Guide on General Purpose Anomaly Detection v2.x

Last updated on 19th April 2022

Overview

Key Files

- Core script(s):
  - Anomaly Detection v2.x-SERVER_SERVERRUNSPEC.ipynb → for general purpose anomaly detection (main script described)
  - Anomaly Detection Stitcher v1.x.ipynb → for stitching anomaly detection results with relevant event identifiers
- Reference File(s) for script:
  - ReferenceIndex.xlsx → For tagging event attribute information
  - ISCS Crash Notes.csv → For tagging potential ISCS “Blue Screen of Death” Crash Events
- Reference Files(s) for additional suppression of “Normal” state events (Optional Add-On):
  - ats-B0001-EventAttrUnique.xlsx
  - cms-B0001-EventAttrUnique.xlsx
  - ecs-B0001-EventAttrUnique.xlsx
- Note:
  - A CSV file version of the Excel files may also be used if one does not have the relevant library support for CSV files
  - “Normal” state refers to mundane events and the intended functioning state of the system when it is working normally, business as usual, and thus not limited to events with a “normal” value

General Concept

- Background:
    - Train operators are currently inundated by thousands of ISCS events per day, which results in potential tunnel vision and alarm fatigue
- Objective(s):
    - Strategic goal(s):
        - Reduce alarm fatigue of operators
        - Identify blind spots in system health
    - Tactical goal(s):
        - Identify anomalous patterns in the system event logs for triage by operator
            - This is to be used on conjunction with data visualisation methods

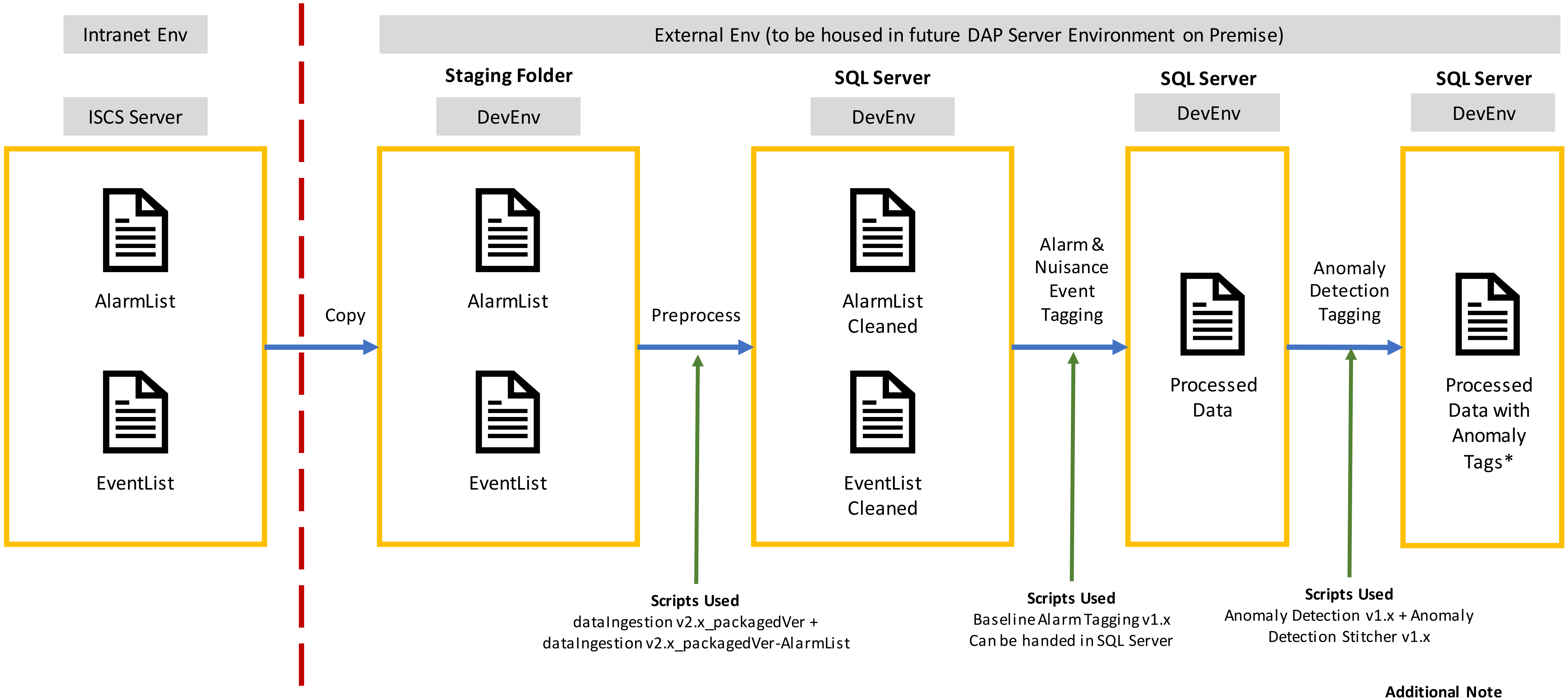
General Concept

- The issue of uncertainty
  - Known-known
    - Easy to define rule / strategy to isolate cases
    - Countermeasure Example(s):
      - Tagging of Nuisance Events to identify faulty equipment by predefined rules
  - Known-unknown
    - Targeted investigation based on hypothesis to convert to a known-known
    - Countermeasure Example(s):
      - Sump Pump Monitoring → Pending data availability in NEL
      - ISCS Crash Warning monitoring
  - Unknown-unknown
    - Difficult to define due to inherent blind spots and rarity of cases
    - Countermeasure Example(s):
      - Data visualisation to monitor macro level trends
      - Anomaly detection algorithms to identify cases for triage en masse

