Alarm correlation network fault prediction

Network fault prediction involves identifying the root cause of device failures within a network, particularly when multiple devices report faults simultaneously. Each device in the network is equipped with monitoring capabilities that trigger alarms when specific faults are detected. These alarms are critical indicators of underlying issues, but in a large network, the sheer volume of alarms can obscure the true source of the problem. To address this, various approaches leverage machine learning algorithms and alarm correlation techniques to pinpoint the root cause efficiently. This section outlines the foundational aspects of our implementation, including the network features used, the data generation process, and the first approach employing XGBoost and association rule mining to identify root cause devices

Understanding Network Features and Feature Selection

Before delving into the methodologies, it's essential to understand the features that characterize the devices in the network and their relevance to fault prediction. The dataset comprises a comprehensive set of features that capture both the operational state and environmental conditions of each device. These features were carefully selected to provide a holistic view of device health, network hierarchy, and fault indicators, enabling accurate root cause analysis. Below is a detailed list of the features used in this study, along with their significance:

- **Timestamp**: Represents the timeline of the log entry for each data record. This feature helps allow us to track the sequence of events and identify patterns in fault occurrences over time.
- **Equipment ID**: A unique identifier for each device (e.g., device_001). This helps in tracking individual device behavior and correlating faults across the network.
- **Equipment Type**: Specifies the type of device, such as Router, Firewall, or Switch. Different device types have varying roles in the network hierarchy, which can influence their likelihood of being a root cause. For example, a router failure is more likely to affect downstream devices than a switch failure.
- **Location**: Indicates the physical location of the device (e.g., London, Tokyo). Location-based patterns can reveal environmental or regional factors contributing to faults, such as power grid issues in a specific city.
- **EquipmentAgeDays**: Measures the number of days the device has been in use. Older devices may be more prone to hardware failures, making this feature a potential predictor of faults.
- **Alarm_SpanLoss**: An alarm that triggers when there is significant signal degradation over a fiber optic link. High span loss can indicate issues like fiber degradation or misalignment, often a root cause of downstream signal issues.
- **Alarm_OpticalReturnLoss**: An alarm for excessive reflected signal in the optical link. This can point to connector issues or fiber cuts, which are critical physical layer problems.
- **Alarm_Temperature**: Detects temperature abnormalities in the device. Overheating can lead to hardware failures, making this a key indicator of potential root causes.

- **Alarm_Voltage**: Flags voltage abnormalities. Voltage fluctuations can cause power-related issues, often cascading to affect other devices in the network.
- **SpanLoss**: A numerical measure of signal strength loss over the fiber span. This complements the Alarm_SpanLoss by providing a quantitative metric for analysis.
- **OpticalReturnLoss**: Measures the reflected signal strength in the optical link, complementing Alarm_OpticalReturnLoss with a numerical value.
- **Temperature**: Records the device's operating temperature. This helps in assessing whether temperature-related alarms are justified and their impact on device performance.
- **Voltage**: Measures the device's operating voltage, providing a quantitative basis for Alarm_Voltage.
- **PowerOutage**: A binary indicator (yes/no) of whether the device is affected by a power outage. Power outages in parent devices can cause widespread faults in downstream devices.
- **FiberCut**: A binary indicator (yes/no) of whether a fiber cut has occurred. Fiber cuts are a common physical layer issue that can disrupt connectivity for multiple devices.
- **ParentDeviceID**: The identifier of the upstream device in the network hierarchy. This is critical for understanding the hierarchical relationships between devices, as a fault in a parent device can propagate to its children.
- **UpstreamDevice**: Indicates the status of the upstream device (up or down). A downed upstream device can directly cause downstream failures.
- **DownstreamDevice**: Indicates the status of downstream devices (e.g., all up, partial down). This helps quantify the impact of a device's failure on the network.
- **InterfaceStatus**: The overall status of the device's interfaces (e.g., up, down, testing). Interface issues can indicate connectivity problems that may originate from a root cause.
- **InterfaceInErrors**: Counts the number of incoming interface errors. High error rates can signal underlying issues in the network link.
- **InterfaceOutErrors**: Counts the number of outgoing interface errors, complementing InterfaceInErrors to provide a full picture of interface health.
- **CPUUtilization**: Measures the CPU usage percentage of the device. High CPU usage can lead to performance bottlenecks, potentially causing or exacerbating faults.
- **MemoryUtilization**: Measures the memory usage percentage. Similar to CPU utilization, high memory usage can indicate resource constraints that contribute to failures.
- **TemperatureStatus**: Categorizes the temperature state (e.g., normal, critical, warning). This provides a qualitative assessment of temperature-related risks.
- **FanStatus**: Indicates the health of internal fans (e.g., normal, warning). Faulty fans can lead to overheating, a common root cause of device failure.
- **PowerStatus**: Reflects the status of the power module (e.g., normal, failure). Power issues in a parent device can cascade to affect downstream devices.
- **AlarmCount**: The total number of alarms triggered by the device. A high alarm count can indicate a device under stress, potentially a root cause candidate.
- **DownstreamImpactScore**: A computed score reflecting the number of downstream devices affected by the device's failure. This is a critical feature for identifying root causes, as a high score suggests a device higher in the hierarchy with a broader impact.

