KDetect: Unsupervised Anomaly Detection for Cloud Systems Based on Time Series Clustering

Swati Sharma, Amadou Diarra, Fredrico Alvares, Thomas Ropars

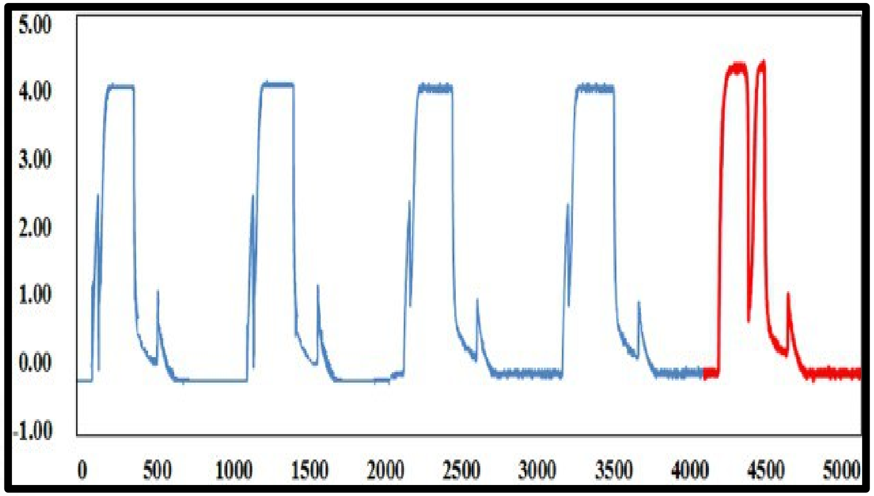
24-6-2020

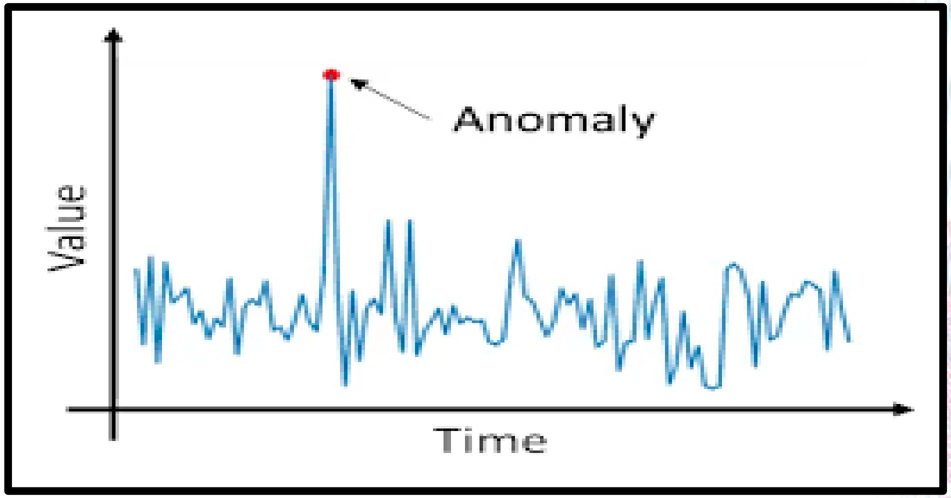
Context

- Cloud Computing runs large part of IT Infrastructure.
- Large number of Virtual Machines (VMs) – several thousands.
- Each executing services of unknown nature.
- Non-intrusive VM analysis by cloud provider.
- VMs typically monitored by resource consumption metrics.

Problem Domain

- Anomaly Detection – consequential for VM monitoring.
- Anomaly – unexpected system load/behavior based on collected system metrics.



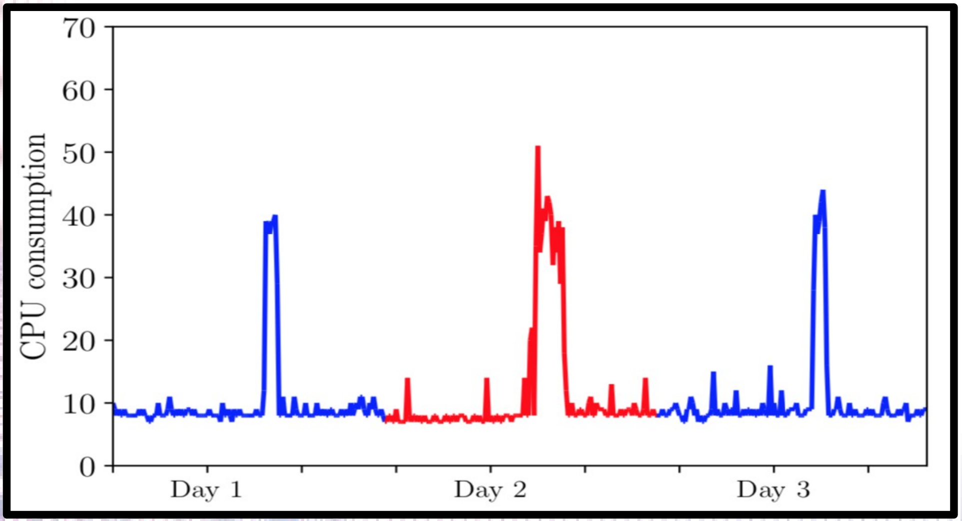


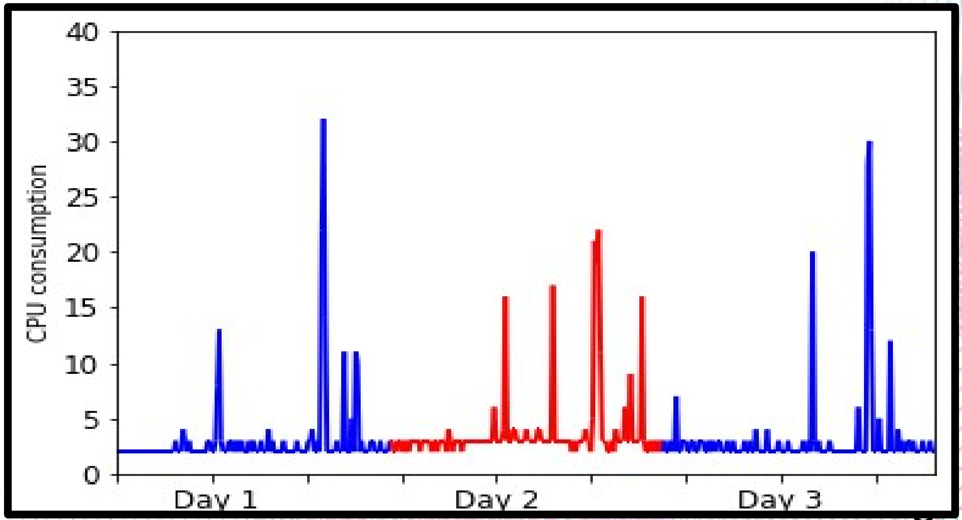
Objectives

- Generic solution to detect anomalies.
- Processing unlabelled time series.
- High accuracy (recall & precision) in anomaly detection.
- Quick Execution.

Challenges

- **Large Data Sizes -**
  - Execution Time per VM.
  - No labels available.
- **Data Content -**
  - Diverse normal & abnormal behavior.
  - Noise along with seasonal data.





