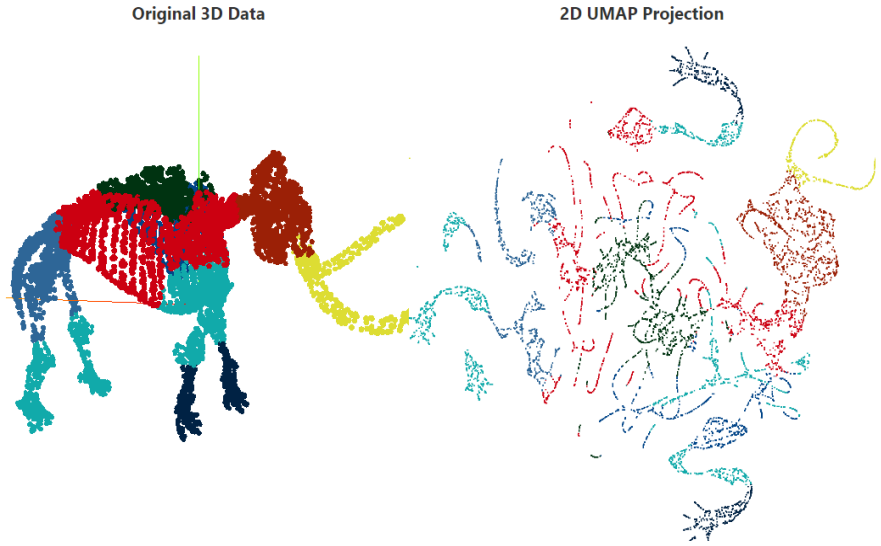


Illustration of original vs reduced data using UMAP - source

References

1. How UMAP works with illustrations
2. Math behind UMAP
3. UMAP documentation with hyperparameter details
4. Original UMAP paper

HDBSCAN vs. KMeans

- K-means performs best when clusters are:
  1. “round” or spherical
  2. equally sized
  3. equally dense
  4. most dense in the center of the sphere
  5. not contaminated by noise/outliers

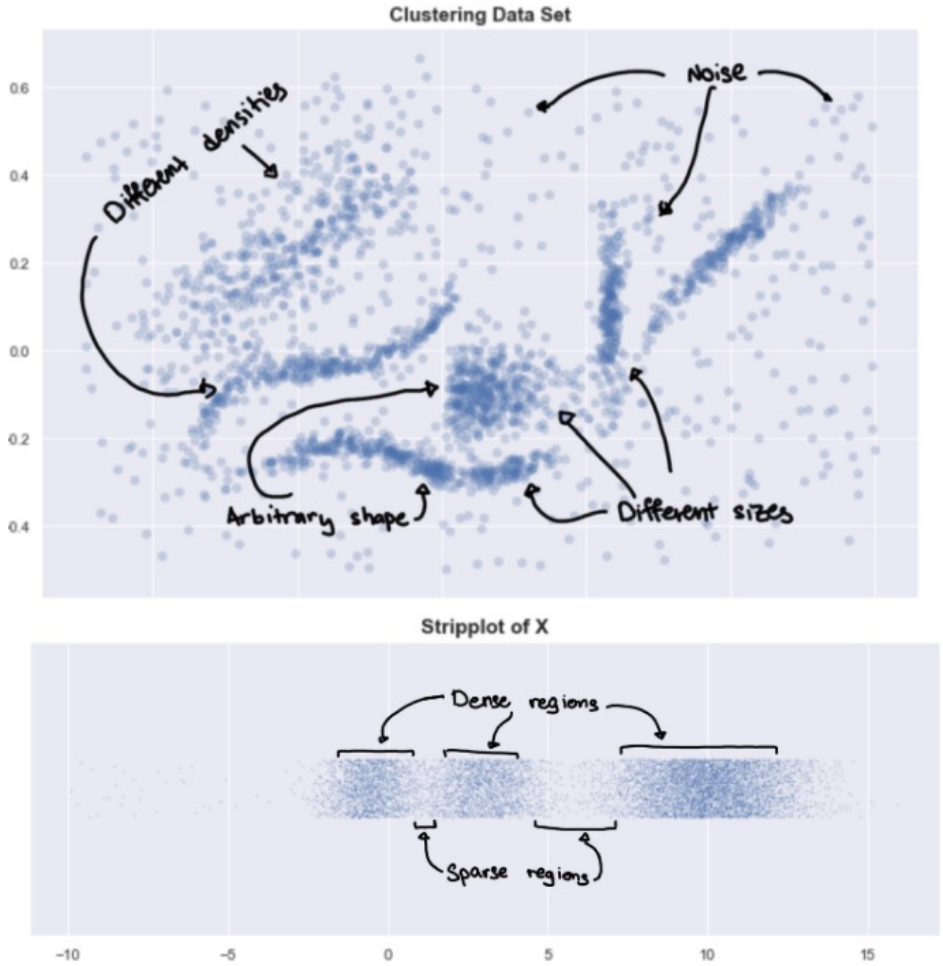
- HDBSCAN uses a density-based approach which makes few implicit assumptions about the clusters.