- **IsRootCause**: A binary label (yes/no) indicating whether the device is the root cause of a fault. This is the target variable for prediction, determined based on predefined logic and thresholds.

These features collectively provide a multidimensional view of the network, capturing device-specific metrics, environmental conditions, hierarchical relationships, and fault indicators. Their selection ensures that both direct (e.g., alarms) and indirect (e.g., downstream impact) factors are considered in root cause analysis.

Data Generation Process

The dataset used for this analysis was synthetically generated to simulate a telecom network with 120 devices. The generation process was carried out using a Python script (data_generation.py), which created realistic device metrics and fault scenarios. Here's a detailed breakdown of the process:

1. **Device Creation**: We simulated 120 devices, each assigned a unique EquipmentID (e.g., ce-ro-2 , cpe-sw-4 etc). Devices were categorized into types (EquipmentType) such as Routers, Firewalls, and Switches, reflecting a typical network composition.
2. **Hierarchical Structure**: A network hierarchy was established by assigning ParentDeviceID values, ensuring that devices were organized in a tree-like structure with parent-child relationships (e.g., a router as a parent to multiple switches).
3. **Feature Generation**:
    - **Metrics with Weights**: Features like CPUUtilization, MemoryUtilization, Temperature, Voltage, SpanLoss, and OpticalReturnLoss were generated by assigning random values within realistic ranges, weighted to reflect typical device behavior. For example, CPUUtilization was generated with a higher probability of moderate values (20 - 100%) to mimic normal operation, with occasional spikes to simulate stress conditions.
    - **Status Indicators**: Categorical features like TemperatureStatus, FanStatus, PowerStatus, InterfaceStatus, UpstreamDevice, and DownstreamDevice were generated using random choices with predefined probabilities. For instance, TemperatureStatus was set to “normal” 80% of the time, “warning” 15%, and “critical” 5%, reflecting realistic distributions.
    - **Alarms and Faults**: Alarms (Alarm_SpanLoss, Alarm_OpticalReturnLoss, Alarm_Temperature, Alarm_Voltage) were triggered based on thresholds applied to their corresponding metrics. For example, Alarm_Temperature was set to 1 if Temperature exceeded a threshold (e.g., 40.0).
    - **Binary Faults**: Features like PowerOutage and FiberCut were generated as binary variables with a low probability of occurrence (e.g., 5%) to simulate rare but impactful events.
    - **Hierarchical Impact**: The DownstreamImpactScore was computed based on the number of downstream devices affected by a fault, weighted by the device's position in the hierarchy. A router failure, for instance, would have a higher impact score than a switch failure.
4. **Root Cause Labeling**: The IsRootCause column was generated using a logic-based approach. A device was labeled as a root cause if it met certain conditions, such as:
    - High AlarmCount (e.g., >5 alarms).

    - Temperature status not normal and interface errors >250
    - Cpu utilization or memory utilization is high
    - Fan status or interface status down.

This synthetic dataset mimics the complexity of a real telecom network, providing a controlled environment to test fault prediction approaches while ensuring that the data reflects realistic fault scenarios and hierarchical dependencies.

Approach 1: Root Cause Prediction Using XGBoost and Alarm Correlation with Association Rule Mining

In a telecom network with 100 devices organized in a hierarchical structure, a fault affecting all devices is often traceable to a single root cause—typically a parent device whose failure cascades downstream. For example, a power outage in a parent router can trigger alarms in all connected switches and endpoints, resulting in 100 simultaneous error reports. Identifying the root cause amidst this flood of alarms is a critical challenge for network reliability and efficient troubleshooting. The goal of this approach is to:

- Identify the root cause device(s) responsible for widespread faults.
- Uncover patterns in alarms associated with root cause devices to improve future fault detection.
- Reduce the noise from correlated alarms, focusing on actionable insights.

Methodology

This approach combines a machine learning algorithm (XGBoost) for root cause prediction with association rule mining to analyze alarm correlations among root cause devices. Here's a detailed breakdown of the process:

1. **Root Cause Prediction with XGBoost**:
    - **Why XGBoost?** XGBoost is a powerful gradient-boosting algorithm widely used for classification tasks, particularly with tabular data. It excels in handling imbalanced datasets (as IsRootCause is likely imbalanced, with fewer root cause devices) and provides feature importance scores, which help understand the drivers of faults. Its ability to capture complex, non-linear relationships between features makes it ideal for this task.
    - **Data Preparation**: The dataset was preprocessed by encoding categorical variables (e.g., EquipmentType, Location) using one-hot encoding and scaling numerical features (e.g., CPUUtilization, DownstreamImpactScore) with MinMaxScaler to normalize their values. Irrelevant columns like Timestamp, EquipmentID, and ParentDeviceID were excluded from the prediction task but retained for later temporal analysis.
    - **Model Training**: The dataset was split into training (80%) and testing (20%) sets. XGBoost was trained on the training set to predict IsRootCause, using features like alarms, status indicators, and impact scores.The model was evaluated on the test set, yielding an accuracy of 96%. The classification report showed:
    - Precision for class 0 (non-root cause): 0.99
    - Precision for class 1 (root cause): 0.88

