Assignment

Q1. What is the role of feature selection in anomaly detection?

Ans:Feature selection plays a crucial role in anomaly detection by influencing the quality and effectiveness of the detection process. The key roles of feature selection in anomaly detection include:

1. Dimensionality Reduction:

- Anomaly detection is often challenged by datasets with a large number of features or dimensions. Many features may be irrelevant, redundant, or contribute noise, making it harder to distinguish between normal and anomalous instances. Feature selection helps reduce the dimensionality of the dataset by selecting a subset of relevant features, simplifying the analysis and improving computational efficiency.

2. Improved Model Performance:

- Selecting the most informative and relevant features contributes to the creation of more accurate anomaly detection models. By focusing on features that truly characterize normal and anomalous behavior, the model is better equipped to identify patterns and deviations.

3. Enhanced Interpretability:

- A reduced set of features facilitates a clearer understanding of the factors contributing to anomalies. Interpretable models are valuable for domain experts who need insights into why certain instances are flagged as anomalies. Selecting meaningful features can help in building models that are more interpretable and actionable.

4. Mitigating the Curse of Dimensionality:

- The curse of dimensionality refers to challenges that arise when working with high-dimensional data. In anomaly detection, as the number of dimensions increases, the density of the data decreases, making it difficult to identify patterns. Feature selection helps mitigate the curse of dimensionality by focusing on the most relevant dimensions and avoiding unnecessary computational costs associated with high-dimensional spaces.

5. Noise Reduction:

- Irrelevant or noisy features can introduce variability that hinders the ability of anomaly detection models to distinguish between normal and anomalous instances. Feature

selection aids in reducing noise by excluding irrelevant or redundant features, leading to more robust anomaly detection models.

6. Handling Collinearity:

- Collinearity, where features are highly correlated, can affect the stability and interpretability of anomaly detection models. Feature selection helps address collinearity by choosing a subset of features that capture essential information while minimizing redundancy.

7. Preventing Overfitting:

- In anomaly detection, models may be prone to overfitting, especially in high-dimensional spaces. Feature selection mitigates overfitting by focusing on the most informative features and reducing the risk of the model capturing noise in the data.

8. Efficient Resource Utilization:

- Selecting a subset of features not only improves computational efficiency but also optimizes the use of resources. Feature selection is especially important in real-time or resource-constrained applications, where the efficiency of the anomaly detection process is critical.

9. Addressing Data Imbalance:

- In many anomaly detection scenarios, normal instances significantly outnumber anomalous instances, leading to class imbalance. Feature selection can help address this imbalance by emphasizing features that better characterize both normal and anomalous behavior.

In summary, feature selection in anomaly detection is a strategic process that enhances the quality of models, improves interpretability, and addresses challenges associated with high-dimensional and noisy data. It contributes to the overall effectiveness and efficiency of anomaly detection systems in various domains.

Q2. What are some common evaluation metrics for anomaly detection algorithms and how are they
computed?
Ans:Evaluating the performance of anomaly detection algorithms is crucial to assess their effectiveness in identifying anomalies while minimizing false positives. Several metrics are

commonly used to evaluate the performance of anomaly detection algorithms. Here are some common evaluation metrics:

1. True Positive Rate (Sensitivity, Recall):

- Formula:
- True Positive Rate=True PositivesTrue Positives + False Negatives
- True Positive Rate=
    - True Positives + False Negatives
        - True Positives
            -
-
- Description: Measures the proportion of actual anomalies that are correctly identified by the model. A higher true positive rate indicates better sensitivity to anomalies.

2. False Positive Rate (Fallout):

- Formula:
- False Positive Rate=False PositivesFalse Positives + True Negatives
- False Positive Rate=
    - False Positives + True Negatives
        - False Positives
            -
-
- Description: Measures the proportion of normal instances that are incorrectly classified as anomalies. A lower false positive rate is desirable.

3. Precision (Positive Predictive Value):

- Formula:
- Precision=True PositivesTrue Positives + False Positives
- Precision=
    - True Positives + False Positives
        - True Positives
            -
-
- Description: Measures the accuracy of the model in correctly identifying anomalies among instances flagged as anomalies. A higher precision indicates fewer false positives.

4. F1 Score:

- Formula:
- $F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$
- $F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$
- Description: Combines precision and recall into a single metric, providing a balance between them. It is useful when there is an imbalance between the classes.