General Concept

- Nature of data
  - Unlike most other anomaly detection methods, we do not have the luxury of working with raw numeric data as predictors for anomaly detection
  - In contrast we have to work with text-based system logs
    - This data has been cleaned and had alarm events and nuisance events tagged prior
  - These text-based system logs only record discrete data (event states are process and logic driven) when a change of state occurs rather than a periodic record of the current system event state at a fixed time interval
  - The lack of any discrete transaction basket ID also makes association mining to detect possible correlations in event combos an impossible task
  - Anomalies in a well functioning system tends to be inherently rare making supervised learning impossible due to the lack of test cases for training
- Anomaly Detection limitations
  - More narrowly defined anomaly detection use cases can potentially yield better accuracy but lacks scalability
  - Anomaly detection algorithms are primarily based on statistical outliers / relative rarity, and hence a large sample window is required to baseline against
  - General purpose anomaly detection methods completements this by identifying statistical anomalies in the data
    - statistical anomaly ≠ negative anomaly of interest to operators
    - Batch size needs to be significantly large for statistical anomalies to be found
  - The issue of batch size would greatly impact the anomaly detection results, especially for CMS data which needs to be broken up into smaller chunks for analysis in order to keep the memory requirements manageable, as CMS data is a dumping ground for assorted miscellaneous events that could not be classified into ATS and CMS
  - Due to constant system maintenance and renewal, the environment would change overtime resulting in a potential drift in alarm patterns that must be accounted for
- Hence,
  - There is a need to extract the key categorical features in some form of machine readable format (usually in numeric form)
  - Event occurrence frequency is contingent on the nature of that specific system and needs to be benchmarked accordingly
  - General purpose anomaly detection algorithms to be used to detect anomalies (broad stroke)
  - Data visualisation dashboards (bird’s eye view) and more targeted anomaly detection use cases to be used in conjunction too

Dev WorkFlow For Anomaly Detection



Scripts Used
dataIngestion v2.x_packagedVer +
dataIngestion v2.x_packagedVer-AlarmList

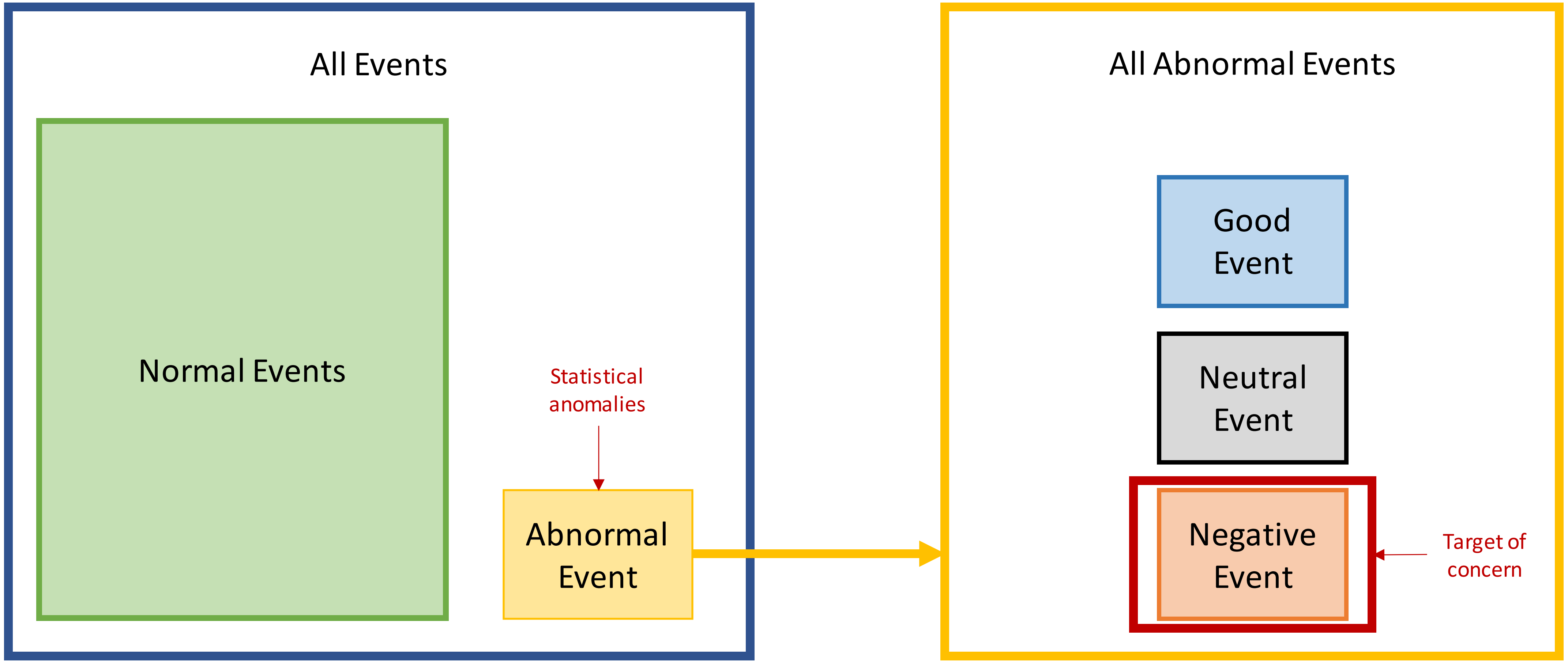
Scripts Used
Baseline Alarm Tagging v1.x
Can be handed in SQL Server

Scripts Used
Anomaly Detection v1.x + Anomaly Detection Stitcher v1.x

Note
In the original batch run, the processed files are exported as CSV files, which makes indexing and querying the files computationally expensive. Going forward, we would be exploring the storage of the data in SQL Server for the production version in the DAP server, though the intermediate processing steps of Alarm Tagging and Nuisance Event Tagging may be performed via SQL Server or Kafka.

Additional Note
Processed data + processed data with anomaly tags can then be piped to Splunk or Power BI for further analysis after being stored in SQL Server

Inherent Issues of Anomaly Detection



Methodology Overview

1. Load and compile cleaned data as a batch
    1. Categorisation of text logs by event attributes have been done at this stage
2. Tag data parameters with reference index data
3. Tag data parameters with ISCS crash notes data
4. Tag time values
5. Set data types
6. Junk redundant fields
7. Clean up event description data (optional step)
8. Baselining
9. Performing rolling window computation
10. Extract N-1 values for comparison
11. Perform data reduction / compression
12. Perform anomaly detection
13. Compile results for export

