Isolation Forest: Anomaly Detection

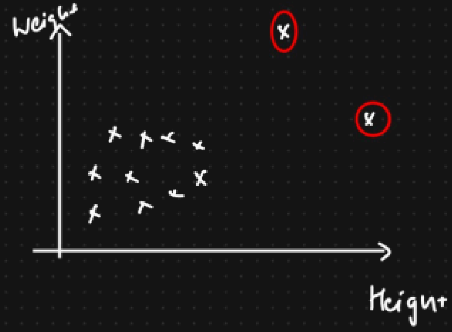
1. **Core Idea:**
    - Isolation Forest operates on the principle that anomalies (outliers) are "few and different."
    - Because they are rare and have distinct feature values, they should be easier to *isolate* from the rest of the data points compared to normal points.
2. **How it Works:**
    - **Ensemble Method:** It builds an ensemble (a "forest") of multiple Isolation Trees (iTrees).
    - **Building an iTree:**
        - A random subsample of the data is selected.
        - The algorithm recursively partitions (splits) this subsample:
            - It randomly selects a feature.
            - It randomly selects a split value between the minimum and maximum values of that feature in the current data subset.
            - Data points are divided based on whether their value for the selected feature is above or below the split value.
        - This continues until each data point is isolated in its own leaf node or a predefined maximum tree depth is reached.
        - Above is the decision tree.
    - **Anomaly Identification:**
        - Anomalous points, being different, tend to require fewer random splits to be isolated. They will generally be closer to the root of the iTrees (i.e., have a shorter *path length*).
        - Normal points are more numerous and similar, requiring more splits to isolate, resulting in longer path lengths.
    - **Anomaly Score:**
        - The algorithm calculates the average path length for each data point across all the iTrees in the forest.
        - This average path length is used to compute an anomaly score, typically normalized to be between 0 and 1.
        - **Score Interpretation:**
            - Scores close to 1 indicate a high likelihood of being an anomaly (short average path length).
            - Scores significantly less than 0.5 suggest a normal data point (long average path length).
            - Scores around 0.5 indicate ambiguity.
3. **Key Parameters (Common in libraries like scikit-learn):**
    - `n_estimators`: The number of iTrees to build in the forest. More trees generally lead to more stable results. (Default often 100).
    - `max_samples`: The number (or fraction) of samples used to build each individual iTree. Controls the degree of subsampling. (Default often 'auto' or 256).

- `contamination`: An estimate of the proportion of outliers expected in the dataset (e.g., 0.01 for 1%). This helps set the threshold for classifying points as anomalies vs. normal points when using the `predict` method. (Range (0, 0.5]).
- `max_features`: The number (or fraction) of features to consider when making a random split.

4. **Advantages:**
    - **Efficiency:** Computationally efficient with low memory requirements and often linear time complexity, making it suitable for large datasets.
    - **Scalability:** Works well with high-dimensional data where distance-based methods often struggle.
    - **No Distance Metric:** Doesn't rely on distance calculations, avoiding issues related to the "curse of dimensionality."
    - **No Distribution Assumptions:** Doesn't require data to fit a specific statistical distribution.
    - **Handles Swamping/Masking:** Subsampling helps reduce the impact of masking (dense anomaly clusters) and swamping (normal points close to anomalies).
5. **Disadvantages:**
    - **Parameter Sensitivity:** Performance can be sensitive to parameter choices, especially `contamination`.
    - **Axis-Parallel Splits:** May not be optimal for datasets where anomaly separation requires diagonal boundaries.
    - **Interpretability:** Can be less interpretable than simpler methods ("black box" nature), although techniques exist to help explain predictions (e.g., SHAP).
    - **Local Anomalies:** May sometimes be less effective at detecting anomalies that are only outliers within a specific local region.
    - **Clustered Anomalies:** Performance might degrade if anomalies form dense clusters themselves.
6. **Common Applications:**
    - Fraud detection (financial transactions, insurance claims)
    - Network intrusion detection
    - Identifying faulty sensor data or equipment failure (predictive maintenance)
    - Detecting outliers in healthcare data
    - Data preprocessing/cleaning

Anomaly Detection [To detect Outliers]