- Rather than looking for clusters with a particular shape, it looks for regions of the data that are denser than the surrounding space. The mental image you can use is trying to separate the islands from the sea or mountains from its valleys.

- Self-adjusting (HDBSCAN) is the most data-driven of the clustering methods, and thus requires the least user input.

- For Self-adjusting (HDBSCAN), the reachability distances can be thought of as nested levels of clusters. Each level of clustering would result in a different set of clusters being detected.

- Self-adjusting (HDBSCAN) chooses which level of clusters within each series of nested clusters will optimally create the most stable clusters that incorporate as many members as possible without incorporating noise.



References

1. Working of HDBSCAN
2. HDBSCAN hyperparameters
3. Documentation

Density Based Clustering Validation

Silhouette Scores vs DBCV

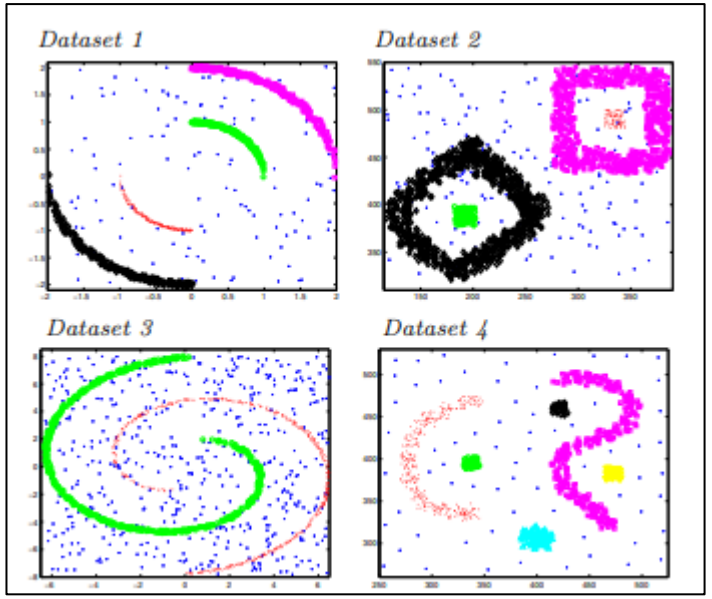
- Density-based clustering algorithms seek partitions with high density areas of points (clusters, not necessarily globular) separated by low density areas, possibly containing noise objects. In these cases relative validity indices proposed for globular cluster validation may fail.

- Unlike other relative validity indices, DBCV has two major benefits -
  1. It directly takes into account density and shape properties of clusters
  2. Properly deals with noise objects, which are intrinsic to the definition of the density-based clustering

- Density-based clusters are defined as regions of high density separated from other such regions by regions of low density. Considering such a model we can expect a good density based clustering solution to have clusters in which the lowest density area inside each cluster is still denser than the highest density area surrounding clusters.

- DBCV computes the least dense region inside a cluster and the most dense region between the clusters, which are used to measure the within and between cluster density connectedness of clusters.