Key Variables Targeted

1. ENVIRONMENT
2. GeoCode
3. FUNCTIONAL_CATEGORY
4. AssetClass
5. AssetSubClass
6. EVENT_DESC_CAT
7. EVENT_STATUS
8. Severity_Class
9. SCS_TIME
10. isAlarm
11. NuisanceAlarm
12. RepeatAlarm
13. AltAlarm2
14. AltAlarm3
15. SeverityRank
16. CrashWarning
17. DayofWeek
18. Weekend
19. HourofDay
20. EngHours
21. RWEC_Ratio_Flat
22. RWEC_Ratio_Seasonal
23. RWEC_Ratio_SeasonalAlarm
24. RWSS_LocAsset
25. RWSS_LocAssetClass
26. RWSS_Loc
27. sentimentScore
28. SentimentClassImprove_N-1
29. SCS_TIME_logDelta_N-1
30. SEVERITY_GEO_N-1
31. isAlarm_GEO_N-1
32. NuisanceAlarm_GEO_N-1
33. CrashWarning_GEO_N-1
34. sentimentScore_GEO_N-1
35. SCS_TIME_logDelta_GEO_N-1
36. RWEC_Ratio_Flat_GEO_N-1
37. RWEC_Ratio_Seasonal_GEO_N-1
38. RWEC_Ratio_SeasonalAlarm_GEO_N-1

Detailed Methodology

Key libraries

- Python version: 3.8 onwards
- pandas – for basic data processing
- numpy – for basic data processing
- datetime – for time logging and manipulations
- time – for time logging
- sys – for system logging and operations
- os – for system logging and operations
- textblob – for sentiment analysis
- swifter – for acceleration of pandas operations using vectorisation
- memory_profiler – for logging memory consumption at the function level
- psutil – for logging memory consumption at that point instant
- sklearnex – for acceleration of scit-kit learn packages on Intel processors
- sklearn – for machine learning operations
- pyod – for anomaly detection operations

Initial Set Up

1. Load relevant libraries
2. Prepare relevant reference index files
3. Set current reference directories and input/output folders
4. Define run parameters
    1. Target server data (ATS/CMS/ECS)
    2. Window period
    3. Engineering Hours vs Revenue Hours
    4. Data Reduction (On/Off)
    5. Functional Category clusters
        1. Grouping events into logical functional category clusters is a data reduction strategy to reduce memory requirements
        2. It also has the effect of reducing the number of anomalies detected due to the baseline of defining a statistical anomaly changing
    6. Additional Cleaning Requirement (True / False)
5. Optional Steps:
    1. Periodic logging of memory requirements
    2. Display configuration

Load and compile cleaned data as a batch

1. Connect to relevant file directory
2. Load all relevant files in directory
    - In practice, this should be connected to a database where the last 5 weeks of data is loaded only via the use of some sort of query function
        - The last 4 weeks is used for batch analysis
        - Whereas the 1st week of data is used to account for cut-off error
            - A shorter time period of 1 additional day prior may be used for data reduction purposes instead
3. If the data size is too large, filter the data by functional category clusters to reduce the data size
4. Auto-terminate script if dataframe is empty

Tag data parameters with reference index data

1. Load event logs in reference index as a new dataframe (master/main index)
2. Tag Geographical information
   1. GeoCode
   2. GeographicalCat
   3. GeoSector
3. Tag Severity information
   1. Serverity_Class
   2. ServerityRank
4. Load ISCS Crash combo look up table

Tag time values

1. Convert time values into a datetime data type
    1. DATETIME_SENT (Optional Step as field would be deleted)
    2. DATETIME_RECEIVED (Optional Step as field would be deleted)
    3. EQUIPMENT_DATE (Optional Step as field would be deleted)
    4. ACQUISITION_DATE (Optional Step as field would be deleted)
    5. SCS_TIME
    6. TIME_CODE
2. Decompose SCS_Time to extract the following values
    1. Date
    2. Day of week
    3. Weekend (True/False)
    4. Hour of day
    5. Minute of day
3. Filter events by Engineering Hours or Revenue Hours
    1. Engineering Hours is from 0030hrs to 0430hrs daily
    2. Filter option is dependent on the run

Set Data Types and Junk Redundant Fields

1. Convert fields to the correct datatype
    1. 'ALARM_ID': 'str' (Optional Step as field would be deleted)
    2. 'USER_ID': 'str' (Optional Step as field would be deleted)
    3. 'VALUE': 'str' (Optional Step as field would be deleted)
    4. 'VALUE_STATE': 'category' (Optional Step as field would be deleted)
    5. 'ACKNOWLEDGEMENT_REQUIRED': 'bool'
    6. 'GeoSector': 'category'
    7. 'Severity_Class': 'category'
    8. 'HIDDEN': 'bool' (Optional Step as field would be deleted)
    9. 'THEME': 'category' (Optional Step as field would be deleted)
    10. 'FUNCTIONAL_CATEGORY': 'category'
    11. 'GEOGRAPHICAL_CATEGORY': 'category'
    12. 'ASSET_ID_RAW': 'str'
    13. 'ASSET_DESCRIPTION': 'str' (Optional Step as field would be deleted)
    14. 'EVENT_DESCRIPTION': 'str' (Optional Step as field would be deleted)
    15. 'EVENT_STATUS': 'category'
    16. 'OPERATOR_INITIALS': 'str' (Optional Step as field would be deleted)
    17. 'ASSET_DESC_CAT': 'category'
    18. 'EVENT_DESC_CAT': 'category'
    19. 'TrainID': 'str' (Optional Step as field would be deleted)
    20. 'CarID': 'str' (Optional Step as field would be deleted)
    21. 'ServiceID': 'str' (Optional Step as field would be deleted)
    22. 'AssetClass': 'category'
    23. 'AssetSubClass': 'category'

2. Drop irrelevant columns below
    1. ENTRY_CODE
    2. ALARM_ID
    3. USER_ID
    4. VALUE
    5. VALUE_STATE
    6. HIDDEN
    7. THEME
    8. EQUIPMENT_DATE
    9. ACQUISITION_DATE
    10. ASSET_DESCRIPTION
    11. EVENT_DESCRIPTION
    12. OPERATOR_INITIALS
    13. TrainID
    14. CarID
    15. ServiceID
    16. DATETIME_SENT
    17. DATETIME_RECEIVED
    18. USER1

Clean up event description data (optional step)