↳ Play a important Role



$\boxed{\text{IPL}}$

1	15
2	10
3	12
$\boxed{4}$	$\boxed{100}$

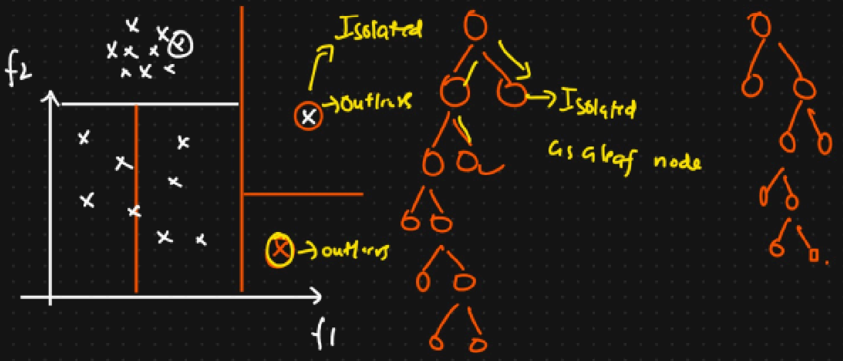
> 36

Many Trees

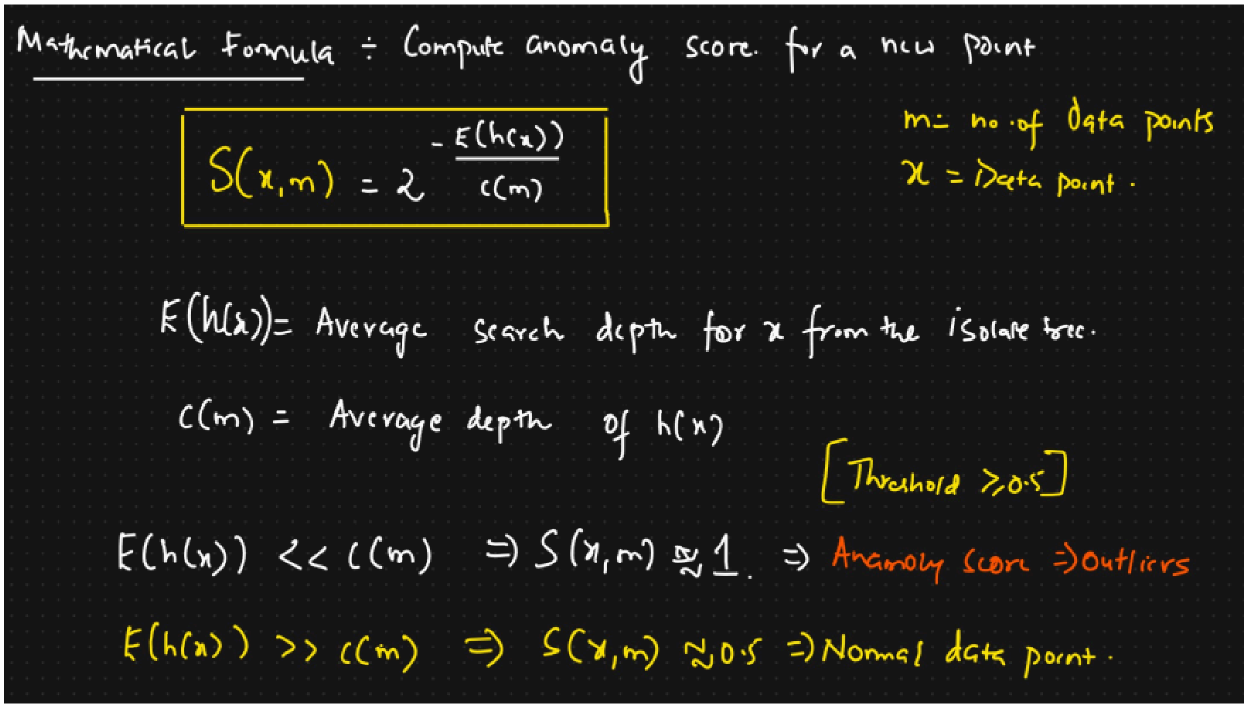
① Isolation Forest [Decision Trees].

Isolated Trees

f1	f2	f3	f4
–	–	–	–
–	–	–	–
–	–	–	–
–	–	–	–



Anomaly Score



Mathematical Formula ÷ Compute anomaly score for a new point

$$\boxed{S(x,m) = 2^{-\frac{E(h(x))}{c(m)}}}$$

m = no. of Data points

x = Data point.