5. Area Under the Receiver Operating Characteristic (ROC) Curve (AUC-ROC):

- Description: The ROC curve is a graphical representation of the trade-off between true positive rate and false positive rate across different thresholds. AUC-ROC measures the area under the ROC curve, with higher values indicating better performance.

6. Area Under the Precision-Recall (PR) Curve (AUC-PR):

- Description: Similar to AUC-ROC, AUC-PR measures the area under the precision-recall curve. It is particularly useful when dealing with imbalanced datasets.

7. Matthews Correlation Coefficient (MCC):

- Formula:
- $MCC = \frac{True\ Positives \times True\ Negatives - False\ Positives \times False\ Negatives}{(True\ Positives + False\ Positives)(True\ Positives + False\ Negatives)(True\ Negatives + False\ Positives)(True\ Negatives + False\ Negatives)}$
- $MCC = \frac{True\ Positives \times True\ Negatives - False\ Positives \times False\ Negatives}{\sqrt{(True\ Positives + False\ Positives)(True\ Positives + False\ Negatives)(True\ Negatives + False\ Positives)(True\ Negatives + False\ Negatives)}}$
- Description: Takes into account true positives, true negatives, false positives, and false negatives, providing a balanced measure of classification performance.

8. Precision at a Given Recall Level:

- Description: Precision is calculated at a specific recall level, often chosen based on the application's requirements. It helps understand the trade-off between precision and recall.

9. Confusion Matrix:

- Description: A table that shows the counts of true positives, true negatives, false positives, and false negatives. It provides a comprehensive view of the model's performance.

10. Kappa Statistic:

- Formula:
- $\text{Kappa} = \frac{\text{Observed Accuracy} - \text{Expected Accuracy}}{1 - \text{Expected Accuracy}}$
- $\text{Kappa} = \frac{\text{Observed Accuracy} - \text{Expected Accuracy}}{1 - \text{Expected Accuracy}}$
- Description: Measures the agreement between the predicted and actual classifications, considering the possibility of random agreement.

11. Average Precision (AP):

- Description: Calculates the area under the precision-recall curve, providing a single-value summary of precision across different recall levels.

12. Recall at a Given False Positive Rate:

- Description: Measures recall at a specified false positive rate, giving insights into the model's performance at controlling false positives.

Important Considerations:

- The choice of evaluation metrics depends on the specific goals of the anomaly detection task and the characteristics of the dataset.
- A combination of metrics may provide a more comprehensive understanding of the model's performance.

- The interpretation of metrics should consider the context of the application and the consequences of false positives and false negatives.

When selecting an evaluation metric, it's essential to consider the specific goals and requirements of the anomaly detection task, such as the importance of correctly identifying anomalies, minimizing false positives, and achieving a balance between precision and recall.

Q3. What is DBSCAN and how does it work for clustering?
Ans:DBSCAN, which stands for Density-Based Spatial Clustering of Applications with Noise, is a popular clustering algorithm used in data mining and machine learning. It was introduced by Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu in 1996. DBSCAN is particularly effective in identifying clusters of arbitrary shapes and handling noise in the data.

Here's a brief overview of how DBSCAN works:

Density-Based Clustering:

DBSCAN defines clusters as dense regions in the data space separated by areas of lower point density. It doesn't assume a specific number of clusters and can discover clusters of various shapes and sizes.
Core Points, Border Points, and Noise:

- Core Points: A data point is considered a core point if it has at least a specified number of neighboring points (a predefined radius around it) within that radius.
- Border Points: A data point is considered a border point if it is within the radius of a core point but doesn't have enough neighbors to be a core point itself.
- Noise Points: Points that are neither core nor border points are considered noise points.

Steps of DBSCAN:

- Initialization: Choose an arbitrary data point that has not been visited.
- Density-Based Exploration:
    - If the chosen point is a core point, form a new cluster and include all reachable points (within the specified radius) in the cluster.
    - If the chosen point is a border point, assign it to the cluster of one of its core points.
- Continue the process until no more points can be added to the cluster.
- Move to an unvisited point and repeat the process.

Result:

- The algorithm continues until all points have been visited.
- The clusters formed are dense regions of points, and points that are not part of any cluster are considered noise.