    - Recall for class 1: 0.95
    - F1-score for class 1: 0.92
    - Confusion Matrix: [[155 5], [2 38]], indicating only 7 misclassifications out of 200 test samples.
    - **Purpose**: The trained model analyzes historical data to identify devices likely to be root causes of faults. For example, a router with a PowerOutage and high DownstreamImpactScore might be flagged as a root cause, as its failure could explain errors in downstream devices.

2. **Alarm Correlation with Association Rule Mining**:
    - **Why Association Rule Mining?** Once root cause devices are identified, understanding the patterns in their alarms is crucial for future fault prediction. Association rule mining, implemented using the Apriori algorithm, identifies frequent co-occurrences of alarms (e.g., Alarm_Voltage → Alarm_SpanLoss), revealing relationships that can be used to trace faults back to their source.
    - **Process**:
        - **Data Selection**: Focus on devices labeled as root causes (either predicted by XGBoost or actual IsRootCause = 1). Extract alarm-related features (Alarm_SpanLoss, Alarm_Voltage, etc.) for these devices.
        - **Apriori Algorithm**: Convert alarm data into a boolean format (e.g., alarm triggered = 1, not triggered = 0). Apply the Apriori algorithm with a minimum support threshold (e.g., 0.05) to identify frequent alarm combinations, followed by generating association rules with a minimum confidence threshold (e.g., 0.5).
        - **Example Rule**: A rule like Alarm_Voltage → Alarm_SpanLoss with high confidence (e.g., 0.9) indicates that when a voltage alarm is triggered in a root cause device, a span loss alarm is likely to follow, suggesting a causal relationship.
    - **Purpose**: These rules help in understanding alarm patterns specific to root cause devices. For instance, if a future device exhibits both Alarm_Voltage and Alarm_SpanLoss, it can be quickly identified as a potential root cause based on historical patterns, streamlining troubleshooting.

Temporal Analysis: Sequence Analysis

Temporal analysis was conducted to explore the sequence of alarms in root cause devices.

- **Process**:
    - The dataset was grouped by EquipmentID, and Timestamp was converted to a datetime format for chronological sorting.
    - Focused on devices with IsRootCause == 1, extracting their alarm columns and timestamps.
    - A function (check_sequence) was defined to check if Alarm_Voltage precedes Alarm_SpanLoss in the timeline for each device. For each device, the

timestamps of Alarm_Voltage and Alarm_SpanLoss were compared; if a span loss alarm occurred after a voltage alarm, the pair was marked as causal.

- **Result**: In 21.3% of cases, Alarm_Voltage preceded Alarm_SpanLoss, suggesting a potential temporal relationship where voltage issues may lead to signal degradation.
- **Purpose**: This analysis provides a temporal clue for root cause tracing. In a network hierarchy, a parent device's voltage failure might occur first, followed by downstream effects like span loss, helping engineers prioritize voltage-related checks.

6. Granger Causality Test for Causal Inference

The Granger Causality Test was applied to statistically test the causal relationship between Alarm_Voltage and Alarm_SpanLoss.

- **Why Granger Causality Test?** The Granger Causality Test is a statistical method used to determine if one time series can predict another. In this context, it tests whether Alarm_Voltage can predict Alarm_SpanLoss, providing a more rigorous assessment of causality beyond simple temporal precedence.
- **Process**:
  - **Data Selection**: Focused on a single EquipmentID with sufficient observations for time-series analysis. The dataset was filtered to include Timestamp, Alarm_Voltage, and Alarm_SpanLoss for the selected device.
  - **Observation Check**: The number of observations per EquipmentID was checked. The test requires at least 10 observations for a maxlag of 3 (to examine up to 3 time steps in the past). However, the maximum observations per device was 9, so the maxlag was reduced to 1.
  - **Data Preparation**: The data was converted into a time-series format by setting Timestamp as the index, and alarm columns were cast to integers (0 or 1).
  - **Granger Causality Test**: The test was run using grangercausalitytests from statsmodels, with Alarm_SpanLoss as the dependent variable and Alarm_Voltage as the predictor. The test evaluates whether past values of Alarm_Voltage improve the prediction of Alarm_SpanLoss beyond using only past values of Alarm_SpanLoss.
- **Result**:
  - With maxlag=1, the test yielded a p-value of 0.6462 (F-test: F=0.2381, df_denom=5, df_num=1). Since the p-value is greater than 0.05, there is no statistically significant evidence that Alarm_Voltage Granger-causes Alarm_SpanLoss.
  - Other test statistics (chi2, likelihood ratio) also confirmed this finding (p-values > 0.5).
- **Interpretation**:
  - The lack of statistical significance suggests that Alarm_Voltage does not Granger-cause Alarm_SpanLoss in this dataset. This could be due to the limited number of observations (9 per device), which reduces the test's power to detect causality.
  - Despite the statistical result, the temporal sequence analysis (21.3% precedence) still provides practical evidence of a potential relationship, which may be validated with more data.

