**Documentation: Outlier Detection and Analysis**

### **1. Overview of the Dataset**

The dataset under analysis consists of a mix of categorical and numerical features. Specifically:

**Categorical Variables:**

* category\_id: Represents the classification or group to which a case belongs (e.g., product type, process step).
* market\_id: Indicates the geographic or business market for the case.
* segment: Defines the customer or business segment (e.g., Retail vs. Wholesale).

**Numerical Variables:**

* event\_time: Continuous variable indicating the time taken to complete an event or case.
* frequency: Discrete numerical variable representing the number of occurrences for a case.

Additionally, each row includes a case\_id as a unique identifier.

### **2. Why Global Univariate Outlier Detection Might Be Insufficient**

Global univariate outlier detection treats the entire dataset as a single population. This approach assumes uniformity across all categorical segments and does not consider the contextual nuances of specific groups.

**Why This Is a Problem:**

* Business processes can behave differently across markets or segments.
* The same value might be normal in one market and anomalous in another.
* Global analysis can incorrectly flag normal data as outliers (false positives) or miss contextual anomalies (false negatives).

**Example:** An event\_time of 60 minutes:

* Could be a normal average in market Z.
* Could be a significant delay in market X, which typically completes in 20 minutes.

Thus, **contextual outlier detection** — by segmenting data using relevant categorical features — becomes essential for meaningful insights.

### **3. Role of the Kruskal-Wallis Test**

To determine which categorical features are statistically associated with the numerical features, we use the **Kruskal-Wallis H-Test**. This non-parametric test checks whether the distributions of a numerical variable differ significantly between two or more groups.

**Purpose:**

* Avoid arbitrarily including categorical variables in multivariate outlier detection.
* Only include those that significantly impact numerical behavior.

#### Hypotheses:

* **Null Hypothesis (H0):** The numerical variable has the same distribution across all categories (independent).
* **Alternative Hypothesis (H1):** At least one category has a different distribution (dependent).

**If p-value < 0.05:** Reject H0, indicating dependency. **If p-value ≥ 0.05:** Fail to reject H0, indicating independence.

This process helps in narrowing down relevant features for modeling and enhances the precision of outlier detection.

### **4. Label Encoding for Categorical Variables**

Since models like Isolation Forest require numerical input, we must encode categorical variables. We use **Label Encoding** because:

* It converts categories into integers in a memory-efficient and compact way.
* Isolation Forest does not rely on distances, so the ordinal values don’t distort the model.

**Why not One-Hot Encoding?**

* It increases dimensionality, which can reduce model performance and introduce sparsity.
* Not necessary for models like Isolation Forest which split on feature thresholds.

### **5. Why Isolation Forest for Outlier Detection?**

**Isolation Forest** is particularly well-suited for detecting anomalies in mixed-type and high-dimensional datasets due to the following reasons:

* **Isolation over distance:** Instead of using distance-based metrics (which fail with label-encoded data), it isolates anomalies by recursive random splits.
* **Scalability:** Linear time complexity with respect to the number of records makes it scalable to large datasets.
* **No distribution assumptions:** Unlike statistical models, it doesn’t require normality or linearity in data.
* **Effectiveness with encoded categories:** Works well even when categories are represented as integers, avoiding issues seen in Mahalanobis Distance or k-NN.

### **6. Interpreting and Visualizing Results**

#### a. **Univariate Global Analysis**

* Visualized using a boxplot of event\_time for the entire dataset.
* Detects outliers relative to the global distribution.
* Ignores market- or segment-specific patterns.

#### b. **Filtered Univariate Analysis**

* Same boxplot, but filtered for a specific categorical value (e.g., market\_id = 'X').
* More precise and relevant for that subset.

#### c. **Multivariate Analysis**

* Uses a scatter plot of event\_time vs. frequency, colored by outlier prediction.
* Hovering reveals details like case\_id, market\_id, etc.
* Captures interactions between features and identifies contextual anomalies.

### **7. Choosing Variables for Outlier Detection**

#### When to Use Univariate Analysis:

* When only one feature is of interest.
* Easier to interpret, visualize, and explain.
* Use separate plots for different features if they are uncorrelated.

#### When to Use Multivariate Analysis:

* When you suspect that outliers depend on the combination of variables.
* Allows detection of complex patterns (e.g., high frequency + high duration).

#### Should We Plot Frequency vs. Time?

* **Yes, if:** There’s potential interaction or correlation between them.
* **No, if:** They are independent. In that case, analyze them separately using individual univariate models.

Use correlation tests or domain knowledge to decide.

### **8. Correlation Test Between Numerical Variables**

To determine whether two numerical features like event\_time and frequency are correlated, we can apply several methods:

#### **Pearson Correlation Coefficient**

Assesses linear relationships.

from scipy.stats import pearsonr  
  
corr, p\_value = pearsonr(data['event\_time'], data['frequency'])  
print(f"Pearson Correlation Coefficient: {corr:.2f}, p-value: {p\_value:.4f}")  
if p\_value < 0.05:  
 print("=> Statistically significant linear correlation.")  
else:  
 print("=> No statistically significant linear correlation.")

#### **Spearman Rank Correlation**

Assesses monotonic relationships (useful for non-linear).

from scipy.stats import spearmanr  
  
corr, p\_value = spearmanr(data['event\_time'], data['frequency'])  
print(f"Spearman Correlation Coefficient: {corr:.2f}, p-value: {p\_value:.4f}")  
if p\_value < 0.05:  
 print("=> Statistically significant monotonic correlation.")  
else:  
 print("=> No statistically significant monotonic correlation.")

#### **Visualization**

import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.jointplot(x='event\_time', y='frequency', data=data, kind='reg')  
plt.suptitle('Scatter Plot with Regression Line')  
plt.show()  
  
sns.heatmap(data[['event\_time', 'frequency']].corr(), annot=True, cmap='coolwarm', fmt='.2f')  
plt.title('Correlation Heatmap')  
plt.show()

#### **Interpretation:**

* Coefficient near ±1: strong correlation.
* Coefficient near 0: weak/no correlation.
* Always pair numerical correlation with visual plots to uncover hidden relationships.

Use these insights to decide if both variables should be used together or modeled separately.

### **9. Summary**

* Contextual understanding is crucial for detecting meaningful outliers.
* Use the Kruskal-Wallis test to check feature relevance before modeling.
* Apply label encoding for compatibility with Isolation Forest.
* Use global, filtered, and multivariate approaches depending on the business question.
* Use correlation tests (Pearson, Spearman) to evaluate numerical relationships.
* Visualizations help validate and communicate findings effectively.
* Isolation Forest offers a robust, flexible solution for mixed-type anomaly detection.