Contributions

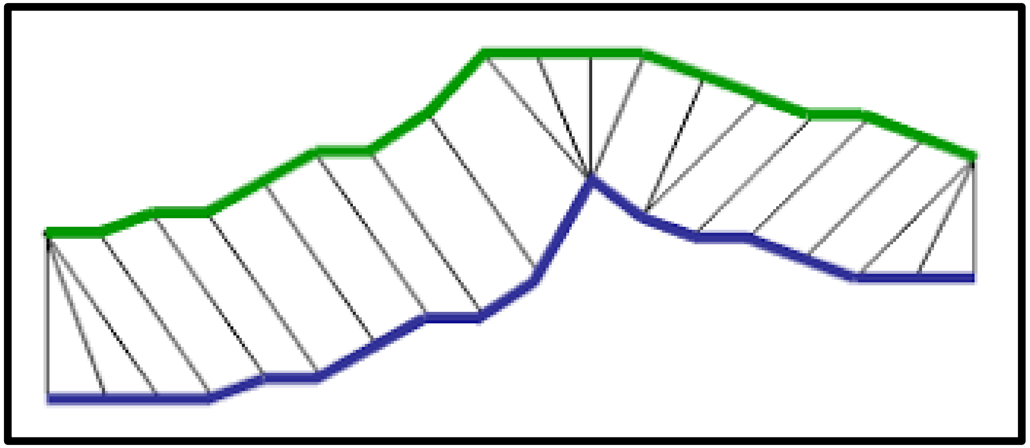
- **KDetect –**
  - Unsupervised learning technique to detect anomalies.
  - In time series exhibiting periodic behavior.
  - Dynamic Partitional Clustering Based Solution.
  - Generic heuristics without any configuration changes
- Evaluation done on production dataset from EasyVirt.
- Recall more than 94% & Precision more than 95%.
- Fast execution (330 days data analyzed in under 3 mins).

Related Work

- **Anomaly Detection in Cloud** -
    - **[Aggarwal2017]** Adaptive Real-Time - Analyze nodes running similar applications & predict next values to detect outliers.
    - **[Zhang2019]** Cross-Dataset Transfer Learning - Orthogonal to our solution. Transfer anomalies patterns from 1 cloud to next.
- **Unsupervised Anomaly Detection for Time Series -**
    - **[Xu2018]** Donut - State-of-the-art. Variational Auto-Encoder based.
    - **[Paparrizos2015]** k-Shape - Basic block of every KDetect iteration.

k-Shape

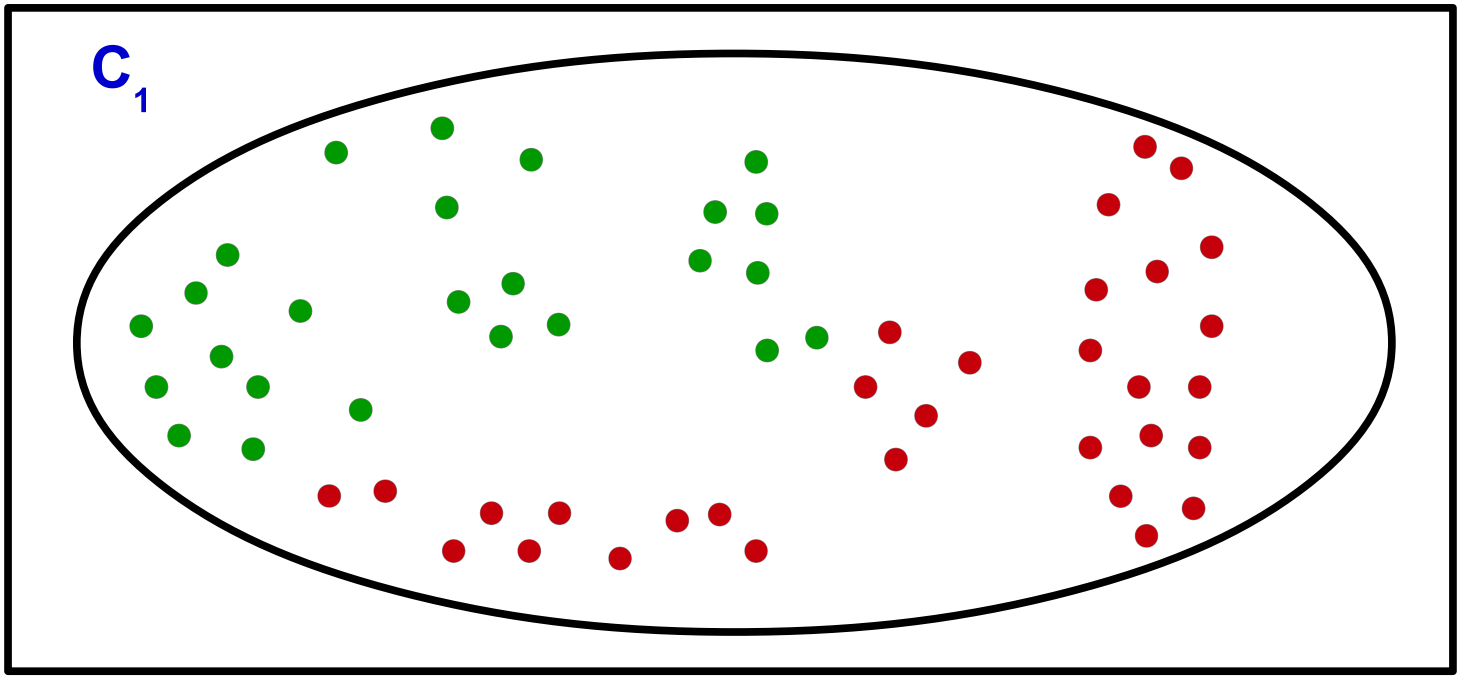
- Iterative Refinement Clustering algorithm.
- Uses Shape Based Distance (SBD) measure.
- Positioning in Euclidean Space - shape comparison.
- Number of clusters (k) required to be known in advance.