1. Remove train direction info in df[“EVENT_DESC_CAT”]
    1. E.g. N/B , N-N, N, S/B, S-S, S
2. Simplify control takeover scenarios for df[“EVENT_DESC_CAT”]
    1. If contains “Automatic Hold applied to TrainCar stalled” to “Automatic Hold applied to TrainCar stalled”
    2. If contains “Control Hand Over for ECS - Environmental Control System” to “Control Hand Over for ECS - Environmental Control System”
    3. If contains “Control Hand Over for ECS - Smoke Extraction System” to “Control Hand Over for ECS - Smoke Extraction System”
    4. If contains “Control Hand Over for ECS - Tunnel Ventilation System” to “Control Hand Over for ECS - Tunnel Ventilation System”
    5. If contains “Control Hand Over for SIG - Control Train ATC” to “Control Hand Over for SIG - Control Train ATC”
    6. If contains “Control Hand Over for SIG - Platform Equipment” to “Control Hand Over for SIG - Platform Equipment”
    7. If contains “Control Hand Over for SIG - Track Side Equipment” to “Control Hand Over for SIG - Track Side Equipment”
    8. If contains “Control Hand Over for TrainBorne CCTV” to “Control Hand Over for TrainBorne CCTV”
    9. If contains “Control Hand Over for TrainBorne PA” to “Control Hand Over for TrainBorne PA”
    10. If contains “Control Hand Over for TrainBorne PEC” to “Control Hand Over for TrainBorne PEC”
    11. If contains “Control Hand Over for TrainBorne PIS/VPIS” to “Control Hand Over for TrainBorne PIS/VPIS”
    12. If contains “Control Take Over for All Functions” to “Control Take Over for All Functions”
    13. If contains “Control Take Over for PIS - Passenger Information” to “Control Take Over for PIS - Passenger Information”

Clean up event description data (optional step)

3. Simplify Operator Calls scenarios for df[“EVENT_DESC_CAT”]
    1. If contains “accepts a PEC call” to “OPERATOR accepts a PEC call”
    2. If contains “terminates all PEC calls” to “OPERATOR terminates all PEC calls”
    3. If contains “terminates PEC call” to “OPERATOR terminates PEC call”
4. Simplify df[“EVENT_DESC_CAT”] for the following scenarios
    1. If contains “Automatic Hold applied to TrainCar stalled” to “Automatic Hold applied to TrainCar stalled”
    2. If contains “Free all paths for Station” to “Free all paths for Station”
    3. If contains “Gama Status Request For an Atc” to “Gama Status Request For an Atc”
    4. If contains “change password on NelVisu” to “OPERATOR change password on NelVisu”
    5. If contains “Track-Side Atc Status Request for An Atc” to “Track-Side Atc Status Request for An Atc”

Clean up event description data (optional step)

5. Simplify Operator Calls scenarios for df[“EVENT_DESC_CAT”]
    1. If contains “Train found at” and “instead of Train” to “Train found at SUBLOCATION instead of Train”
    2. If contains “Train still not a Man RTL” and “origin after wait period” to “Train still not a Man RTL origin after wait period”
    3. If contains “Timetable” and “download” to “Timetable download”
    4. If contains “Timetable” and “successfully autoloaded” to “Timetable successfully autoloaded”

Baselining

- The general idea is to get a baseline to compare the point value of the current set against
- Baselining is meant to account for seasonal variations in the amount of events during different days of the week and times of day
- Large deviations from the baseline values could be indicative of an anomaly
- This should be baseline data should ideally be taken from a period of stability and good performance in terms of system health, with as long a period as possible, and not the same sample window being tested against.
- The latter is to ensure a consistent time window for comparison and to reduce unnecessary repeat computations which would slow down the script
- However, this baseline should be updated from time to time to account for drift in system behaviour over time when the system are progressively updated

Baselining

1. Get event count as the basic baseline by the following as a new dataframe
    1. Date
    2. ENVIRONMENT
    3. FUNCTIONAL_CATEGORY
    4. DayofWeek
    5. HourofDay
2. Get hourly event count threshold based on the 90th Percentile value by by the following as a new dataframe
    1. ENVIRONMENT
    2. FUNCTIONAL_CATEGORY

Baselining

3. Filter out events of asset classes and functional categories with very low occurrence rates
4. Get seasonal hourly event count threshold based on the 90th Percentile value by the following as a new dataframe
   1. Date
   2. ENVIRONMENT
   3. FUNCTIONAL_CATEGORY
   4. AssetClass
   5. DayofWeek
   6. HourofDay

Baselining

5. Get seasonal hourly alarm event count threshold based on the 90th Percentile value by by the following as a new dataframe
    1. Date
    2. ENVIRONMENT
    3. FUNCTIONAL_CATEGORY
    4. AssetClass
    5. isAlarm
    6. DayofWeek
    7. HourofDay
6. Flatten multi-index dataframe of event count threshold computed for easier indexing later
7. Merge event count threshold values computed to main dataframe
8. Export event threshold values as a CSV for future reference
    1. In future runs, one may load the threshold values for comparison from these CSV files
9. Delete intermidairy dataframes containing event threshold values

Performing rolling window computation

- The rolling window computation would allow one to compare the current event count against the baseline values
- It would also provide an aggregated view of any system degradation over time using severity ranking
  - Others < Low < Urgent < Critical
  - Maintenance < Operational
  - isAlarm False < isAlarm True
  - NuisanceAlarm True < NuisanceAlarm False
  - NuisanceAlarm False < isAlarm True

Performing rolling window computation

1. Create a new copy of the main dataframe, keeping only the following fields
    1. SCS_TIME
    2. ENVIRONMENT
    3. FUNCTIONAL_CATEGORY
    4. AssetClass
    5. isAlarm
    6. ENTRY_CODE_SUFFIX
2. Rename the field “ENTRY_CODE_SUFFIX” to “Counter” for Ease of Lookup later
3. Compute df_rollingWindow_flat
    1. Perform a rolling window count of “Counter” of 1hr with a minimum period of 1, by the following as a new dataframe
        1. ENVIRONMENT
        2. FUNCTIONAL_CATEGORY
    2. Flatten computed rolling window count multi-index dataframe and drop any duplicates
    3. Merge computed rolling window values with main dataframe
    4. Delete dataframe with rolling window values
    5. Compute ratio of rolling window values against baseline values
        1. df["RWEC_Ratio_Flat"] = (df["RWEC_Flat"] / df["ECThreshold_Flat"])
    6. Drop redundant values in main dataframe

Performing rolling window computation

4. Compute Rolling Window Count - Seasonal
    1. Perform a rolling window count of “Counter” of 1hr with a minimum period of 1, by the following as a new dataframe
        1. ENVIRONMENT
        2. FUNCTIONAL_CATEGORY
        3. Asset_Class
    2. Flatten computed rolling window count multi-index dataframe and drop any duplicates
    3. Merge computed rolling window values with main dataframe
    4. Delete dataframe with rolling window values
    5. Compute ratio of rolling window values against baseline values
        1. df["RWEC_Ratio_Seasonal"] = (df["RWEC_Seasonal"] / df["ECThreshold_Seasonal"])
    6. Drop redundant values in main dataframe