- **Purpose**: The Granger Causality Test adds a statistical layer to the temporal analysis, aiming to confirm whether the observed sequence of alarms reflects a causal relationship. While the test was inconclusive here, it highlights the need for more data to perform robust causal inference

The network fault prediction approach implemented for identifying root cause devices in a telecom network proved to be highly effective, achieving a prediction accuracy of 96% using the XGBoost classifier. By integrating multiple techniques XGBoost for root cause prediction, association rule mining for alarm correlation, temporal sequence analysis, and the Granger Causality Test for causal inference this methodology provided a robust framework for diagnosing network faults.

Approach 2 : Clustering

The clustering-based approach was chosen to address the problem by grouping devices into distinct profiles based on their operational characteristics, alarm patterns, and fault behaviors.This approach is suitable for

- **Behavioral Segmentation**: Clustering groups devices with similar performance metrics (e.g., CPU utilization, alarm counts) and fault patterns (e.g., frequency of root cause occurrences). This segmentation helps identify clusters of "high-risk" devices prone to being root causes, enabling targeted monitoring.
- **Unsupervised Learning Advantage**: Since labeled data (IsRootCause) may not always be available in real-world scenarios, clustering provides an unsupervised method to discover patterns without requiring labeled training data.
- **Comprehensive Device Profiling**: By aggregating metrics at the device level (e.g., mean CPU utilization, total alarms), clustering captures holistic device behavior, revealing underlying structures that may correlate with fault likelihood.
- **Enhanced Interpretability**: Visualizations like PCA scatter plots and radar charts make it easier to interpret clusters, helping engineers understand device behavior and prioritize maintenance.
- **Complementary to Predictive Models**: Clustering can complement the previous XGBoost-based approach by identifying device groups for further predictive analysis, improving fault detection strategies.

This approach leverages multiple clustering techniques (K-Means, Hierarchical, DBSCAN, and conceptually Gaussian Mixture Models) to ensure robust grouping and uses extensive EDA and visualizations to validate and interpret the results.

Goal

The primary goal is to enhance network fault management by:

1. Grouping devices into clusters based on operational and fault-related characteristics.
2. Identifying clusters with a high proportion of root cause devices to narrow down troubleshooting efforts in a multiple device network.
3. Providing insights into device behavior (e.g., alarm patterns, performance trends) for proactive maintenance and efficient fault resolution.

Methodology

The approach is structured into several key phases: Enhanced Exploratory Data Analysis (EDA), Feature Engineering for Device Profiling, Clustering, and Visualization. Below is a detailed breakdown of each phase.

1. Enhanced Exploratory Data Analysis (EDA)

Purpose: To understand the dataset's structure, distributions, and relationships before clustering.

- **Data Cleaning**:
    - Converted Timestamp to datetime for time-series analysis.
    - Ensured PowerOutage and FiberCut were integers (binary flags: 0 or 1).
    - Identified numerical, categorical, and binary columns for analysis:
        - Numerical: EquipmentAgeDays, SpanLoss, OpticalReturnLoss, Temperature, Voltage, interface_in_errors, interface_out_errors, cpu_utilization, memory_utilization, alarms_count, downstream_impact_score.
        - Categorical: EquipmentType, Location, upstream_status, downstream_status, interface_status, temperature_status, fan_status, power_status.
        - Binary: IsRootCause, Alarm_SpanLoss, Alarm_OpticalReturnLoss, Alarm_Temperature, Alarm_Voltage, PowerOutage, FiberCut.
- **Univariate Analysis**:
    - Plotted histograms with KDE for numerical features to assess distributions (e.g., cpu_utilization showed a right-skewed distribution).
    - Used box plots to identify outliers (e.g., alarms_count had significant outliers, indicating some devices experience frequent alarms).
    - Visualized categorical features with count plots (e.g., EquipmentType distribution showed a mix of Routers, Switches, etc.).
    - Plotted binary flags as bar charts (e.g., IsRootCause showed an imbalance, with fewer root cause instances).
- **Bivariate and Multivariate Analysis**:
    - Created a correlation heatmap for numerical and binary features, revealing relationships like a moderate positive correlation between alarms_count and downstream_impact_score (indicating devices with more alarms cause larger downstream impacts).
    - Used box plots to compare numerical features across categorical variables (e.g., cpu_utilization by EquipmentType showed Routers with higher CPU usage).
    - Generated pair plots for a subset of numerical features (EquipmentAgeDays, cpu_utilization, alarms_count) to explore pairwise relationships.
- **Time-Series Analysis**:
    - Plotted total alarms_count over time, identifying spikes that may correspond to network-wide incidents.
    - Visualized cpu_utilization over time for the top 3 devices by snapshot count, revealing temporal patterns (e.g., periodic spikes in CPU usage).