Comparing the performance of DBCV with other metrics on toy datasets - source

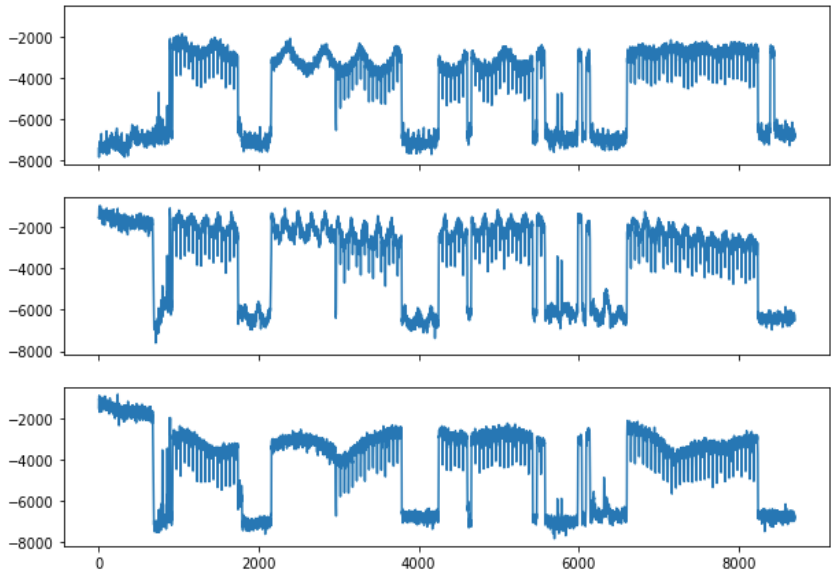


Index	Dataset			
	Dataset 1	Dataset 2	Dataset 3	Dataset 4
DBCV	**0.91**	**0.90**	**0.74**	**0.99**
SWC	0.72	0.21	0.19	0.31
VRC	0.51	0.01	0.02	0.01
Dunn	0.20	0.01	0.01	0.01
CDbw	0.84	0.71	0.04	0.92
MB	0.51	0.01	0.01	0.01

