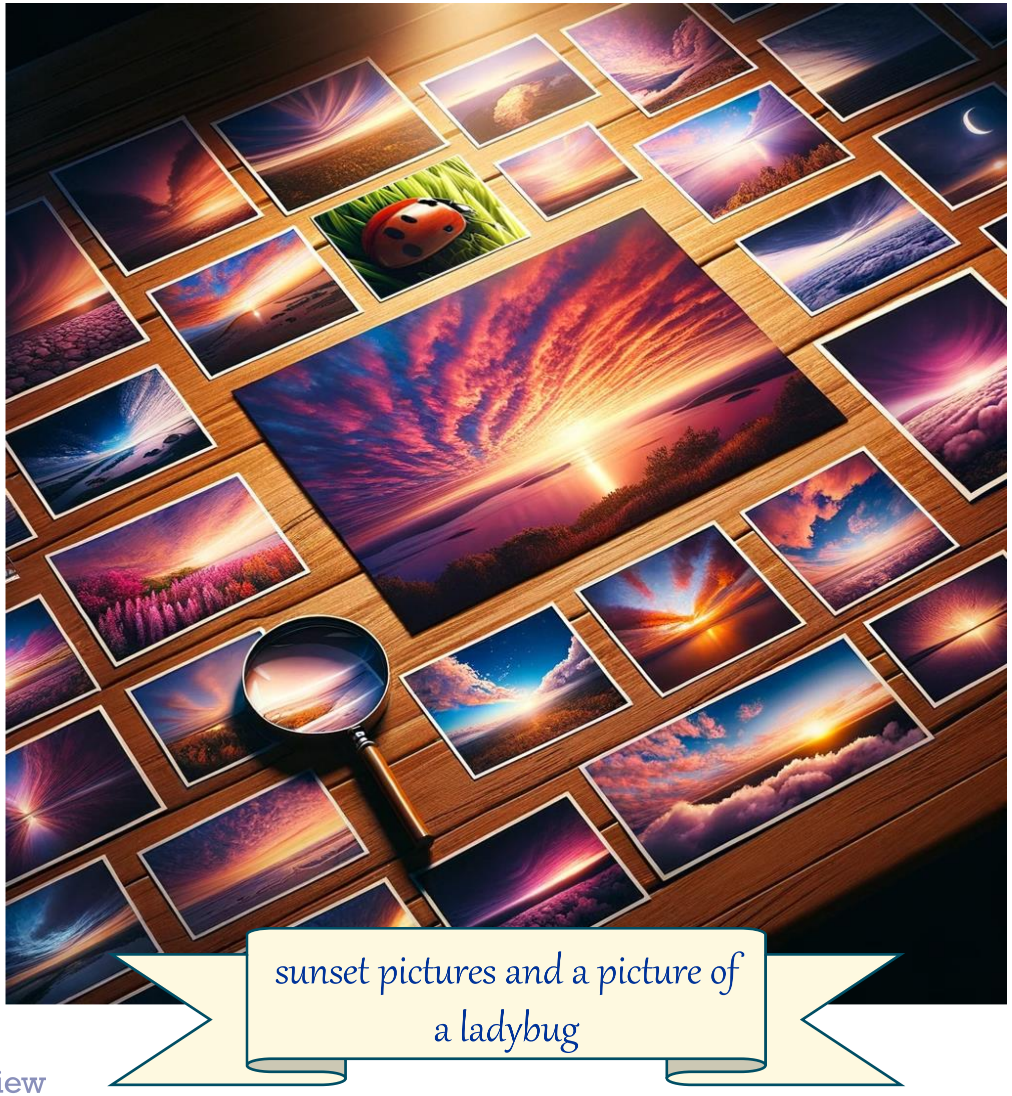