Insights:

- Devices vary significantly in age, performance metrics, and alarm frequencies, suggesting clustering could reveal meaningful groups.
- High correlations between certain metrics (e.g., alarms_count and downstream_impact_score) indicate potential features for clustering.
- Temporal patterns suggest some devices experience recurring issues, which clustering can help isolate.

2. Feature Engineering for Device Profiling

Purpose: To create a device-level dataset suitable for clustering by aggregating metrics for each EquipmentID.

- **Aggregation**:
    - Grouped the dataset by EquipmentID, resulting in 120 unique device profiles.
    - For each device, computed:
        - **Static/Mode Features**: Mode of categorical features (e.g., EquipmentType, Location, upstream_status).
        - **Numerical Aggregates**: Mean, standard deviation, maximum, and median of numerical metrics (e.g., mean_cpu_utilization, std_alarms_count).
        - **Alarm and Event Aggregates**: Sum and mean of alarm flags (e.g., sum_Alarm_Voltage, mean_Alarm_SpanLoss) and binary events (sum_PowerOutage).
        - **Fault Metrics**: Sum and rate of IsRootCause (sum_IsRootCause, rate_IsRootCause).
        - **Total Snapshots**: Number of observations per device (total_snapshots).
    - Resulted in a device_profiles_df with 120 rows and 63 columns.
- **Preprocessing for Clustering**:
    - Dropped EquipmentID from the feature set (X_profile) to focus on behavioral metrics.
    - Separated numerical and categorical columns:
        - Numerical: EquipmentAgeDays_Max, mean_cpu_utilization, std_alarms_count, etc.
        - Categorical: EquipmentType, Location, mode_upstream_status, etc.
    - Applied a ColumnTransformer with:
        - Numerical pipeline: Imputed missing values with the median and applied StandardScaler for standardization.
        - Categorical pipeline: Imputed missing values with the most frequent category and applied OneHotEncoder to convert categories into binary features.
    - Output: X_profile_processed with 120 rows and 74 columns (after one-hot encoding).

Outcome: A processed dataset capturing comprehensive device profiles, ready for clustering.

3. Enhanced Clustering Methods

Purpose: To group devices into clusters based on their profiles, identifying patterns that may correlate with root cause behavior.

- **K-Means Clustering**:
  - Tested K values from 2 to 10 (limited by the number of samples, 120).
  - Evaluated using:
    - Inertia (Elbow Method): To identify a "knee" in the inertia curve.
    - Silhouette Score: To measure cluster cohesion and separation (higher is better).
    - Davies-Bouldin Index: To assess cluster compactness and separation (lower is better).
    - Calinski-Harabasz Index: To evaluate cluster variance ratio (higher is better).
  - Optimal K was 2 (based on the highest silhouette score), resulting in clusters of 68 and 52 devices.
  - Applied K-Means with K=2, assigning cluster labels to device_profiles_df['KMeans_Cluster'].
- **Hierarchical Clustering (Agglomerative)**:
  - Generated a dendrogram using Ward's linkage to visualize the hierarchical structure.
  - Applied Agglomerative Clustering with K=2 (for consistency with K-Means), resulting in clusters of 71 and 49 devices.
  - Assigned cluster labels to device_profiles_df['Agg_Cluster'].
- **DBSCAN**:
  - Used heuristic parameters (eps = 0.5 * number of features, min_samples = 5% of samples), noting that tuning is needed.
  - DBSCAN identified 1 cluster with 120 devices and 0 noise points, indicating the parameters were too lenient (eps too large).
  - Assigned cluster labels to device_profiles_df['DBSCAN_Cluster'].
- **Gaussian Mixture Models (GMM) - Conceptual**:
  - Noted that GMM could be applied to find probabilistic clusters, using BIC/AIC to select the optimal number of components.
  - Skipped full implementation due to complexity but highlighted its potential for capturing complex cluster shapes.

Insights:

- K-Means and Hierarchical Clustering both suggested 2 main clusters, indicating a natural split in device behavior (e.g., high-risk vs. stable devices).
- DBSCAN’s failure to identify multiple clusters underscores the need for parameter tuning (e.g., using a k-distance graph to select eps).

4. Cluster Validation and Interpretation

Purpose: To validate the quality of clusters and interpret their characteristics.