Solution: KDetect Algorithm

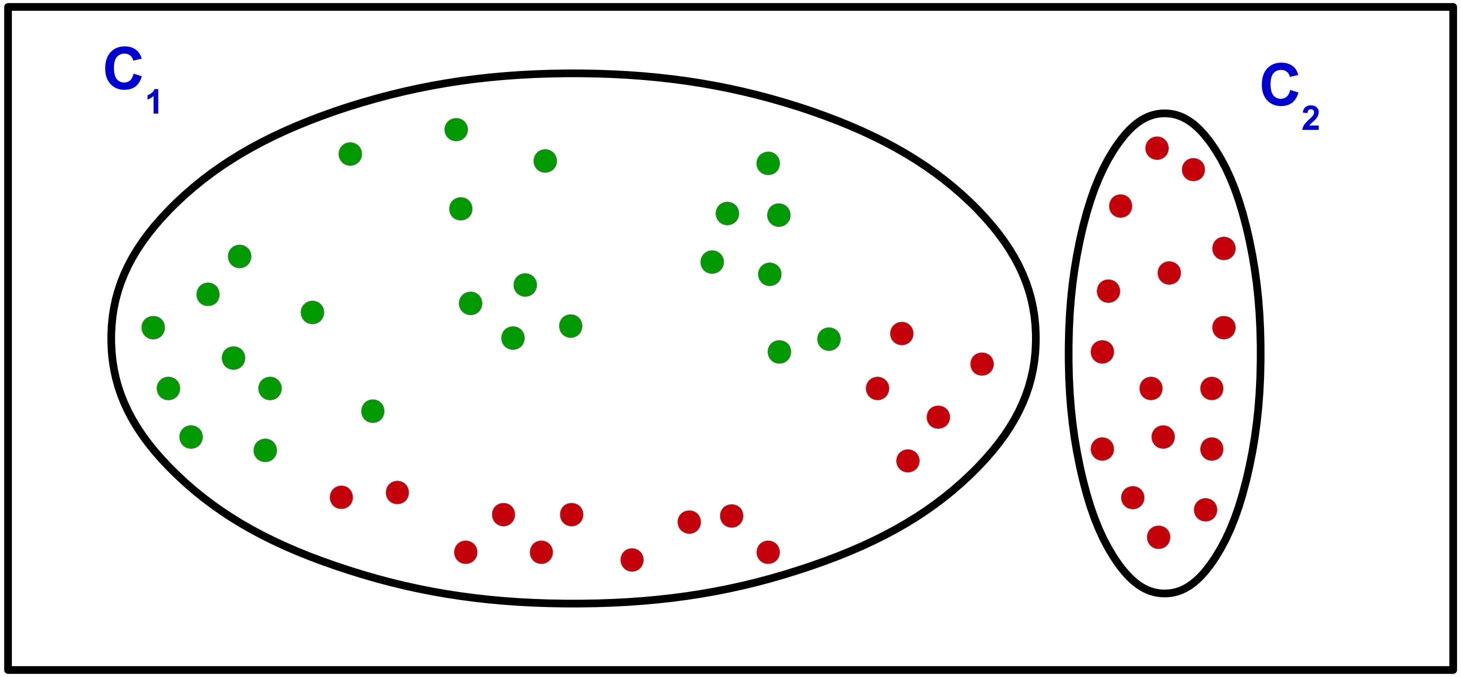
- Unsupervised Iterative Refinement Clustering algorithm.
- Progressively increase 'k' and cluster time series into normal & abnormal.
- Challenges -
    - Deciding what k gives good segregation?
    - How to label each cluster ('N/'Ab') at every iteration?
- Provides generic heuristics to solve these challenges without specific application to a particular VM.

KDetect



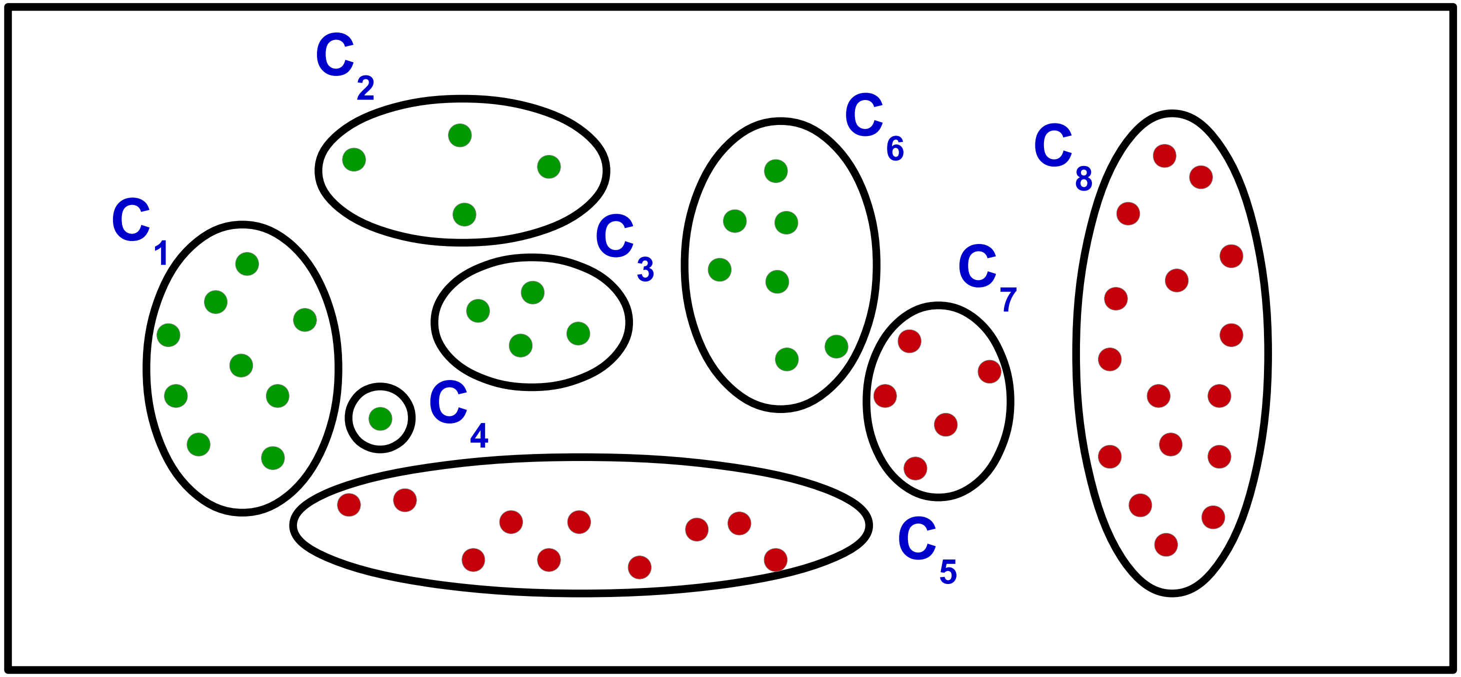
Initially : C_1 – Single cluster for all time series

KDetect



At k=2, Bigger cluster is assumed to be normal.