Performing rolling window computation

5. Compute Rolling Window Count – Seasonal Alarm
    1. Perform a rolling window count of “Counter” of 1hr with a minimum period of 1, by the following as a new dataframe
        1. ENVIRONMENT
        2. FUNCTIONAL_CATEGORY
        3. Asset_Class
        4. isAlarm
    2. Flatten computed rolling window count multi-index dataframe and drop any duplicates
    3. Merge computed rolling window values with main dataframe
    4. Delete dataframe with rolling window values
    5. Compute ratio of rolling window values against baseline values
        1. df["RWEC_Ratio_SeasonalAlarm"] = (df["RWEC_SeasonalAlarm"] / df["ECThreshold_SeasonalAlarm"])
    6. Drop redundant values in main dataframe

Performing rolling window computation

6. Create a new copy of the main dataframe, keeping only the following fields for rolling window computation of severity level
    1. SCS_TIME
    2. ENVIRONMENT
    3. FUNCTIONAL_CATEGORY
    4. GEOGRAPHICAL_CATEGORY
    5. AssetClass
    6. AssetSubClass
    7. isAlarm
    8. NuisanceAlarm
    9. SeverityRank
    10. EQUIPMENT_NAME
    11. ASSET_ID_RAW

Performing rolling window computation

7. Compute moderated event severity score for each event
    1. moderation_baseline = 0.1
    2. moderation_isAlarm = 0.55
    3. isAlarm = True (1) or False (0)
    4. NuisanceAlarm = True (1) or False (0)
    5. moderation_nonNuisance = 1 - moderation_isAlarm - moderation_baseline
    6. SeverityScore = SeverityRank * ((isAlarm] * moderation_isAlarm) + ((NuisanceAlarm == False) * moderation_nonNuisance) + moderation_baseline)
8. Delete redundant variables in rolling window dataframe

Performing rolling window computation

8. Compute df_SeverityRank_LocAssetClass
   1. Perform rolling window mean on severity score by
      1. ENVIRONMENT
      2. GEOGRAPHICAL_CATEGORY
      3. EQUIPMENT_NAME
      4. ASSET_ID_RAW
   2. Drop redundant variables
   3. Flatten multi-index dataframe
   4. Drop duplicates
   5. Merge computed values to main dataframe
   6. Delete intermediary dataframe with rolling window values
   7. Drop redundant variables after merger

Performing rolling window computation

9. Compute df_SeverityRank_LocAsset
    1. Perform rolling window mean on severity score by
        1. ENVIRONMENT
        2. GEOGRAPHICAL_CATEGORY
        3. AssetClass
        4. AssetSubClass
    2. Drop redundant variables
    3. Flatten multi-index dataframe
    4. Drop duplicates
    5. Merge computed values to main dataframe
    6. Delete intermediary dataframe with rolling window values
    7. Drop redundant variables after merger

Performing rolling window computation

10. Compute df_SeverityRank_Loc
    1. Perform rolling window mean on severity score by
        1. ENVIRONMENT
        2. GEOGRAPHICAL_CATEGORY
    2. Drop redundant variables
    3. Flatten multi-index dataframe
    4. Drop duplicates
    5. Merge computed values to main dataframe
    6. Delete intermediary dataframe with rolling window values
    7. Drop redundant variables after merger
11. Drop redundant variables
12. Impute infinity values to 0
13. Filter data to the desired window period to keep data manageable
    1. This step should be done upfront if baselining has already been done in a separate run

Extract N-1 values for comparison

- The general idea is to compare the current state against potential precursor / prior values that may indicate a degradation in system health
  - Ideally more prior values should be used, but that would increase the computation load and one would be capped by the hardware limitationsl
- This is also supported via the use of text analytics tools like sentiment scoring of the event description fields to identify potential event state degradation
  - Granted that the logs are system generated, sentiment analysis is less precise in identifying changes in system health
  - However, it can be used to complement the scoring based on severity score, both of which, were based on arbitrarily pre-defined scoring methods

Extract N-1 values for comparison

1. Create new combined field call EventAttr
   1. df["FUNCTIONAL_CATEGORY"] + " - " + df["EVENT_DESC_CAT"] + " - " + df["EVENT_STATUS"]
2. Perform sentiment analysis on EventAttr to get sentiment score
3. Classify sentiment score results as positive or negative or neutral
   1. Sentiment score < 0: negative
   2. Sentiment score = 0: neutral
   3. Sentiment score > 0: positive

Extract N-1 values for comparison

4. Extract past N-1 values from the same asset
    1. Sort values by
        1. ENVIRONMENT
        2. FUNCTIONAL_CATEGORY
        3. EQUIPMENT_NAME
        4. ASSET_ID_RAW
        5. SCS_TIME
    2. Get the following values by ENVIRONMENT, FUNCTIONAL_CATEGORY, EQUIPMENT_NAME and ASSET_ID_RAW
        1. SEVERITY
        2. SEVERITY_Worsen $\rightarrow$ computed via the difference between Severity Rank N-0 and N-1
        3. isAlarm
        4. NuisanceAlarm
        5. CrashWarning
        6. sentimentScore
        7. SentimentClass
        8. sentimentScoreDelta
        9. SentimentClassImprove $\rightarrow$ computed via the difference between Severity Class N-0 and N-1
        10. SCS_TIME_logDelta $\rightarrow$ computed via the log difference between the SCS_TIME N-0 and N-1
        11. Event Attr $\rightarrow$ get this sole field, ignoring the rest, if data reduction is activated