- **Validation Metrics**:

    - Silhouette Score for K-Means (optimal K=2) was the highest among tested K values, indicating good cluster separation.
    - Davies-Bouldin and Calinski-Harabasz scores supported the choice of K=2, showing compact and well-separated clusters.
- **Interpretation**:
    - **Cluster 0 (52 devices)**: Likely "stable" devices with lower rate_IsRootCause, fewer alarms, and moderate CPU utilization.
    - **Cluster 1 (68 devices)**: Likely "high-risk" devices with higher rate_IsRootCause, more frequent alarms, and higher CPU utilization.
    - Used violin plots to compare features across clusters (e.g., mean_alarms_count was significantly higher in Cluster 1).

5. Enhanced Visualizations for Clustering (Focused on K-Means)

Purpose: To visualize and interpret the K-Means clusters for actionable insights.

- **PCA Scatter Plot**:
    - Reduced X_profile_processed to 2D using PCA and plotted devices, colored by K-Means cluster.
    - Showed clear separation between Cluster 0 and Cluster 1, validating the clustering quality.
- **Box/Violin Plots**:
    - Compared features like mean_cpu_utilization, mean_alarms_count, and rate_IsRootCause across clusters.
    - Confirmed that Cluster 1 devices have higher alarm counts and root cause rates, making them candidates for closer monitoring.
- **Radar Chart**:
    - Plotted normalized centroids of K-Means clusters for features like EquipmentAgeDays_Max, mean_cpu_utilization, and mean_alarms_count.
    - Highlighted that Cluster 1 has a larger "footprint" in fault-related metrics, reinforcing its high-risk nature.

Conclusion

The clustering-based approach successfully grouped 120 devices into meaningful profiles, identifying two primary clusters: Cluster 0 (52 devices, stable) and Cluster 1 (68 devices, high-risk). With K-Means achieving an optimal K of 2 (based on silhouette score), the approach effectively narrowed the troubleshooting scope in a 100-device network by focusing on Cluster 1 devices, which exhibited higher rate_IsRootCause (e.g., 0.44 vs. 0.25 in Cluster 0), more frequent alarms (mean_alarms_count higher in Cluster 1), and greater downstream impact. Visualizations like PCA scatter plots and radar charts provided actionable insights, confirming Cluster 1 as the priority for fault investigation. Hierarchical Clustering aligned with K-Means (K=2), while DBSCAN highlighted the need for parameter tuning. This method reduced the troubleshooting scope by 32% (from 100 to 68 devices) and enabled further prioritization, minimizing network downtime. Clustering offered a complementary perspective by focusing on device behavior rather than alarm causality, making it ideal for proactive monitoring.

Approach3 : Ensemble Fault Detection Approach

The ensemble fault detection approach integrates multiple methods like rule-based detection, graph-based propagation analysis, and Gaussian Mixture Model (GMM)-based anomaly detection—to achieve a robust and comprehensive fault identification strategy. Unlike the previous clustering-based approach (which focused on grouping devices by behavior) and the Granger Causality approach (which emphasized causal relationships between alarms), this method combines diverse perspectives to improve accuracy and reliability in identifying critical devices. Here's why this approach is suitable:

- **Multi-Method Integration**: By combining rule-based, graph-based, and GMM-based methods, the ensemble approach leverages the strengths of each:
    - **Rule-Based Detection**: Captures domain-specific knowledge through predefined thresholds (e.g., high fault severity, downstream impact), ensuring interpretable and actionable results.
    - **Graph-Based Detection**: Models the hierarchical network structure, propagating fault severity from child to parent devices to identify upstream root causes.
    - **GMM-Based Anomaly Detection**: Identifies statistical outliers in operational metrics (e.g., SpanLoss, Temperature), capturing anomalies that may not be caught by rules or graphs.
- **Improved Precision and Recall**: Each method has its limitations (e.g., rule-based may miss subtle anomalies, GMM may overfit to noise). Ensemble scoring mitigates these by requiring consensus across methods, reducing false positives while capturing diverse fault patterns.
- **Network Context Awareness**: The graph-based component explicitly models device relationships (via ParentDeviceID), addressing the hierarchical nature of the telecom network where faults propagate downstream.
- **Actionable Insights with Visualizations**: The approach includes extensive visualizations (e.g., fault propagation graphs, anomaly scatter plots), making it easier for engineers to interpret results and prioritize devices for investigation.

This approach aims to balance sensitivity (detecting true root causes) and specificity (avoiding false positives), making it ideal for real-world fault management where both missed faults and unnecessary investigations are costly.

Goal

The primary goal is to enhance network fault management by:

1. Identifying critical devices likely to be root causes of faults using an ensemble of rule-based, graph-based, and GMM-based methods.
2. Reducing the troubleshooting scope in a 100-device network by prioritizing devices with high ensemble fault scores.
3. Providing interpretable insights into fault patterns (e.g., severity, propagation, anomalies) through visualizations and reports for proactive maintenance and efficient fault resolution.

Methodology

The approach is structured into several key phases: Data Preprocessing, Feature Engineering, Individual Fault Detection Methods, Ensemble Integration, and Visualization. Below is a detailed breakdown of each phase.