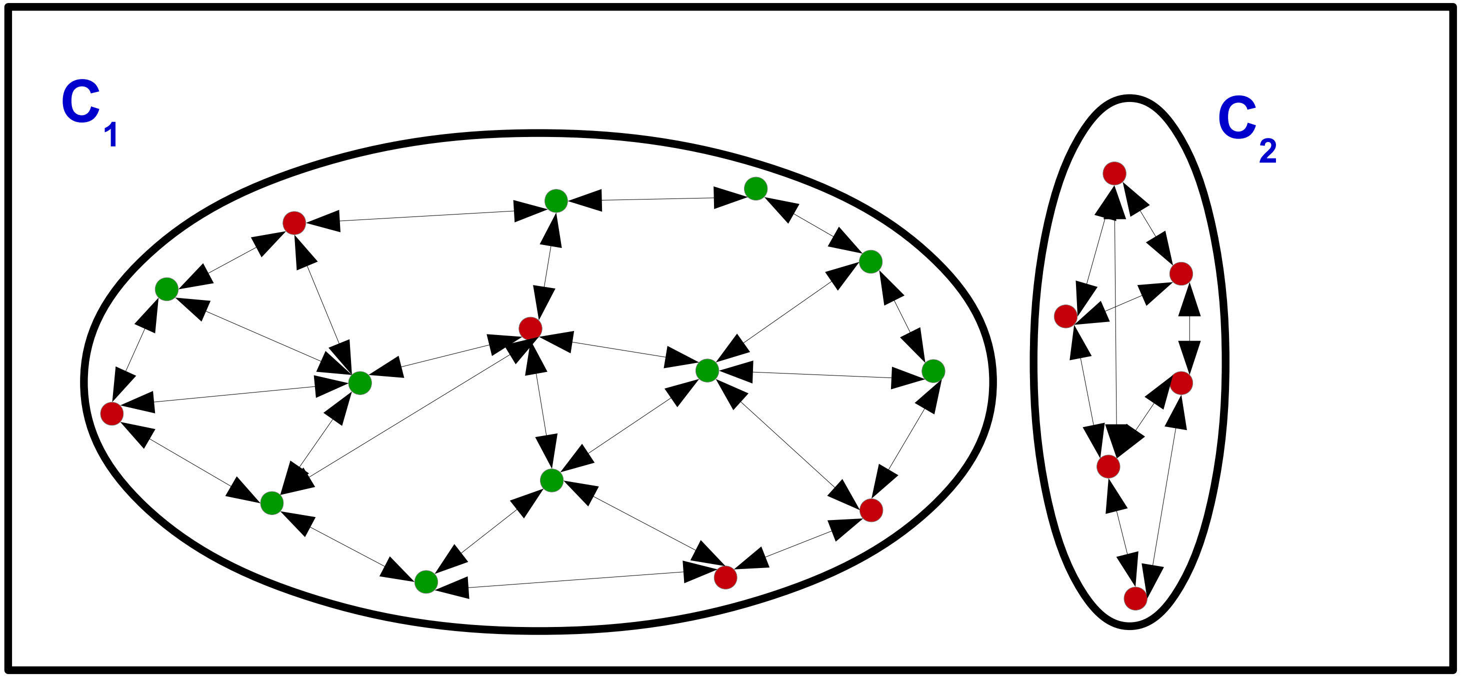
KDetect



At auto-halt iteration -

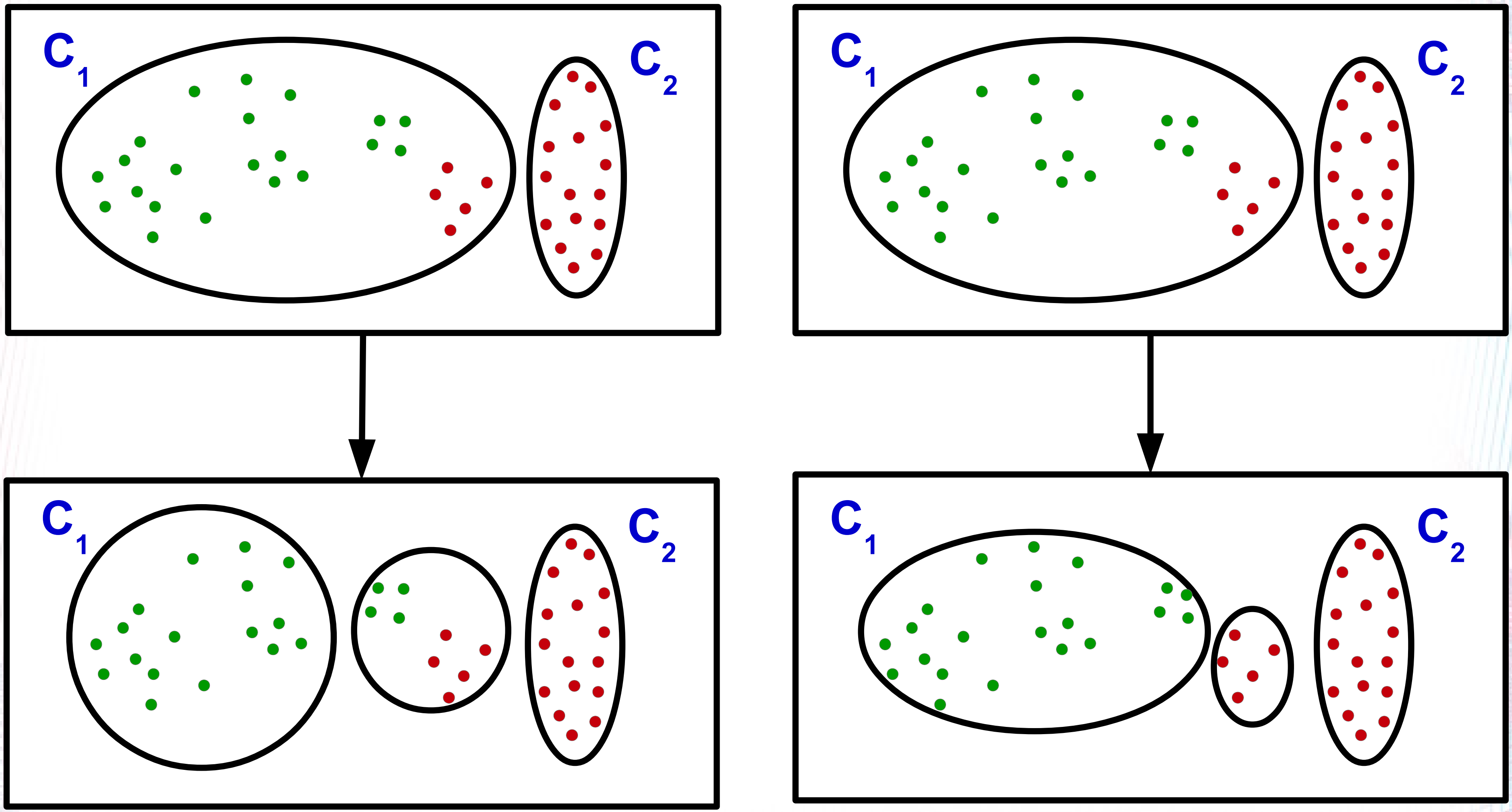
- Good segregation of normal & abnormal clusters.
- Clusters labelled 'N/Ab'.

Cluster Segregation Metrics : Density



Cluster Density - avg of distance (SBD) between any 2 time series (degree of similarity between time series).