Extract N-1 values for comparison

5. Extract past N-1 values from the same asset
    1. Sort values by
        1. ENVIRONMENT
        2. FUNCTIONAL_CATEGORY
        3. GEOGRAPHICAL_CATEGORY
        4. EQUIPMENT_NAME
        5. ASSET_ID_RAW
        6. SCS_TIME
    2. Get the following values by ENVIRONMENT, FUNCTIONAL_CATEGORY and GEOGRAPHICAL_CATEGORY
        1. SEVERITY
        2. SEVERITY_Worsen → computed via the difference between Severity Rank N-0 and N-1
        3. isAlarm
        4. NuisanceAlarm
        5. CrashWarning
        6. sentimentScore
        7. SentimentClass
        8. sentimentScoreDelta
        9. SentimentClassImprove → computed via the difference between Severity Class N-0 and N-1
        10. SCS_TIME_logDelta → computed via the log difference between the SCS_TIME N-0 and N-1
        11. RWEC_Ratio_Flat_GEO
        12. RWEC_Ratio_Seasonal_GEO
        13. RWEC_Ratio_SeasonalAlarm_GEO
        14. Event Attr → get this sole field, ignoring the rest, if data reduction is activated

Final Data Preparation

1. Check if dataframe is empty and terminate it if empty
2. Drop redundant fields
3. Convert all null values to 0
4. Export main dataframe as a CSV (optional step as save point)
5. Create a copy of main dataframe to perform anomaly detection work on (new dataframe to be called df0), whilst dropping the following values
    1. EQUIPMENT_NAME
    2. ASSET_ID_RAW
    3. FUNCTIONAL_CATEGORY
    4. EVENT_DESC_CAT
    5. ENVIRONMENT
    6. EVENT_STATUS
    7. Severity_Class
6. Perform hash encoding of categorical values in df0
    1. Traditional dummy and one-hot encoding would take up too much memory
7. Impute infinity values to 0

Anomaly Detection

- For this general purpose anomaly detection, we would be using an ensemble method of 5 different algorithms to generate predictions which would be voted on based on
  - Anomaly classification
  - Mean anomaly probability
- These 5 algorithms are chosen for their memory efficiency and speed as we need to keep compute costs low and have fast turnover of the data analysis
- Without ground truth tagging, it would be impossible to tune any of the algorithms' parameters with any remote accuracy, hence the parameter values used are merely a best guess

Anomaly Detection

1. Perform PCA from the PYOD library on the data in df0 to generate classifications and anomaly probability
    1. Settings:
        1. Explained variance: 0.99 or 0.8 (data reduction mode)
        2. contaminationAmt = 0.0002
2. Perform PCA using the sklearn library (scit-learn) to perform data reduction on the data
    1. Settings:
        1. n_components: 0.95 or 0.8 (data reduction mode)
    2. From experimental observation, this does not meaningfully reduce the data size, hence this step is optional. PCA on top of Hash encoding may not make logical sense analytics wise too.

Anomaly Detection

3. Perform Isolation Forest from the PYOD library on the data in df0 to generate classifications and anomaly probability
    1. Settings:
        1. n_jobs (CPU cores) = -1 (all)
        2. contaminationAmt = 0.0002
        3. random_state = 888
4. Perform Histogram-based Outlier Score (HBOS) from the PYOD library on the data in df0 to generate classifications and anomaly probability
    1. Settings:
        1. n_bins= 10
        2. contaminationAmt = 0.0002
        3. Alpha = 0.1
        4. Tol = 0.5

Anomaly Detection

5. Perform Cluster-based Local Outlier Factor (CBLOF) from the PYOD library on the data in df0 to generate classifications and anomaly probability
    1. Settings:
        1. n_jobs (CPU cores) = -1 (all)
        2. contaminationAmt = 0.0002
        3. random_state = 888
        4. check_estimator = False
6. Perform Lightweight On-line Detector of Anomalies (LODA) from the PYOD library on the data in df0 to generate classifications and anomaly probability
    1. Settings:
        1. n_jobs (CPU cores) = -1 (all)
        2. contaminationAmt = 0.0002
        3. random_state = 888
        4. check_estimator = False

Anomaly Detection

1. Get mean anomaly probability using the anomaly probabilities of the 5 anomaly detection algorithms
2. If the mean anomaly probability is 70% or higher, classify it as an anomaly
3. Compute outlier vote total of each event but summing the values of the classifications of the 5 anomaly detection algorithms
    1. True = 1
    2. False = 0
4. Export data as CSV

Anomaly Detection

5. Append event identifiers to the anomaly detection files (keep original sorting) using Anomaly Detection Stitcher v1.1.ipynb and export as CSV
    1. Event Identifier files: SERVER-##-AnomalyTagging_RAW
    2. Anomaly Detection Outputs: SERVER-##-AnomalyTagging_Output
    3. Depending on the final pipeline design, exporting as CSV may not be necessary
6. Additional suppression of false positive events (normal-type events) by comparing events against the relevant index list (pre-identified by SMEs). This may be done at the dashboard layer.
    1. ats-B0001-EventAttrUnique.xlsx
    2. cms-B0001-EventAttrUnique.xlsx
    3. ecs-B0001-EventAttrUnique.xlsx

Appendix

Anomaly Detection Data Output

Env	File Name	Eng/Rev Hr	Processing Method
ATS	ats-B0001-anomalyTaggedFULL	Eng	No Data Reduction + ALL Clusters
ATS	ats-EngHr-B0001-anomalyTaggedFULL	Rev	No Data Reduction + ALL Clusters
ECS	ecs-B0001-anomalyTaggedFULL	Eng	No Data Reduction + ALL Clusters
ECS	ecs-EngHr-B0001-anomalyTaggedFULL	Reb	No Data Reduction + ALL Clusters
CMS	cms-B0001-anomalyTaggedFULL-AltRun	Rev	Data Reduction + AltRun Cluster
CMS	cms-B0001-anomalyTaggedFULL-CCTV-A	Rev	Data Reduction + CCTV Cluster (first 2 weeks)
CMS	cms-B0001-anomalyTaggedFULL-CCTV-B	Rev	Data Reduction + CCTV Cluster (next 2 weeks)
CMS	cms-B0001-anomalyTaggedFULL-PassengerInfo	Rev	Data Reduction + PassengerInfo Cluster
CMS	cms-EngHr-B0001-anomalyTaggedFULL-All	Eng	Data Reduction + ALL Clusters

Note
ITESS and StationEquipment for the CMS Environment have too few relevant events for anomaly detection