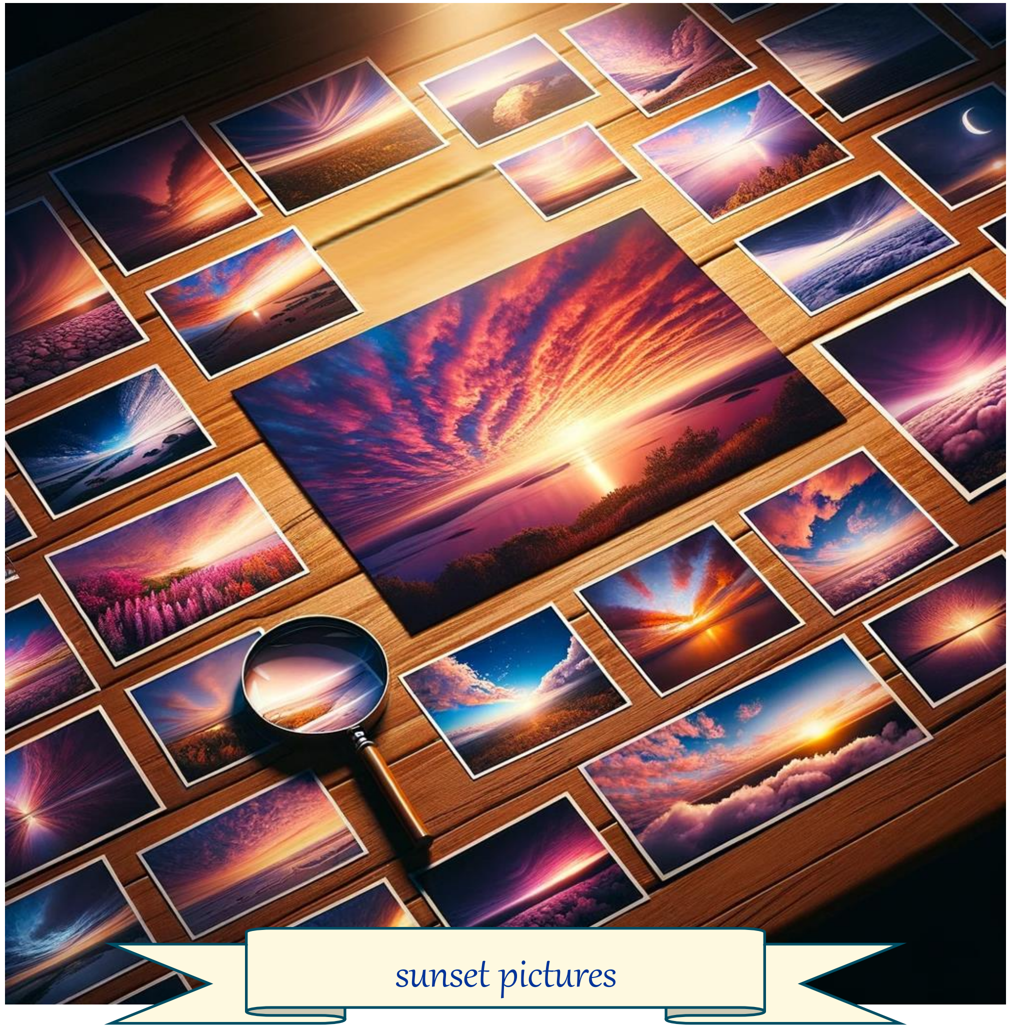
Information-Theoretic Models