1. Data Preprocessing

Purpose: To clean and prepare the dataset for analysis.

- **Loading and Initial Exploration**:
  - Loaded the dataset (telecom_merge1.csv) into a Pandas DataFrame with 1000 entries.
  - Displayed basic information (e.g., column types) and checked for missing values using df.info() and df.isnull().sum().
  - Previewed the first 5 rows to understand the data structure.
- **Data Cleaning**:
  - Converted Timestamp to datetime for time-series analysis.
  - Encoded binary columns (FiberCut, PowerOutage) using LabelEncoder to ensure they are numeric (0 or 1).
  - Handled missing values:
    - Numerical columns (e.g., SpanLoss, Voltage): Filled with the median.
    - Categorical columns (e.g., EquipmentType, Location): Filled with the mode.
  - Encoded categorical columns (EquipmentType, Location, upstream_status, downstream_status, interface_status) using LabelEncoder for numerical processing.
- **Feature Selection**:
  - Selected relevant columns for analysis, including Timestamp, EquipmentID, EquipmentType, Location, ParentDeviceID, SpanLoss, OpticalReturnLoss, Temperature, Voltage, downstream_impact_score, alarms_count, IsRootCause, FaultInjected, and various alarm and status flags.

Outcome: A cleaned and encoded dataset ready for feature engineering and fault detection.

2. Feature Engineering

Purpose: To create features that enhance fault detection by capturing device behavior and fault likelihood.

- **Normalization**:
  - Normalized numerical features (SpanLoss, OpticalReturnLoss, Temperature, Voltage, alarms_count) using StandardScaler to ensure they are on the same scale for GMM-based anomaly detection.
- **Fault Severity Calculation**:
  - Defined a calculate_fault_severity function to compute a composite FaultSeverity score based on weighted contributions from normalized features:
    - SpanLoss (if > 2): 0.5 * value

        - OpticalReturnLoss (if < -2): -0.5 * value
        - Temperature (if > 2): 0.5 * value
        - Voltage (if < -2): -0.5 * value
        - alarms_count: 0.5 * value
    - Applied this to create a FaultSeverity column, with values ranging from -0.752 to 2.482 (mean: 0.110, std: 0.655).
- **Degraded Status Indicator**:
    - Created a binary DegradedStatus column (0 or 1) to flag devices with degraded network conditions based on upstream_status, downstream_status, or interface_status (e.g., "down", "degraded", "all_down"). 80.6% of devices were flagged as degraded.
- **Root Cause Flag**:
    - Created a RootCauseFlag column (0 or 1) by combining IsRootCause and FaultInjected (1 if either is 1). 29.4% of entries were flagged as potential root causes.
- **Combined Fault Label**:
    - Created a FaultLabel column (0 or 1) to indicate the presence of a fiber cut or power outage (FiberCut or PowerOutage = 1). Only 2.5% of entries had a fault label of 1.

Outcome: Enhanced dataset with new features (FaultSeverity, DegradedStatus, RootCauseFlag, FaultLabel) that quantify fault severity and likelihood, providing a foundation for fault detection methods.

3. Individual Fault Detection Methods

Purpose: To apply three distinct methods to identify critical devices, each capturing different aspects of faults.

- **Rule-Based Fault Detection**:
    - Defined thresholds for key metrics: FaultSeverity > 2.0, downstream_impact_score > 0.7, alarms_count > 1.5.
    - Flagged devices based on conditions like FiberCut, PowerOutage, alarm flags (Alarm_SpanLoss, etc.), and threshold exceedances.
    - Generated a report (rule_report) with 758 critical devices, capturing faults like "Temperature Alarm" or "High Downstream Impact."
    - **Insight**: This method is sensitive to predefined rules, capturing a large number of potential issues but may include false positives.
- **Graph-Based Fault Detection**:
    - Built a directed graph using networkx, with nodes representing devices (EquipmentID) and edges representing parent-child relationships (ParentDeviceID → EquipmentID).
    - Propagated fault severity from child to parent nodes, scaling child severity by 0.8 to reflect diminishing impact upstream.
    - Flagged devices as critical if FaultSeverity > 2.0 or FaultLabel = 1, resulting in a report (graph_report) with 4 critical devices.
    - **Insight**: This method excels at capturing hierarchical fault propagation but is conservative, identifying fewer critical devices.

- **GMM-Based Anomaly Detection**:
    - Applied a Gaussian Mixture Model (GMM) with 2 components to features (SpanLoss, OpticalReturnLoss, Temperature, Voltage, alarms_count).
    - Calculated anomaly scores and flagged devices with scores below a threshold (mean - 2 * std), resulting in a report (gmm_report) with 2 anomalous devices.
    - **Insight**: GMM effectively identifies statistical outliers but is highly selective, potentially missing some faults.