Cluster Segregation Metrics : Density

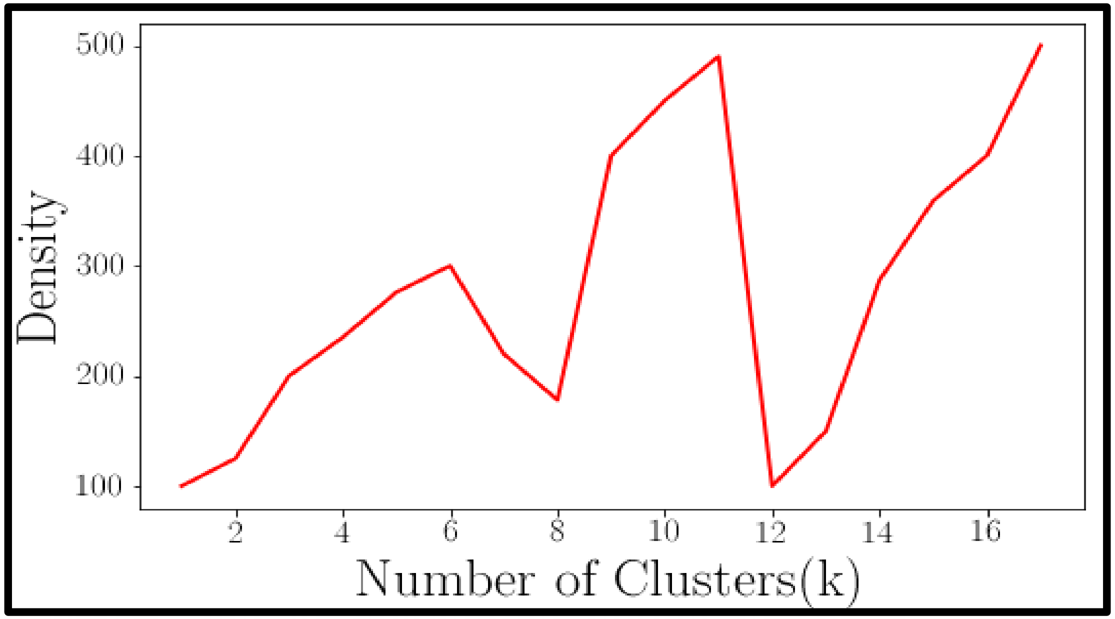


Density Decrease

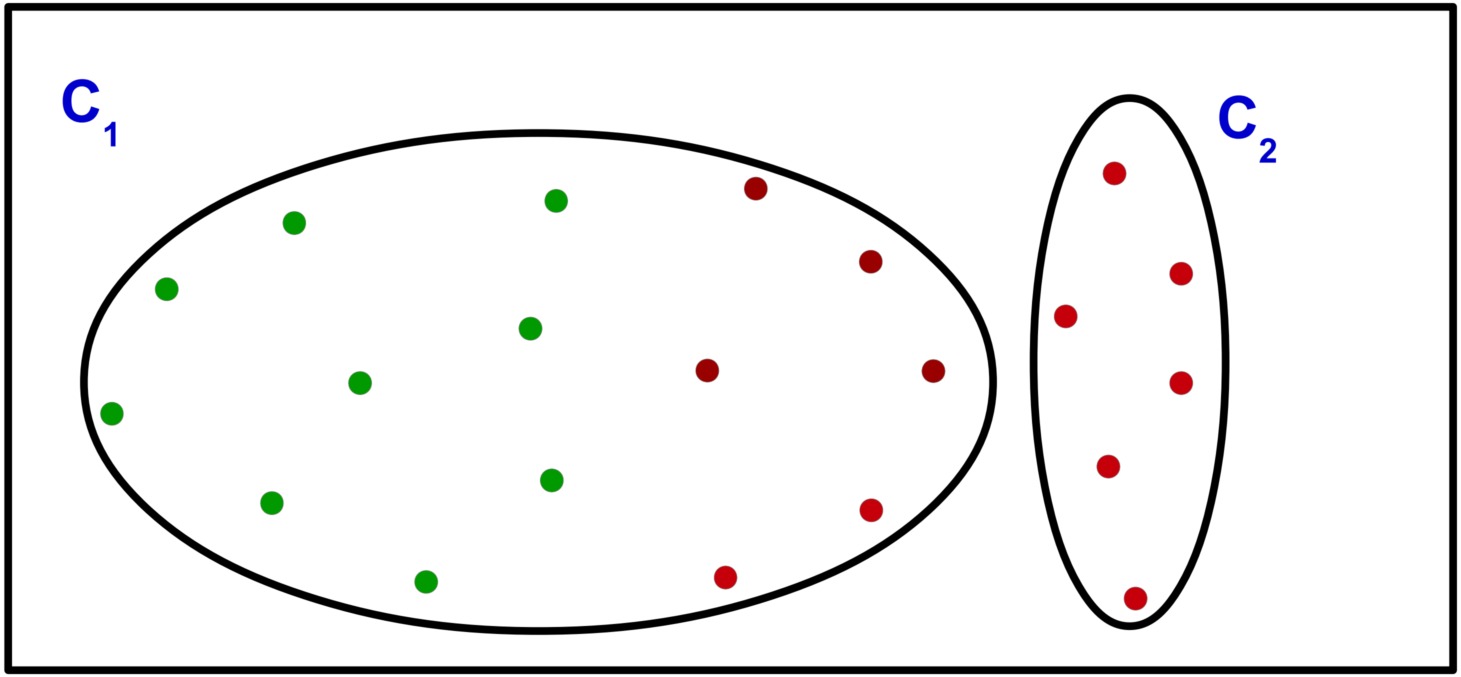
Density Increase

KDetect Auto-Stop

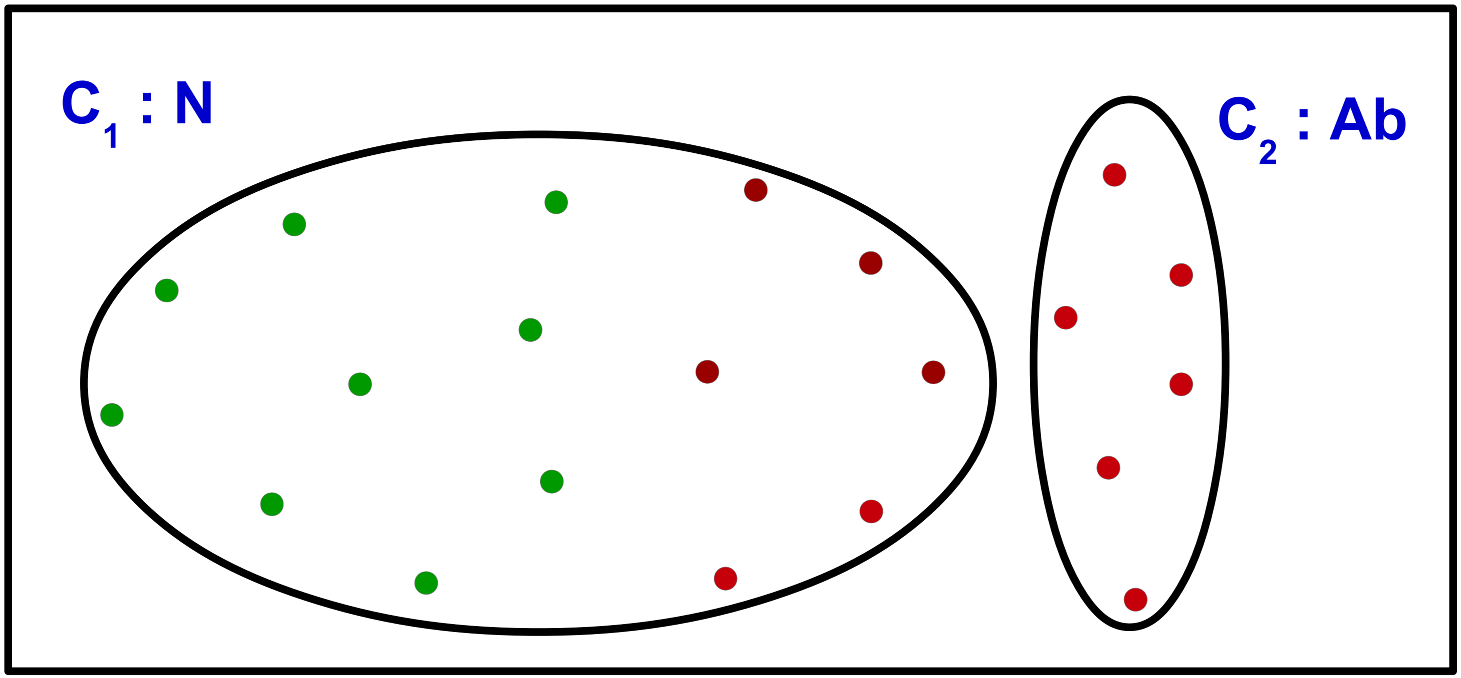
- Density (cluster compactness), Standard Deviation (time series variation).
- Threshold - density increase between 2 consecutive iterations.
- Thresholds - Locate good local optimum.
- Further iterations - Refinement.