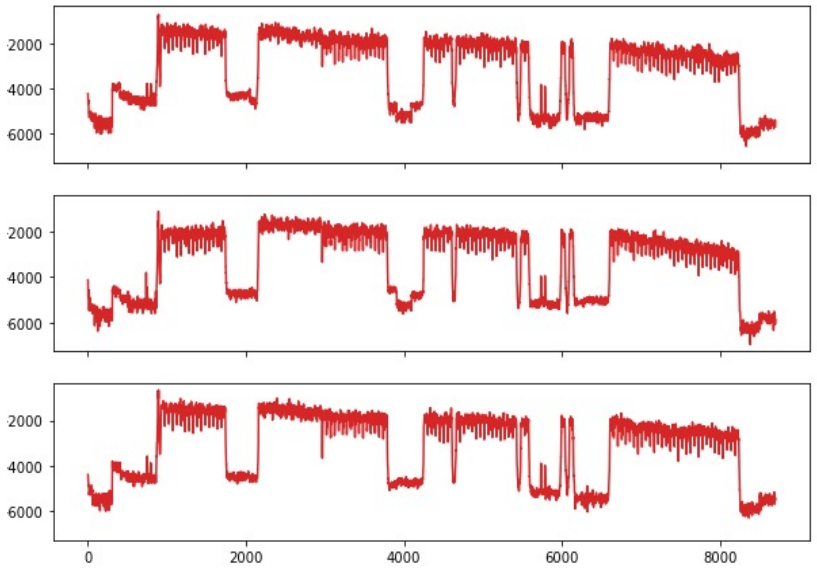
References

1. DBCV paper
2. DBCV GitHub
3. Using DBCV for clustering hyperparameter tuning

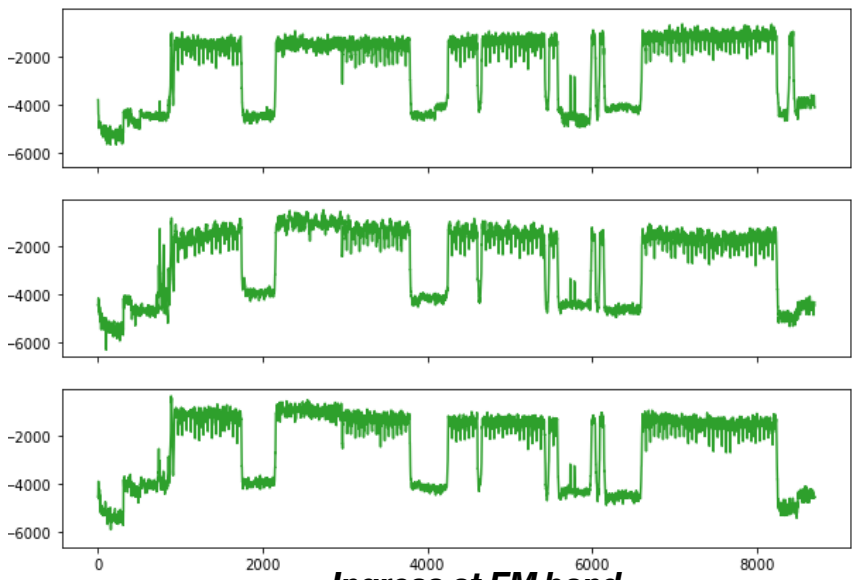
Other Clusters Identified



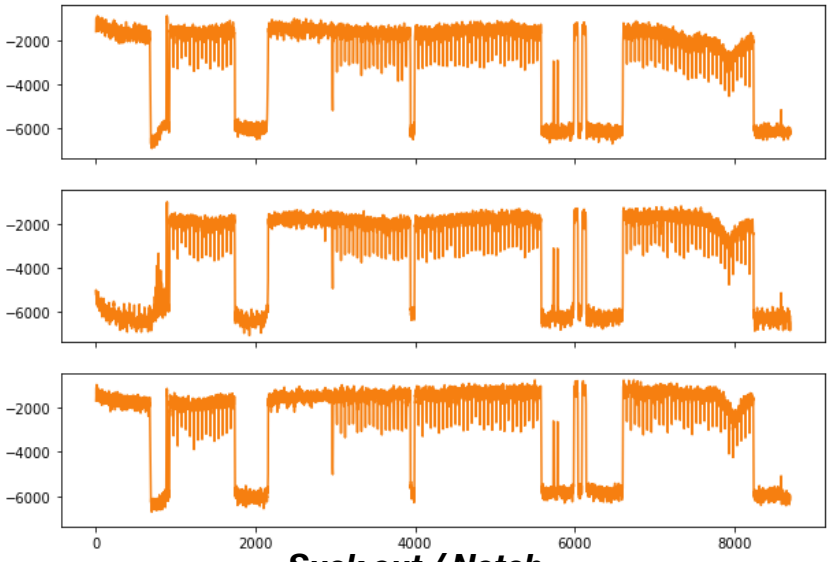
Standing Waves



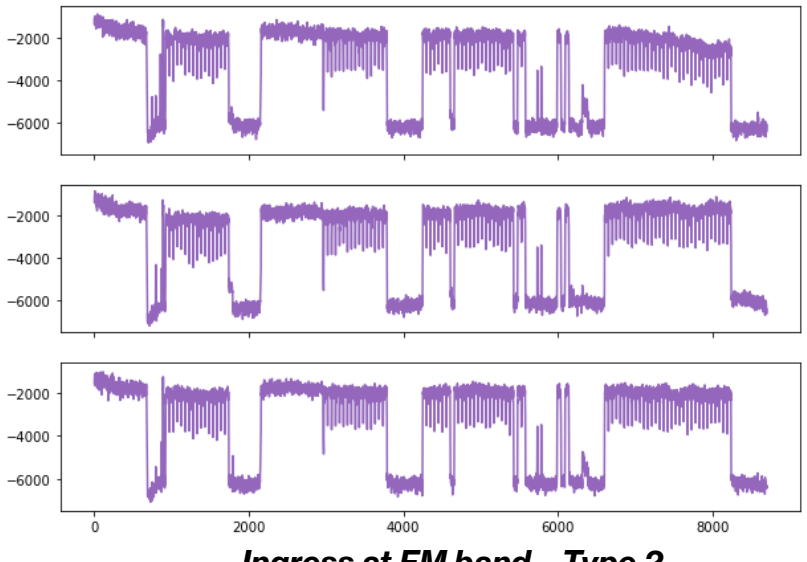
Negative Tilt



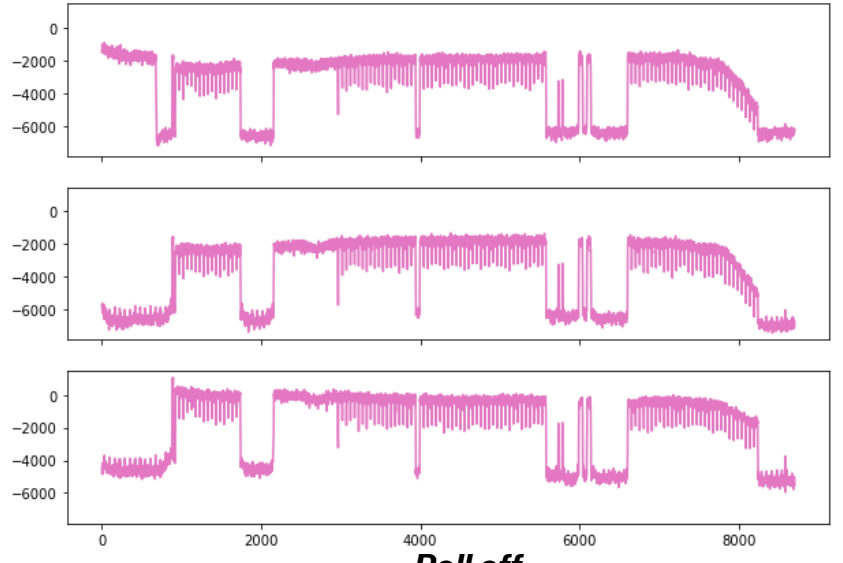
Ingress at FM band



Suck out / Notch



Ingress at FM band – Type 2



Roll off