Outcome: Three complementary reports identifying critical devices from different perspectives: rule-based (broad coverage), graph-based (network-aware), and GMM-based (anomaly-focused).

4. Ensemble Fault Detection

Purpose: To integrate the three methods into a unified framework for more reliable fault detection.

- **Merging Reports**:
    - Combined rule_report, graph_report, and gmm_report with the original DataFrame, aligning on EquipmentID.
    - Renamed fault scores (RuleScore, GraphScore, GMMScore) and filled missing scores with 0.
- **Ensemble Scoring**:
    - Calculated an EnsembleScore as the sum of RuleScore, GraphScore, and GMMScore (each 1 if the device was flagged, 0 otherwise).
    - Combined root cause flags from all methods, labeling a device as a root cause ("Yes") if any method flagged it as such.
- **Critical Device Selection**:
    - Flagged devices as critical if EnsembleScore ≥ 2 or FaultLabel = 1, resulting in a final ensemble_report with 468 critical devices.
    - Included 188 devices with FaultLabel = 1 (fiber cut or power outage).

Outcome: A consolidated report that balances sensitivity and specificity, identifying 468 critical devices (47% of the dataset), with 188 having confirmed faults (FaultLabel = 1).

5. Visualizations

Purpose: To provide interpretable insights into fault patterns and critical devices.

- **Fault Propagation Graph**:
    - Visualized the network graph with critical devices in red, showing fault propagation paths.
- **Feature Distribution Plots**:
    - Plotted histograms for GMM features, revealing distributions (e.g., SpanLoss likely skewed).
- **Label Distribution Plot**:
    - Showed the prevalence of FiberCut, PowerOutage, and FaultLabel, highlighting the rarity of confirmed faults (2.5%).

- **GMM Anomaly Scatter Plot**:
  - Plotted SpanLoss vs. Temperature, highlighting 2 anomalies in red.
- **Ensemble Score Distribution**:
  - Showed the distribution of EnsembleScore, with most critical devices having scores of 2.
- **Fault Severity vs. Downstream Impact**:
  - Scatter plot showed a weak correlation, with FaultLabel = 1 devices scattered across severity levels.
- **Time-Series Fault Plot**:
  - Plotted FaultLabel counts over time, identifying temporal fault patterns.

Outcome: Visualizations provided actionable insights, such as identifying critical devices in the network graph and confirming GMM anomalies as outliers.

6. Practical Application in a 100-Device Network

Scenario: All 100 devices report errors due to a parent device failure.

- **Without This Approach**: Engineers investigate all 100 devices, a time-consuming process.
- **With This Approach**:
  1. **Identify Critical Devices**: Focus on the 468 devices in ensemble_report (though this spans the full dataset, in a 100-device subset, the proportion would be similar, e.g., ~47 devices).
  2. **Prioritize High Ensemble Scores**: Within the critical set, prioritize devices with EnsembleScore ≥ 2 (e.g., cpe-sw-22 with multiple flags).
  3. **Focus on Confirmed Faults**: Further narrow to devices with FaultLabel = 1 (188 devices, or ~19 in a 100-device subset).
  4. **Resolve Fault**: Fix the identified root cause (e.g., a router with a fiber cut), resolving errors in downstream devices.
- **Outcome**: Reduces troubleshooting scope from 100 to ~47 devices (or fewer with FaultLabel prioritization), significantly reducing downtime.

Conclusion

The ensemble fault detection approach successfully identified 468 critical devices out of 1000 entries, with 188 having confirmed faults (FaultLabel = 1). By integrating rule-based (758 devices), graph-based (4 devices), and GMM-based (2 devices) methods, the ensemble method achieved a balanced detection rate, reducing the troubleshooting scope by 53% (from 1000 to 468 devices, or ~47 in a 100-device network). Key insights include:

- **Rule-Based Detection** was the most sensitive, flagging 758 devices but potentially including false positives.
- **Graph-Based Detection** was conservative, identifying 4 devices by leveraging network hierarchy, ensuring focus on upstream root causes.
- **GMM-Based Detection** was highly selective, flagging 2 anomalies, capturing statistical outliers missed by other methods.
- **Ensemble Scoring** provided a robust consensus, prioritizing devices with multiple fault indicators (EnsembleScore ≥ 2) and confirmed faults (FaultLabel = 1).

Visualizations like the fault propagation graph and GMM anomaly scatter plot enhanced interpretability, enabling engineers to focus on critical devices like cpe-sw-22 (high ensemble score) and core-sw-8 (high fault severity via graph). Compared to the clustering approach (which grouped devices by behavior) and Granger Causality approach (which modeled alarm causality), the ensemble method offered a more actionable framework by directly flagging critical devices with a multi-faceted approach. It reduced downtime by narrowing the troubleshooting scope and provided a foundation for proactive maintenance as of May 13, 2025. Future improvements could include fine-tuning GMM components using BIC/AIC, optimizing rule-based thresholds with domain expertise, and incorporating real-time data for dynamic fault detection.