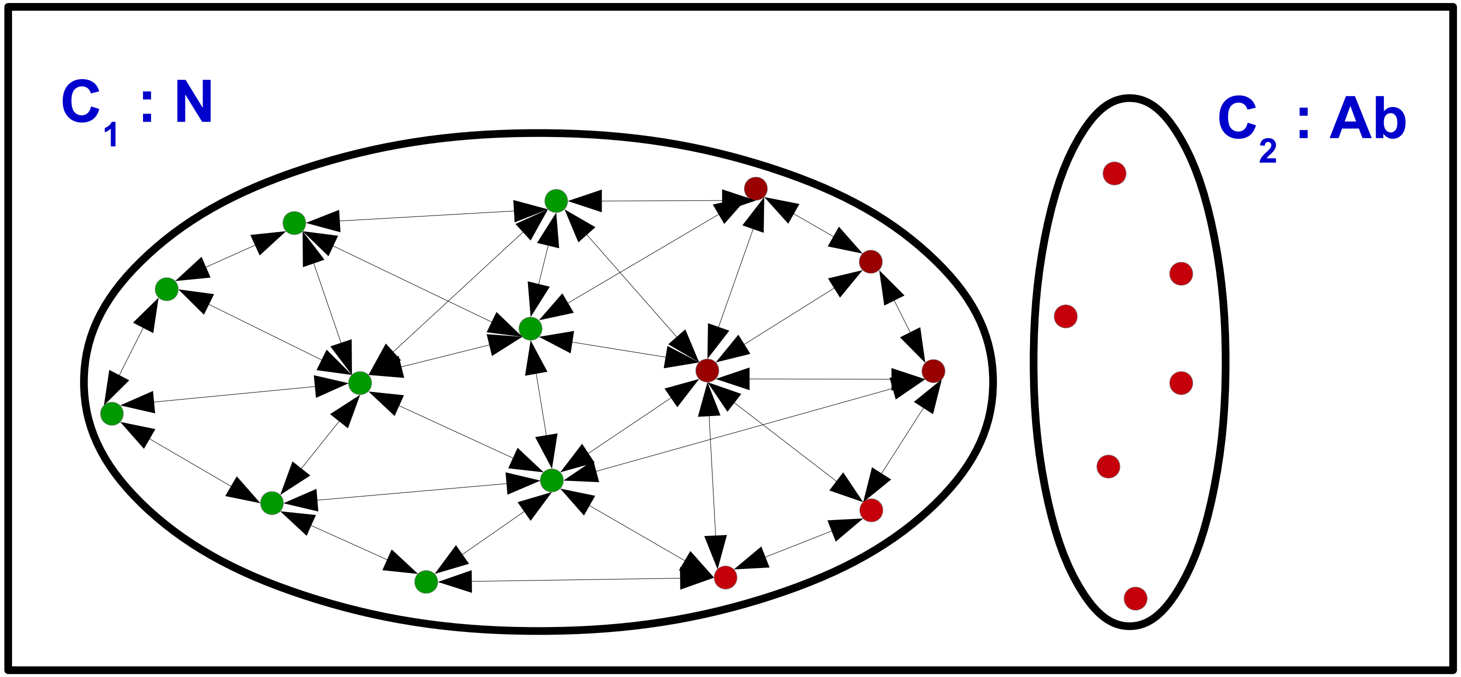
Cluster Labelling



Cluster Labelling

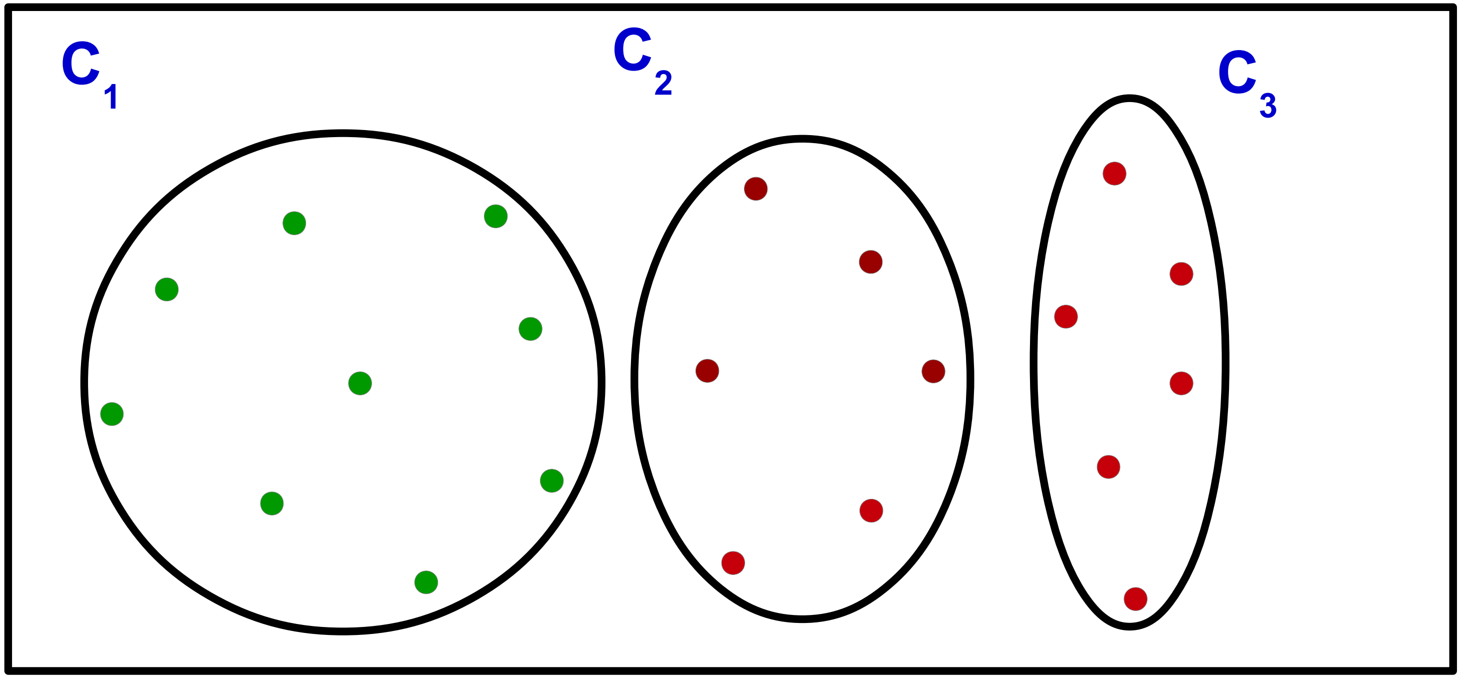


Cluster Labelling



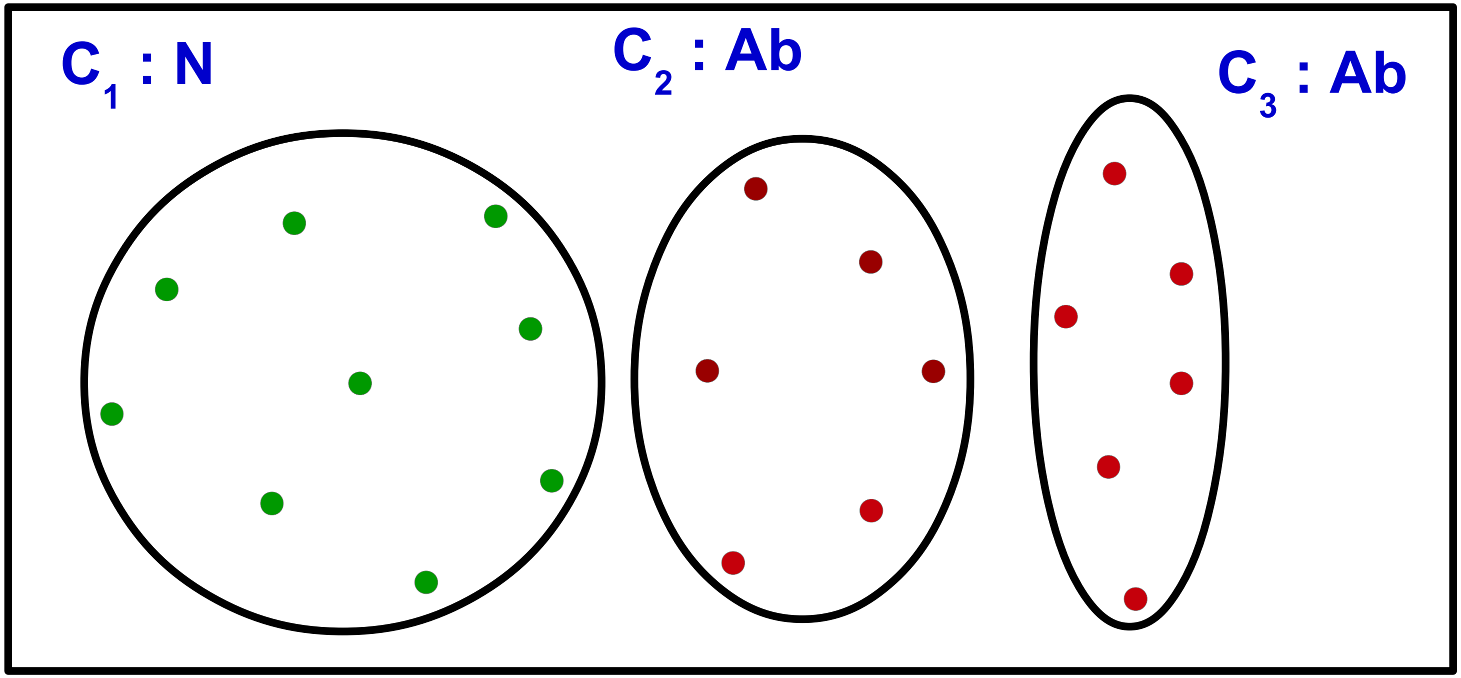
β = 2 x avg. dist. b/w any 2 points in Initial Normal Cluster.

Cluster Labelling



SBD between C_3 & initial normal cluster > β → abnormal label ('Ab').

Cluster Labelling



SBD between C_3 & initial normal cluster > β → abnormal label ('Ab').

Evaluation

- Performance Statistics
- Comparison with State-of-the-Art
- Auto-Stop Criteria
- Execution Time

Setup & Configuration

- K-Shape in Python3 → Tslearn v0.3.0
- Experiments conducted on Server -
    - CPU → 12-core Intel Xeon E5645.
    - Mem → 48 GB.
    - OS → Linux server edition – Debian 4.9.0-4-amd64.