Parameters:

- Epsilon (ε): The radius around each data point to define its neighborhood.
- MinPts: The minimum number of points required to form a dense region (core point).

Advantages:

- Can discover clusters of arbitrary shapes.
- Robust to noise and outliers.

Disadvantages:

- Sensitive to the choice of parameters (ε and MinPts).
- May have difficulty with clusters of varying densities.

In summary, DBSCAN is a density-based clustering algorithm that identifies clusters based on the density of data points. It's a useful algorithm for applications where clusters have varying shapes and densities, and it is capable of handling noisy data effectively.

Q4. How does the epsilon parameter affect the performance of DBSCAN in detecting anomalies?

Ans:The epsilon parameter (often denoted as ε) in DBSCAN controls the radius around each data point within which the algorithm looks for its neighbors. This parameter plays a crucial role in determining the performance of DBSCAN, particularly in the context of detecting anomalies or outliers. Here's how the epsilon parameter affects the performance of DBSCAN in detecting anomalies:

Impact on Cluster Size:

- Smaller ε values result in denser clusters because the algorithm requires a smaller neighborhood to consider a point as a core point.
- Larger ε values lead to larger neighborhoods, potentially merging multiple small clusters into a single larger cluster.

Sensitivity to Local Density:

- A smaller ε makes the algorithm more sensitive to local density variations, making it suitable for identifying smaller, more tightly packed clusters.
- Larger ε values result in a more global perspective, potentially overlooking smaller clusters and focusing on larger, more dispersed structures.

Influence on Noise Handling:

- Smaller ε values can lead to more points being treated as noise, as it becomes harder for points to satisfy the density criteria.
- Larger ε values may incorporate more points into clusters, potentially reducing the number of points labeled as noise.

Identification of Anomalies:

- Anomalies or outliers in the data are often characterized by lower local density compared to the surrounding points.
- Smaller ε values can be more effective in detecting anomalies, as the algorithm is sensitive to small, sparse regions in the data.

- However, too small ε values may result in overly fragmented clusters and may not capture anomalies that are part of larger, sparser structures.

Parameter Tuning Challenges:

- Choosing an appropriate ε value involves trade-offs. It requires understanding the density characteristics of the data and the desired granularity of clusters.
- The performance of DBSCAN in anomaly detection can be sensitive to the choice of ε, and tuning this parameter might be challenging, especially when the density of clusters varies.

Adaptability to Data Characteristics:

- The impact of ε on anomaly detection depends on the underlying characteristics of the data. A good choice for ε depends on the scale of the data and the expected size and density of clusters.

In summary, the epsilon parameter in DBSCAN significantly influences the algorithm's ability to detect anomalies by controlling the size and density of clusters. The choice of an appropriate ε value requires a careful understanding of the data's characteristics and the desired behavior of the algorithm in terms of capturing both small, dense clusters and larger, sparser structures.

Q5. What are the differences between the core, border, and noise points in DBSCAN, and how do they relate
to anomaly detection?

Ans:In DBSCAN (Density-Based Spatial Clustering of Applications with Noise), points in the dataset are classified into three categories: core points, border points, and noise points. These classifications are based on the local density of points within a specified radius (ε) around each data point. Understanding these classifications is essential for comprehending how DBSCAN identifies clusters and handles anomalies.

Core Points:

- A data point is considered a core point if it has at least "MinPts" (a user-defined parameter) number of neighboring points within the distance of ε.
- Core points are at the heart of a cluster and are used as starting points for growing clusters.
- In the context of anomaly detection, core points are typically part of the normal, dense regions of the data. They represent the core of a cluster, and anomalies are often found in regions with fewer core points.

Border Points:

- A data point is a border point if it is within the ε radius of a core point but does not have enough neighboring points to be a core point itself (i.e., fewer than MinPts neighbors).
- Border points are part of a cluster but are on the periphery. They connect clusters and help expand them.

- In anomaly detection, border points are less likely to be anomalies as they are part of clusters. However, anomalies may still be present among border points if they are at the edge of a cluster and deviate significantly from the cluster's characteristics.

Noise Points:

- A data point is considered a noise point (or an outlier) if it is neither a core point nor a border point. It does not have enough neighbors within ε to be part of a cluster.
- Noise points are often isolated points or small groups of points that do not fit well into any cluster.
- In the context of anomaly detection, noise points are potential anomalies. They represent regions of lower density or isolated points that do not conform to the patterns found in the rest of the data.