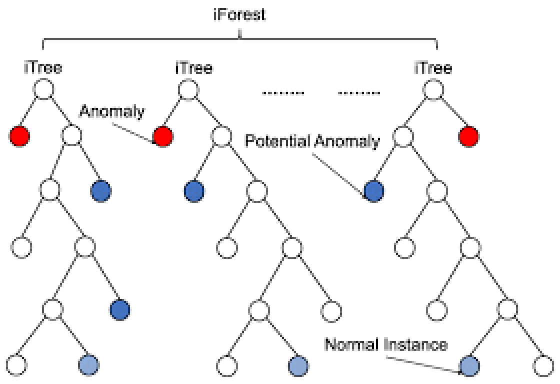
$E(h(x))$ = Average search depth for x from the isolate tree.

$c(m)$ = Average depth of $h(x)$

[Threshold ≥ 0.5]

$E(h(x)) << c(m) \Rightarrow S(x,m) \approx 1 \Rightarrow$ Anamoly score $\Rightarrow$ outliers

$E(h(x)) >> c(m) \Rightarrow S(x,m) \approx 0.5 \Rightarrow$ Normal data point.



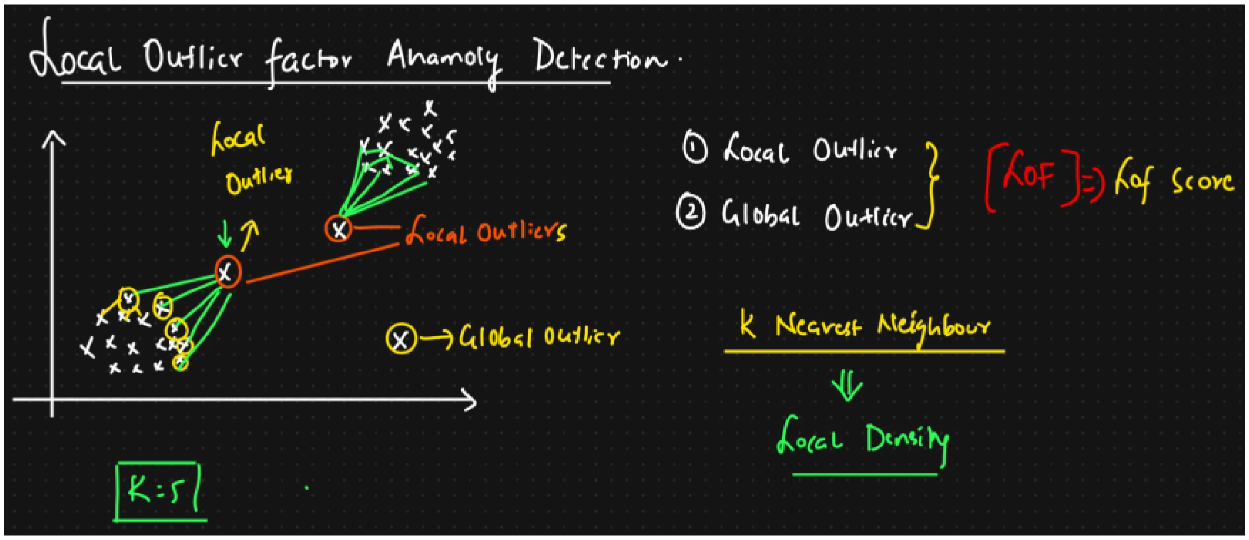
Local Outlier Factor (LOF): Anomaly Detection