Dataset

- **<u>Dataset Description -</u>**
  - Data Collection – French Company EasyVirt.
  - Production Data contains almost 2000 VMs.
  - 4 VMs illustrated –
    - Diverse normal and diverse abnormal behavior.
    - Differentiating normal from abnormal is not trivial.
  - Manual labelling by EasyVirt Experts to evaluate KDetect.
- **<u>Data Characteristics -</u>**
  - Total number of days for each VM ≈ 300.
  - 24-hour time windows to capture time series seasonality.
  - Averaged over 10 minute intervals - 144 points in each TS.
  - Metric = CPU consumption percentage.
  - Normal : Abnormal = 3:1.

Performance Statistics

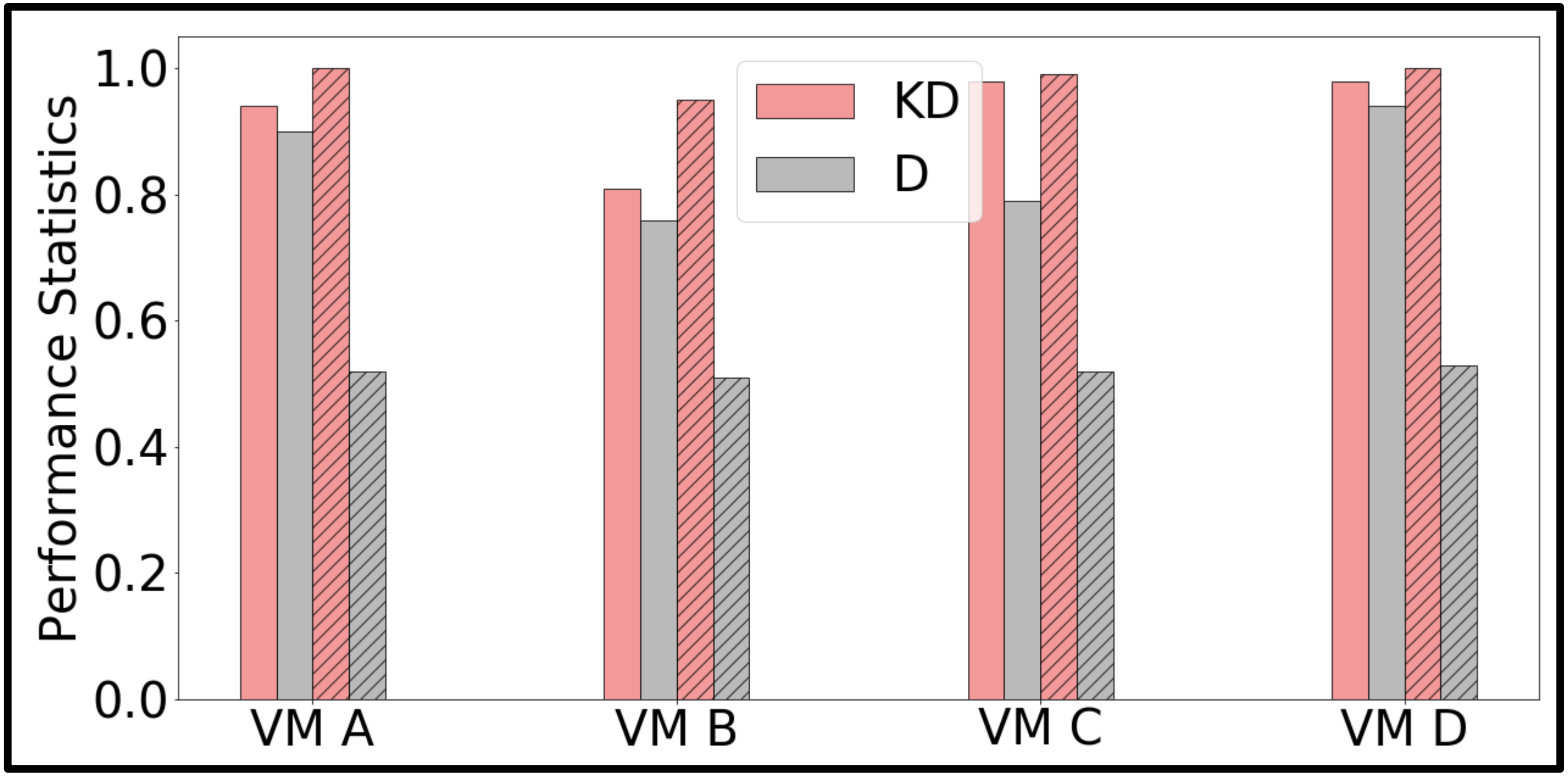
VM	Recall	Precision	FP %
A	0.94	1	0
B	0.81	0.95	1.11
C	0.98	0.99	0.31
D	0.99	1	0