Relation to Anomaly Detection:

- Anomalies among Noise Points: Noise points, by definition, are not part of any cluster and are treated as potential anomalies. Outlying points or regions with lower density are often flagged as anomalies.
- Anomalies in Border Regions: Anomalies may also be found among border points, especially if these points are at the edges of clusters and exhibit characteristics significantly different from the majority of the cluster.
- Normal Regions Defined by Core Points: Core points define the dense, normal regions of the data. Anomalies are typically detected in regions with fewer core points or in regions where the density drops significantly.

In summary, core points represent the dense core of clusters, border points are on the periphery, and noise points are potential anomalies. Anomalies are often associated with regions of lower density, and DBSCAN's ability to classify points into these categories makes it useful for anomaly detection in spatial data.

Q6. How does DBSCAN detect anomalies and what are the key parameters involved in the process?
Ans:DBSCAN (Density-Based Spatial Clustering of Applications with Noise) can be used for anomaly detection by identifying points that do not belong to any dense cluster, designating them as noise points or outliers. The key parameters involved in the anomaly detection process with DBSCAN are:

Epsilon (ε):

- Epsilon is the radius around each data point within which the algorithm looks for its neighbors.

- It influences the size of the neighborhood used to define density around a point.
- A smaller ε makes the algorithm more sensitive to local density variations, potentially leading to more fragmented clusters and a higher likelihood of noise points being identified as anomalies.
- A larger ε, on the other hand, may result in merging clusters and overlooking smaller, sparser structures, potentially missing anomalies.

MinPts (Minimum Points):

- MinPts is the minimum number of neighboring points within the ε radius required for a data point to be considered a core point.
- Core points are central to the formation of clusters. If a point has fewer than MinPts neighbors, it may be classified as a noise point or part of a smaller cluster.
- MinPts influences the sensitivity to the local density required for a point to be considered a core point. Higher MinPts values result in a stricter density criterion.

Clusters and Noise Points:

- DBSCAN classifies points into three categories: core points, border points, and noise points.
- Core points form the dense cores of clusters, while border points are on the periphery. Noise points, which are neither core nor border points, are potential anomalies.
- Anomalies are often located in regions with lower point density, and noise points represent these regions.

Reachability and Connectivity:

- The concept of reachability is crucial in determining which points are reachable from a core point.
- A point is reachable if there is a path of core points from the starting core point to that point.
- Anomalies are often isolated points or points with fewer reachable neighbors.

Cluster Expansion:

- DBSCAN starts with an arbitrary, unvisited data point and expands the cluster by including all reachable points within the ε radius.
- The process continues until no more points can be added to the cluster, and unvisited points are explored sequentially.
- Points that cannot be reached by any cluster formation process are labeled as noise.

In summary, DBSCAN detects anomalies by identifying points that do not belong to any dense cluster. The choice of parameters, particularly ε and MinPts, influences the algorithm's sensitivity to local density and, consequently, its ability to detect anomalies. A careful selection of these parameters is essential for effective anomaly detection using DBSCAN.

Q7. What is the make_circles package in scikit-learn used for?
Ans:In scikit-learn, the `make_circles` function is part of the `datasets` module and is used to generate synthetic datasets consisting of circles. This function is particularly useful for creating datasets that exhibit circular or annular structures, which can be valuable for testing and visualizing clustering algorithms and classification models.

Here's a brief overview of the `make_circles` function:

python

Copy code

```
from                    import

                              100        0.05         0.5                42
```

Parameters:

- `n_samples`: The total number of points in the dataset.
- `noise`: Standard deviation of Gaussian noise added to the data.
- `factor`: Scale factor between the inner and outer circle. A factor of 0 creates concentric circles, and a factor closer to 1 creates more separated circles.
- `random_state`: Seed for reproducibility.

The resulting dataset consists of two features (X) and binary labels (y) indicating the circular structure to which each point belongs. It's often used in machine learning for tasks like clustering and classification to evaluate algorithms' performance on non-linear and more complex structures.

For example, when testing clustering algorithms, the `make_circles` dataset is suitable for algorithms that can handle circular or annular structures, such as DBSCAN, which is effective in clustering non-linear shapes. Similarly, when testing classification algorithms, this dataset can be used to evaluate how well a model can distinguish between the different circular regions.