Information-Theoretic Models

- Idea: Outliers increase the minimum code length required to describe a dataset
- Construct a code book to represent the data in
- Outliers are defined as points whose removal results in the largest decrease in description length



Information-Theoretic Models (cont.)

Example:

1. AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB =: normal data
2. AB AB AC AB AB AB AB AB AB AB AB AB AB AB AB AB AB =: data with outlier

Description:

1. „17 × AB“
2. „2 × AB, AC, 14 × AB“

Information-Theoretic Models (cont.)

- Determination of the minimum-length coding is computationally intractable
- Variety of heuristic models can be used
- Often same methods as other techniques (probabilistic models, frequent pattern mining, histograms, PCA) to create coding representation

- Indirect approach to model outlierness can blunt the scores (aggregate error)

⇒ **information-theoretic models often do not outperform conventional counterparts**

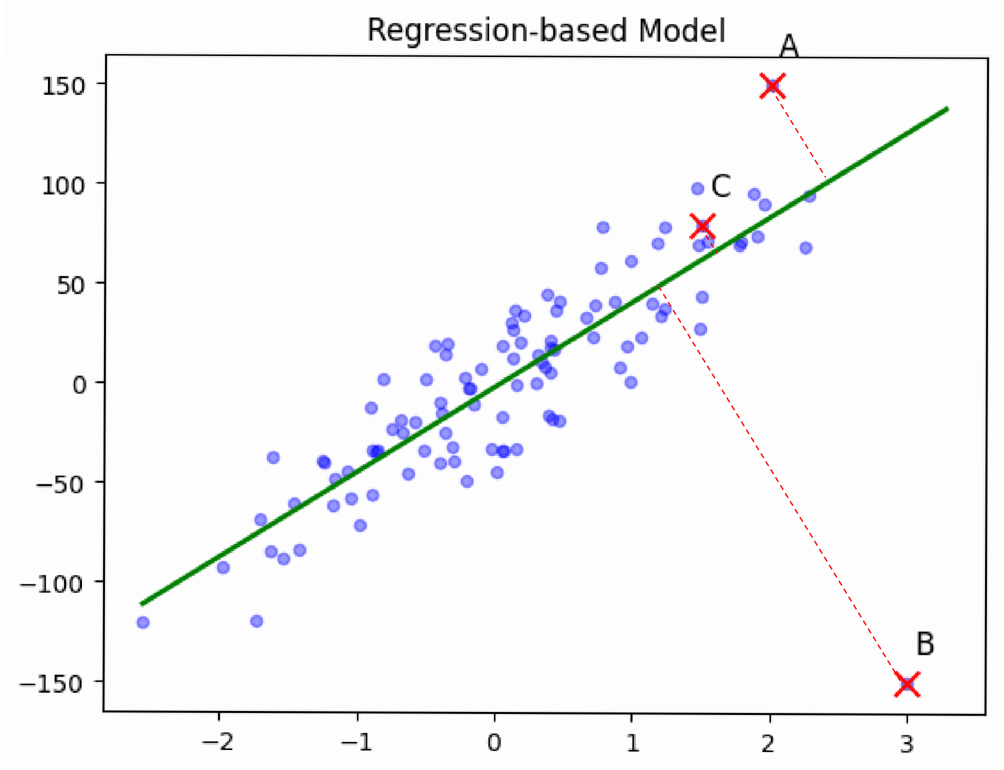
- Best use-case: Where quantifying coding cost is more convenient than measuring derivations

Linear Models

- Data modeled along lower-dimensional subspaces (1-d line in 2-d space)
- Optimal line determined by regression analysis (least-squares fit)
- Distances from data point to line (hyperplane) quantifies outlierness
- Extreme-value analysis can be applied on those scores

Linear model of the points $\{(x_i, y_i), i \in \{1, \dots, N\}\}$ with two coefficients a, b and residuals ε quantifying the modelling error

$$y_i = a \cdot x_i + b + \varepsilon_i \quad \forall i \in \{1, \dots, N\}$$



Linear Models (cont.)

- Parameters a and b are learned from data minimizing $\sum_i \varepsilon_i^2$
- Convex problem that can be solved in closed form
- Squared residuals correspond to outlier score
- More general: residual distance to a lower-dimensional representation as outlier score
- Extreme-value analysis can be applied on those residuals

Linear model of the points $\{(x_i, y_i), i \in \{1, \dots, N\}\}$ with two coefficients a, b and residuals ε quantifying the modelling error

$$y_i = a \cdot x_i + b + \varepsilon_i \;\; \forall i \in \{1, \dots, N\}$$

Linear Models (cont.)