KDetect - recall > 94% in most cases, precision > 95%.

Comparison with State-of-the-Art : Donut

- Implementation in Python3 using Tensorflow 1.5.0 by Donut authors.
- Reconstruction Probability Threshold → normal/abnormal.
  - Each VM - 1000 threshold values tested b/w lowest & highest probability.
- 60% training data & 40% testing data.

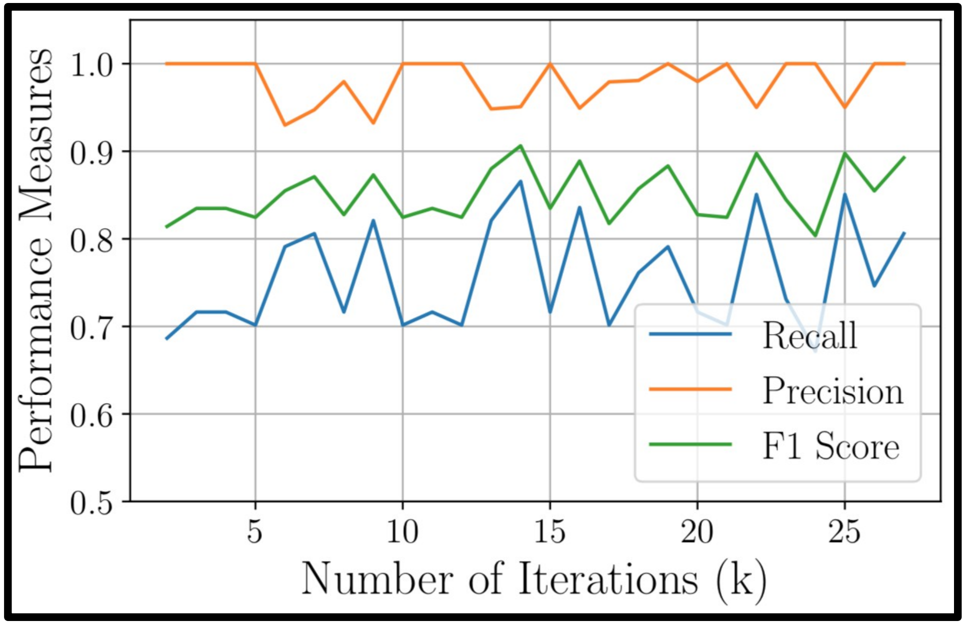
Comparison with State-of-the-Art : Donut



KDetect outperforms Donut - precision → 48%, recall → 20%.

Auto-Stop Criteria Analysis

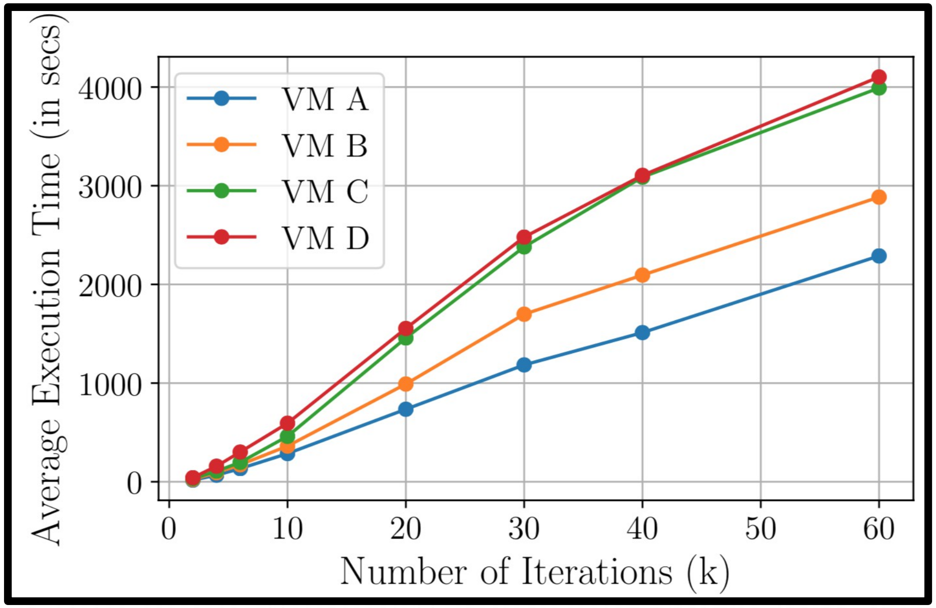
- Performance statistics for VM B.
- Stop at significant local optimum – not 1st.
- Tradeoff → execution time vs. precision.



KDetect selects “good” value of 'k'.

Execution Time Analysis

- Avg of 10 executions.
- Linear increase as function of 'k'.
- Same k → Different execution times for VMs as different sizes.



Execution Time Analysis

- Avg of 10 executions.
- Linear increase as function of 'k'.
- Same k → Different execution times for VMs as different sizes.

Virtual Machine	Auto-Stop Iteration (k)	Execution Time (sec)
VM A	5	100
VM B	7	172
VM C	3	63
VM D	3	101

Fast KDetect execution → < 3 mins in worst case (B).

Conclusions

- **KDetect** -
  - Unsupervised Learning Algorithm to identify anomalies.
  - Time Series exhibiting seasonal behavior.
  - Dynamic Partitional Clustering based solution.
  - Relies on generic heuristics to apply to large number of VMs.
  - Based on k-Shape as a building block.
- **Evaluation for multiple VM traces on production data -**
  - High precision, recall & low false positives.
  - Fast Execution.

Future Work

- Reinforcement Learning - improve Recall and Precision.
- Adapt to run online - reduce lead time for anomaly detection.

Thank You !!