In summary, `make_circles` is a utility function in scikit-learn that facilitates the creation of synthetic datasets with circular or annular structures, making it a useful tool for testing and validating machine learning algorithms on non-linear data.

Q8. What are local outliers and global outliers, and how do they differ from each other?
Ans:Local outliers and global outliers refer to different types of anomalies or outlying points in a dataset, and they are characterized by the extent to which their abnormality is assessed within the context of their local or global neighborhood. The differences between local and global outliers are as follows:

Local Outliers:

- Definition: Local outliers, also known as local anomalies or inliers, are data points that deviate significantly from their local neighborhood but might not stand out when considering the entire dataset.
- Detection Focus: The emphasis is on the local context of each data point, examining its surroundings or neighborhood to identify anomalies.
- Methodology: Techniques such as local density-based methods (e.g., Local Outlier Factor - LOF) or distance-based methods (e.g., k-Nearest Neighbors) are commonly used to identify local outliers.
- Example: In a clustering scenario, a local outlier might be a data point surrounded by other points from a different cluster.

Global Outliers:

- Definition: Global outliers, also referred to as global anomalies or outliers, are data points that deviate significantly when considering the entire dataset, irrespective of their local neighborhood.
- Detection Focus: The analysis extends to the overall characteristics of the entire dataset, identifying points that are exceptional when compared to the entire distribution.
- Methodology: Statistical methods, such as z-scores or interquartile range (IQR), or model-based approaches are often used to detect global outliers.
- Example: In a dataset representing exam scores, a global outlier might be a student with a score much higher or lower than the majority of students, regardless of the local distribution within specific classes.

Differences:

- Context: Local outliers are defined in the context of the local neighborhood of each data point, while global outliers are determined by considering the entire dataset.
- Sensitivity: Local outliers are often more sensitive to variations within smaller regions of the data, whereas global outliers are concerned with overall data distribution and extreme values in the entire dataset.

- Application: Local outliers are useful when anomalies are expected to have a more localized impact or when the dataset has diverse local structures. Global outliers are suitable when the focus is on identifying anomalies that significantly deviate from the overall pattern of the data.

In practice, the choice between detecting local or global outliers depends on the specific characteristics of the dataset and the goals of the analysis. Different outlier detection techniques are employed based on whether the emphasis is on local or global characteristics.

Q9. How can local outliers be detected using the Local Outlier Factor (LOF) algorithm?
Ans:The Local Outlier Factor (LOF) algorithm is a popular method for detecting local outliers in a dataset. It measures the local density deviation of a data point with respect to its neighbors, making it effective in identifying anomalies within specific regions. Here's a general overview of how the LOF algorithm works for detecting local outliers:

Local Density Estimation:
- For each data point, the algorithm estimates its local density by comparing the density of the point with the density of its neighbors. The density is computed based on the distance to the k-nearest neighbors of the point.

Reachability Distance:
- The reachability distance of a point is the distance to its k-nearest neighbor with the highest local density.
- It reflects how far a point is from its neighbors in terms of local density.

Local Reachability Density:
- Local Reachability Density (LRD) is the inverse of the average reachability distance of a point's neighbors.
- It quantifies the local density of a point relative to its neighbors.

LOF Calculation:
- The Local Outlier Factor for each point is computed as the ratio of its local reachability density to the average local reachability density of its neighbors.
- A higher LOF indicates that the point has a lower density than its neighbors, suggesting that it might be a local outlier.

Outlier Identification:
- Points with a significantly higher LOF than the average LOF in the dataset are considered local outliers.
- A higher LOF indicates that the point is less dense compared to its neighbors, making it potentially anomalous in its local context.

Here's an example of using LOF in Python with scikit-learn:

python

Copy code

```
from                    import
import       as


              1  2    1.5  1.8    5  8    8  8    1  0.6    9  11


                                        2


print "LOF Scores:"
```

In this example, points with negative LOF scores are considered outliers. The choice of parameters, such as the number of neighbors (`n_neighbors`), can affect the performance of LOF and should be tuned based on the characteristics of the dataset.

Q10. How can global outliers be detected using the Isolation Forest algorithm?