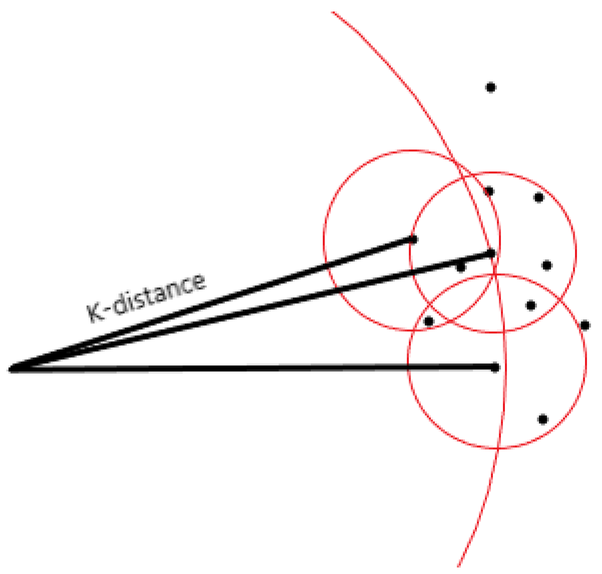
1. **Core Idea:**
   - LOF is an unsupervised, density-based algorithm that identifies outliers by comparing the *local density* of a data point to the local densities of its neighbors.
   - The central concept is that an outlier will have a substantially lower density than its neighbors, whereas an inlier will have a density similar to its neighbors.
   - Its strength lies in detecting outliers in datasets with varying densities, where a point might be an outlier relative to its local neighborhood but not necessarily in a global context.
2. **Key Concepts & How it Works:**
   - **k-Nearest Neighbors (k-NN):** For each point `p`, the algorithm first finds its `k` nearest neighbors. The parameter `k` (often called `n_neighbors`) is crucial.
   - **k-distance(p):** The distance between point `p` and its k-th nearest neighbor.

    - **Reachability Distance (reach-dist_k(p, o)):** This is defined as `max(k-distance(o), actual_distance(p, o))`. It's the true distance from `p` to a neighbor `o`, but it's never smaller than the k-distance of neighbor `o`. This helps smooth density estimates within clusters.
    - **Local Reachability Density (lrd_k(p)):** This measures the local density around point `p`. It's calculated as the inverse of the *average* reachability distance from point `p` to all of its k-neighbors. A higher `lrd` means the point is in a denser region.
    - **Local Outlier Factor (LOF_k(p)):** This is the final anomaly score for point `p`. It's calculated as the ratio of the *average* `lrd` of `p`'s k-neighbors to the `lrd` of `p` itself. `LOF_k(p) = average(lrd of neighbors) / lrd(p)`
3. **Interpreting LOF Scores:**
    - **`LOF ≈ 1`:** The point `p` has a density similar to its neighbors (likely an inlier).
    - **`LOF > 1`:** The point `p` is in a sparser region (lower density) than its neighbors (likely an outlier). The larger the LOF value, the more anomalous the point is considered.
    - **`LOF < 1`:** The point `p` is in a denser region than its neighbors (can happen for points inside a dense cluster).
4. **Key Parameters:**
    - `n_neighbors` (`k`): The number of neighbors to use for local density estimation. This is the most critical parameter. Choosing it too small makes the algorithm sensitive to noise; choosing it too large can blur the locality. A common starting point is 20, but tuning is often needed.
    - `metric`: The distance metric used to measure distances between points (e.g., 'euclidean', 'manhattan', 'minkowski').
    - `contamination` (used in libraries like scikit-learn): An estimate of the expected proportion of outliers in the dataset. This helps set a threshold on the LOF scores to classify points as outliers (-1) or inliers (1).
    - `algorithm`: The algorithm used to compute nearest neighbors ('auto', 'ball_tree', 'kd_tree', 'brute').
    - `novelty`: A parameter (e.g., in scikit-learn) to switch between outlier detection (finding anomalies in the training data) and novelty detection (using the trained model to find anomalies in *new*, unseen data).
5. **Advantages:**
    - **Effective in Varying Densities:** Excels where global methods might fail because it considers local context.
    - **Unsupervised:** No need for labeled data.
    - **Provides Scoring:** Gives a score indicating the *degree* of outlierness, not just a binary label.
    - **Well-Established:** A widely studied and often effective density-based approach.
6. **Disadvantages:**

- **Computational Complexity:** Calculating distances and neighbors for all points can be computationally intensive (potentially O(n²) without optimizations, or O(n log n) with spatial indexing), making it slower for very large datasets.
- **Parameter Sensitivity:** Performance is highly dependent on the choice of `k` (`n_neighbors`).
- **Curse of Dimensionality:** Like other distance-based methods, its effectiveness can decrease in high-dimensional spaces as distances become less meaningful.
- **Score Interpretation:** While >1 indicates an outlier, the magnitude can vary between datasets and parameter settings, making thresholding difficult without the `contamination` parameter or domain expertise.

7. **Common Applications:**
    - Intrusion detection in networks.
    - Fraud detection.
    - Identifying anomalies in geographic data or video streams.
    - Data cleaning and preprocessing.




K-distance