Anomaly Detection Data Output

Env	File Name (Single File / Subset)	Eng/Rev Hr	Processing Method
CMS	cms-B0001-anomalyTaggedFULL-Comms	Rev	No Data Reduction + Comms Cluster
CMS	cms-B0001-anomalyTaggedFULL-Depot	Rev	No Data Reduction + Depot Cluster
CMS	cms-B0001-anomalyTaggedFULL-FireProtection	Rev	No Data Reduction + FireProtection Cluster
CMS	cms-B0001-anomalyTaggedFULL-LENV	Rev	No Data Reduction + LENV Cluster
CMS	cms-B0001-anomalyTaggedFULL-Lighting	Rev	No Data Reduction + Lighting Cluster
CMS	cms-B0001-anomalyTaggedFULL-PowerSys	Rev	No Data Reduction + PowerSys Cluster
CMS	cms-B0001-anomalyTaggedFULL-SCS	Rev	No Data Reduction + SCS Cluster
CMS	cms-B0001-anomalyTaggedFULL-StationFacilities	Rev	No Data Reduction + StationFacilities Cluster
CMS	cms-EngHr-B0001-anomalyTaggedFULL-AltRun	Eng	Data Reduction + AltRun Cluster

Note
ITESS and StationEquipment for the CMS Environment have too few relevant events for anomaly detection

Memory Load Requirements

Stage / Process	ATS – Main (MB)	CMS – Est (MB)	ECS – Main (MB)
Initialisation	177.3	173.8	176.9
Base Data	976.1	4,700.0	185.3
FN MinMaxScaler	19,650.3 (+2,492.8)	Too huge to be processed in 1 run	6,588.23 (+1,950.5)
Pre-Anomaly Detection	19,772.1	Too huge to be processed in 1 run	6,626.8
FN anomalyAlgo_PCA *	10,547.5 (+5,645.3)	Too huge to be processed in 1 run	5,654.8 (+674.8)
FN PCA	5,551.3 / 3,099.5 (+564.1 / +0.0)	Too huge to be processed in 1 run	4,744.2 / 4,705.7 (-253.8 0.0)
FN anomalyAlgo_IsoForest *	18,998.4 (+15,898.7)	Too huge to be processed in 1 run	5,823.9 (+1,118.2)
FN anomalyAlgo_HBOS	4096.7 (+802.5)	Too huge to be processed in 1 run	4,842.1 (+116.1)
FN anomalyAlgo_CBLOF	5,481.1 (+1,940.4)	Too huge to be processed in 1 run	4,880.8 (+95.2)
FN anomalyAlgo_LODA	4,573.9 (+774.5)	Too huge to be processed in 1 run	4,922.5 (+97.6)
Finalised Data	365.5	Too huge to be processed in 1 run	50.4
Active Memory Before Export	19,772.1	Too huge to be processed in 1 run	6,626.8

Notes:
Key Window Period: 1 Jan 2021 (inclusive) to 29 Jan 2021 (exclusive)
Data Dim. At Export: ATS: 3304063x24 | CMS: 19,604,630x24 | ECS: 459392x24

Memory Load Requirements

Stage / Process	CMS – Comms (MB)	CMS – Depot (MB)
Initialisation	173.8	173.6
Base Data	4,700.0	4,700.0
Partitioned Base Data by FuncCat (All day)	386.6	0.7
FN MinMaxScaler	18,254.4 (+2,777.8)	8,010.6 (+11.9)
Pre-Anomaly Detection	24,969.9	24,973.6
FN anomalyAlgo_PCA *	16,815.0 (+829.0)	8,018.3 (+3.2)
FN PCA	15,598.8 / 15,566.0 (-414.2 / +0.0)	8,018.8 / 8018.8 (+0.6 / +0.0)
FN anomalyAlgo_IsoForest *	16,804.8 (+1,238.7)	8,020.1 (+1.3)
FN anomalyAlgo_HBOS	15,873.7 (+244.9)	8,042.1 (+22.1)
FN anomalyAlgo_CBLOF	15,939.4 (+227.3)	8,044.3 (+2.1)
FN anomalyAlgo_LODA	16,008.8 (+217.9)	8,044.4 (+.1)
Finalised Data	105.5	0.1
Active Memory Before Export	24,969.9	24,973.6

Notes:
Key Window Period: 1 Jan 2021 (inclusive) to 29 Jan 2021 (exclusive)
Data Dim. At Export: ATS: 3,304,063x24 | CMS: 19,604,630x24 | ECS: 459,392x24

Memory Load Requirements

Stage / Process	CMS – FireProtection (MB)	CMS – ITESS (MB)	CMS – LENV (MB)
Initialisation	173.4	174.0	173.5
Base Data	4,700.0	4,700.0	4,700.0
Partitioned Base Data by FuncCat (All day)	0.2	0.2	0.07
FN MinMaxScaler	8,011.3 (+11.2)	Insufficient Data	7,907.4 (+12.3)
Pre-Anomaly Detection	24,978.5	Insufficient Data	24,869.6
FN anomalyAlgo_PCA *	8,018.0 (+1.8)	Insufficient Data	7,915.5 (+1.9)
FN PCA	8,018.3 / 8,018.3 (+0.3 / +0.0)	Insufficient Data	7,915.9 / 7,915.9 (+0.3 / +0.0)
FN anomalyAlgo_IsoForest *	8,019.7 (+1.4)	Insufficient Data	7,917.2 (+1.3)
FN anomalyAlgo_HBOS	8,041.5 (+22.0)	Insufficient Data	7,945.9 (+28.9)
FN anomalyAlgo_CBLOF	8,043.5 (+1.8)	Insufficient Data	7,950.4 (+4.2)
FN anomalyAlgo_LODA	8,043.5 (+0.1)	Insufficient Data	7,950.4 (+0.1)
Finalised Data	0.03	Insufficient Data	0.01
Active Memory Before Export	24,978.5	Insufficient Data	24,869.6

Notes:
Key Window Period: 1 Jan 2021 (inclusive) to 29 Jan 2021 (exclusive)
Data Dim. At Export: ATS: 3,304,063x24 | CMS: 19,604,630x24 | ECS: 459,392x24