Ans:The Isolation Forest algorithm is an anomaly detection algorithm designed to identify global outliers in a dataset. It is based on the principle that anomalies are often easier to isolate than normal instances. Here's a general overview of how the Isolation Forest algorithm works for detecting global outliers:

Randomized Partitioning:

- The algorithm builds an ensemble of isolation trees, which are binary trees.
- Each tree is constructed by recursively partitioning the data into subsets using randomly chosen features and split points.

Outlier Isolation:

- During the construction of the trees, anomalies (outliers) are expected to be isolated more quickly than normal instances.

- Anomalies are likely to have shorter paths in the tree as they are isolated in smaller partitions.

Path Length Calculation:

- For each data point, the algorithm calculates the average path length in the ensemble of trees to isolate that point.
- Shorter average path lengths indicate that the point is easier to isolate and is potentially an outlier.

Anomaly Score:

- The average path length for each point is used as an anomaly score. Points with shorter average path lengths are assigned higher anomaly scores.
- Higher anomaly scores indicate that a data point is likely to be a global outlier.

Thresholding:

- A threshold is applied to the anomaly scores to classify points as outliers or inliers.
- Points with anomaly scores exceeding the threshold are considered global outliers.

Here's an example of using Isolation Forest in Python with scikit-learn:

python

Copy code

```
from                  import

import       as


    1  2    1.5  1.8    5  8    8  8    1  0.6    9  11
```

```
                                                    0.2
```

```
print "Outlier Predictions:"
```

In this example, points with a prediction of -1 are considered outliers. The `contamination` parameter is set to the expected proportion of outliers in the dataset and can be adjusted based on domain knowledge or tuning.

Isolation Forest is efficient and effective for detecting global outliers, and it can handle high-dimensional datasets. It's particularly useful when outliers are expected to be less frequent than normal instances.

Q11. What are some real-world applications where local outlier detection is more appropriate than global

outlier detection, and vice versa?

Ans:The choice between local and global outlier detection depends on the characteristics of the dataset and the specific requirements of the application. Here are some real-world applications where each approach may be more appropriate:

Local Outlier Detection:

Network Security:

- Local outlier detection is valuable for identifying unusual patterns or activities in a local context within a network.
- Anomalies such as unusual traffic behavior or communication patterns might be detected by examining local neighborhoods in network data.

Manufacturing Quality Control:

- In manufacturing processes, local outlier detection can be used to identify anomalies in specific regions of a production line.
- Detecting local defects or irregularities in a localized context, such as a specific batch or section of a product, is crucial for maintaining overall product quality.

Health Monitoring:

- Local outlier detection can be applied in health monitoring to identify abnormal patterns in physiological data within localized time intervals.
- It is useful for detecting anomalies that occur in specific time windows, such as irregular heartbeats or spikes in sensor readings during a specific activity.

Fraud Detection in Finance:

- Local outlier detection is suitable for identifying unusual patterns or transactions within localized segments of financial data.
- It can help detect fraudulent activities that might be anomalous within a specific subset of transactions.

Global Outlier Detection:

Financial Fraud Across Multiple Accounts:

- Global outlier detection is appropriate when anomalies need to be identified across multiple accounts or transactions.
- Detecting patterns that are unusual when considering the entire dataset helps identify fraudulent activities that span multiple entities.

Ecological Monitoring:

- In environmental studies, global outlier detection can be used to identify anomalies in large-scale ecological data.
- Detecting anomalies that affect entire ecosystems or regions, rather than localized areas, is crucial for understanding and addressing environmental issues.

Supply Chain Management:

- Global outlier detection can be applied to monitor and identify anomalies across the entire supply chain.
- It helps in detecting abnormalities in the flow of goods or information that may affect the overall efficiency and reliability of the supply chain.

Credit Scoring:

- When assessing creditworthiness, global outlier detection can be used to identify individuals or entities with financial behaviors that deviate significantly from the overall population.
- It helps in identifying high-risk individuals who may have patterns of financial behavior that are globally unusual.

In summary, the choice between local and global outlier detection depends on the specific context of the application. Local outlier detection is suitable when anomalies are expected to be confined to specific local contexts, while global outlier detection is more appropriate when

anomalies may affect the overall system or dataset. Often, a combination of both approaches may be employed to provide a comprehensive view of anomalous patterns in a dataset.