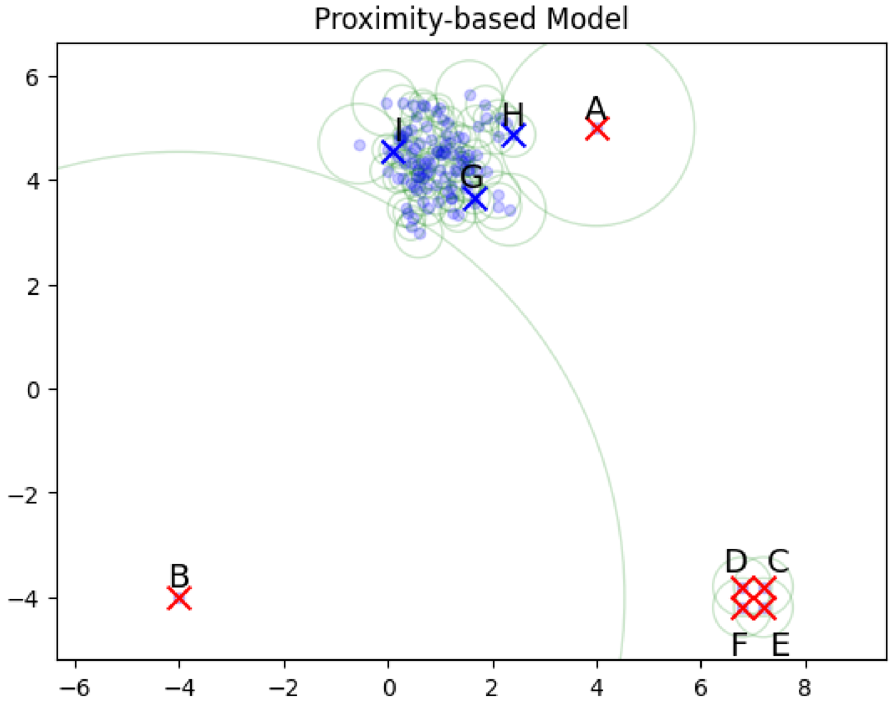
Related approaches:

- PCA: Principal Component Analysis
- OC-SVM: One-Class Support Vector Machine
- SVDD: Support Vector Data Description

Proximity-based Models

- Idea: outlier as isolated points
- Proximity based on similarity or distance functions
- Types of proximity-based models
  - Clustering methods (segment data points)
  - Density-based methods (segment data space)
  - Nearest-neighbor methods (distance to k-th nearest neighbor)
- Outliers as points located far away from dense regions
- Examples:
  - k-nearest neighbor distance, distance to closest cluster centroid, or local density value is the outlier score