Memory Load Requirements

Stage / Process	CMS – Lighting (MB)	CMS – SCS (MB)	CMS – PassengerInfo (MB)
Initialisation	173.4	173.7	173.4
Base Data	4,700.0	4,700.0	4,700.0
Partitioned Base Data by FuncCat (All day)	3.2	295.3	**1,300**
FN MinMaxScaler	8,024.7 (+14.0)	16,988.8 (+2,849.3)	45,820.0 (+11,531.4)
Pre-Anomaly Detection	24,971.5	24,971.1	45,820.0
FN anomalyAlgo_PCA *	8034.5 (+5.3)	15,623.3 (+978.7)	40,186.8 (+3,906.1)
FN PCA	8,031.1 / 8,031.1 (-1.3 / +0.0)	14,319.8 / 14,249.6 (-346.0 / -0.1)	36,715.2 / 34816. 8 (+355.7 / +0.0)
FN anomalyAlgo_IsoForest *	8,032.8 (+1.7)	15,463.3 (+1,213.7)	41,539.4 (+6,722.6)
FN anomalyAlgo_HBOS	7,997.8 (+29.1)	14,455.1 (+213.8)	35,785.0 (+815.1)
FN anomalyAlgo_CBLOF	8,003.3 (+5.5)	14,564.0 (+242.8)	36,569.5 (+1,375.5)
FN anomalyAlgo_LODA	8,003.5 (+0.1)	14,563.9 (+175.1)	36,238.3 (+787.7)
Finalised Data	0.5	85.7	0.4
Active Memory Before Export	24,971.5	24,971.1	45,820.0

Notes:
Key Window Period: 1 Jan 2021 (inclusive) to 29 Jan 2021 (exclusive)
Data Dim. At Export: ATS: 3,304,063x24 | CMS: 19,604,630x24 | ECS: 459,392x24

Memory Load Requirements

Stage / Process	CMS – PowerSys (MB)	CMS – StationEquipment (MB)	CMS – StationFacilities (MB)
Initialisation	173.7	173.6	174.0
Base Data	4,700.0	4,700.0	4,700.0
Partitioned Base Data by FuncCat (All day)	8.4	0.007	3.5
FN MinMaxScaler	8,077.2 (+21.4)	Insufficient Data	7,997.2 (+16.6)
Pre-Anomaly Detection	24,950.4	Insufficient Data	24,869.9
FN anomalyAlgo_PCA *	8,085.9 (+10.0)	Insufficient Data	8,012.6 (+10.1)
FN PCA	8,077.6 / 8,077.6 (-2.7 / +0.0)	Insufficient Data	8008.7 / 8004. 2 (+1.8 / +0.0)
FN anomalyAlgo_IsoForest *	8,080.0 (+2.4)	Insufficient Data	8,017.9 (+13.7)
FN anomalyAlgo_HBOS	8,101.6 (+2.6)	Insufficient Data	8,032.0 (+21.6)
FN anomalyAlgo_CBLOF	8,104.6 (+3.0)	Insufficient Data	8,037.3 (+5.5)
FN anomalyAlgo_LODA	8,104.6 (+0.1)	Insufficient Data	8035.0 (+0.1)
Finalised Data	0.7	Insufficient Data	0.9
Active Memory Before Export	24,950.	Insufficient Data	24,869.9

Notes:
Key Window Period: 1 Jan 2021 (inclusive) to 29 Jan 2021 (exclusive)
Data Dim. At Export: ATS: 3,304,063x24 | CMS: 19,604,630x24 | ECS: 459,392x24

Memory Load Requirements

Stage / Process	CMS – CCTV-A (Mem Saver) (MB)	CMS – CCTV-B (Mem Saver) (MB)	CMS – AltRun (Mem Saver) (MB)
Initialisation	173.4	173.3	172.8
Base Data	4,700.0	4,700.0	4,700.0
Partitioned Base Data by FuncCat (All day)	**2,600**	**2,600**	697.9
FN MinMaxScaler	59,626.4 (+9,681.2)	58,632.8 (+9,360.5)	28,069.4 (+6,104.5)
Pre-Anomaly Detection	59,626.4	58,632.8	28,069.4
FN anomalyAlgo_PCA *	54,071.2 (+2,445.7)	53,184.1 (+2,369.8)	24,411.5 (+1,393.6)
FN PCA	51,888.9 / 50,186.2 (+181.7 / +0.0)	51,072.5 / 49,406.7 (+177.7 / +0.0)	23,131.0 / 22,094.6 (+68.9 / +0.0)
FN anomalyAlgo_IsoForest *	54,117.4 (+3,931.1)	53,361.7 (+3,954.8)	24,171.6 (+2,076.9)
FN anomalyAlgo_HBOS	51,179.3 (+833.8)	50,382.1 (+811.7)	22,624.1 (+440.1)
FN anomalyAlgo_CBLOF	51,396.5 (+821.4)	50,625.0 (+823.2)	22,750.7 (+429.5)
FN anomalyAlgo_LODA	51,664.7 (828.3)	50,868.2 (+807.4)	22,880.7 (+420.3)
Finalised Data	370.7	362.7	193.4
Active Memory Before Export	59,626.4	58,632.8	28,069.4

Notes:
Key Window Period: 1 Jan 2021 (inclusive) to 29 Jan 2021 (exclusive)
Data Dim. At Export: ATS: 3,304,063x24 | CMS: 19,604,630x24 | ECS: 459,392x24

Memory Load Requirements

Stage / Process	CMS – ENG ALL (Mem Saver) (MB)	CMS – ENG AltRun (Mem Saver) (MB)
Initialisation	173.5	172.8
Base Data	4,700.0	4,700.0
Partitioned Base Data by FuncCat (All day)	4,700.0	697.9
FN MinMaxScaler	36,083.8 (+4,291.3)	28,069.4 (+6,104.5)
Pre-Anomaly Detection	39,894.1	28,069.4
FN anomalyAlgo_PCA *	33,562.9 (+1,032.1)	24,411.5 (+1,393.6
FN PCA	31,947.7 / 31,885.3 (-613.1 / +0.0)	23,131.0 / 2094.6 (+68.9 / +0.0)
FN anomalyAlgo_IsoForest *	33,523.5 (+1,638.1)	24,171.6 (+2,076.9)
FN anomalyAlgo_HBOS	32,235.3 (+294.1)	2,2624.1 (+440.1)
FN anomalyAlgo_CBLOF	32,304.0 (+270.2)	22,750.7 (+429.5)
FN anomalyAlgo_LODA	32,408.7 (281.9)	22,880.7 (+420.3)
Finalised Data	127.9	193.4
Active Memory Before Export	39,894.1	28,069.4

Notes:
Key Window Period: 1 Jan 2021 (inclusive) to 29 Jan 2021 (exclusive)
Data Dim. At Export: ATS: 3,304,063x24 | CMS: 19,604,630x24 | ECS: 459,392x24