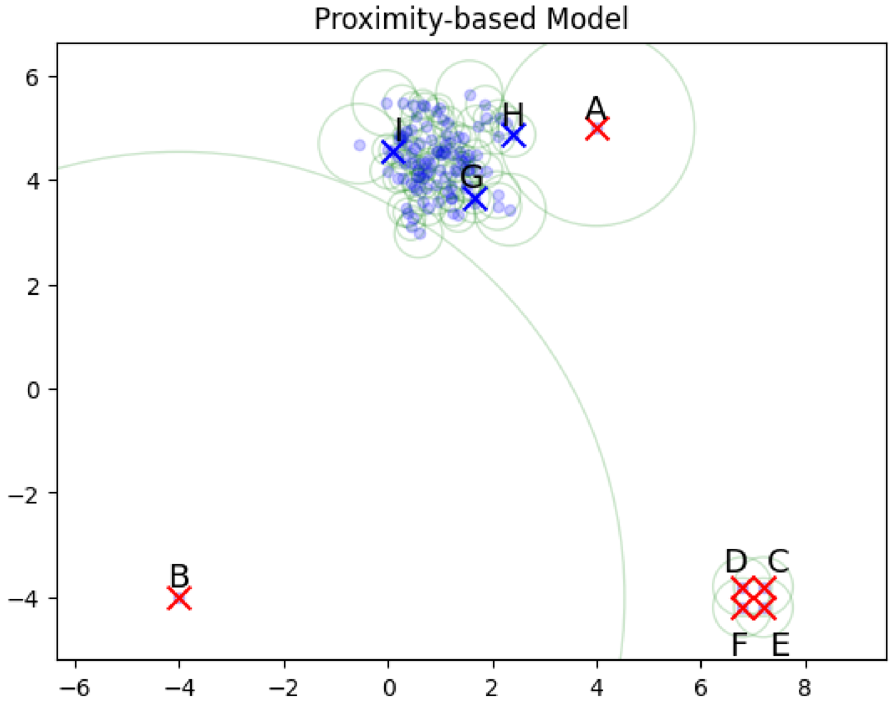
```
Distance to 3rd nearest neighbor for point A: 1.88
Distance to 3rd nearest neighbor for point B: 8.54
Distance to 3rd nearest neighbor for point C: 0.57
Distance to 3rd nearest neighbor for point D: 0.57
Distance to 3rd nearest neighbor for point E: 0.57
Distance to 3rd nearest neighbor for point F: 0.57
Distance to 3rd nearest neighbor for point G: 0.30
Distance to 3rd nearest neighbor for point H: 0.42
Distance to 3rd nearest neighbor for point I: 0.25
```



Proximity-based Models (cont.)

- Clustering: Partitioning and outlier scores vary with clustering algorithm
  ⇒ Cluster multiple times and average

```
Distance to 3rd nearest neighbor for point A: 1.88
Distance to 3rd nearest neighbor for point B: 8.54
Distance to 3rd nearest neighbor for point C: 0.57
Distance to 3rd nearest neighbor for point D: 0.57
Distance to 3rd nearest neighbor for point E: 0.57
Distance to 3rd nearest neighbor for point F: 0.57
Distance to 3rd nearest neighbor for point G: 0.30
Distance to 3rd nearest neighbor for point H: 0.42
Distance to 3rd nearest neighbor for point I: 0.25
```



Disadvantages:

- Small groups of outliers hard to find (which is quite common)
- k-NN is computationally expensive

Proximity-based Models (cont.)

Related approaches:

- DB(ε, π)-Outlier: Density-Based Outlier Detection
- k-NN Outlier: k-Nearest Neighbors Outlier Detection
- LOF: Local Outlier Factor
- LOCI: Local Correlation Integral
- Cluster-based Outlier Factor

Outlier Ensembles

- Meta algorithms that combine the outputs of multiple algorithms
- Examples from classification: boosting, stacking, bagging, subspace sampling

- **Sequential ensembles** (e.g. boosting)
  - Algorithms are applied sequentially such that the succeeding algorithms are influenced by previous algorithms
  - Result can be weighted combination of algorithms or output of the final algorithm etc.

Outlier Ensembles (cont.)

```
Algorithm SequentialEnsemble(Data Set: D,
        Base Algorithms: A1 . . . Ar)
begin
    j = 1;
    repeat
      Pick an algorithm Aj based on results
        from past executions;
      Create a new data set fj (D) from D
        based on results from past executions;
      Apply Aj to fj(D);
      j = j + 1;
    until(termination);
    report outliers based on combinations of
      results from previous executions;
end
```

```python
def SequentialEnsemble(D, A):
    j = 0
    past_results = []
    while not termination_condition(j, D):
        Aj = pick_algorithm(j, past_results)
        fj_D = create_new_dataset(D, past_results)

        results = apply_algorithm(Aj, fj_D)
        past_results.append(results)

        j += 1

    outliers = report_outliers(past_results)
    return outliers
```

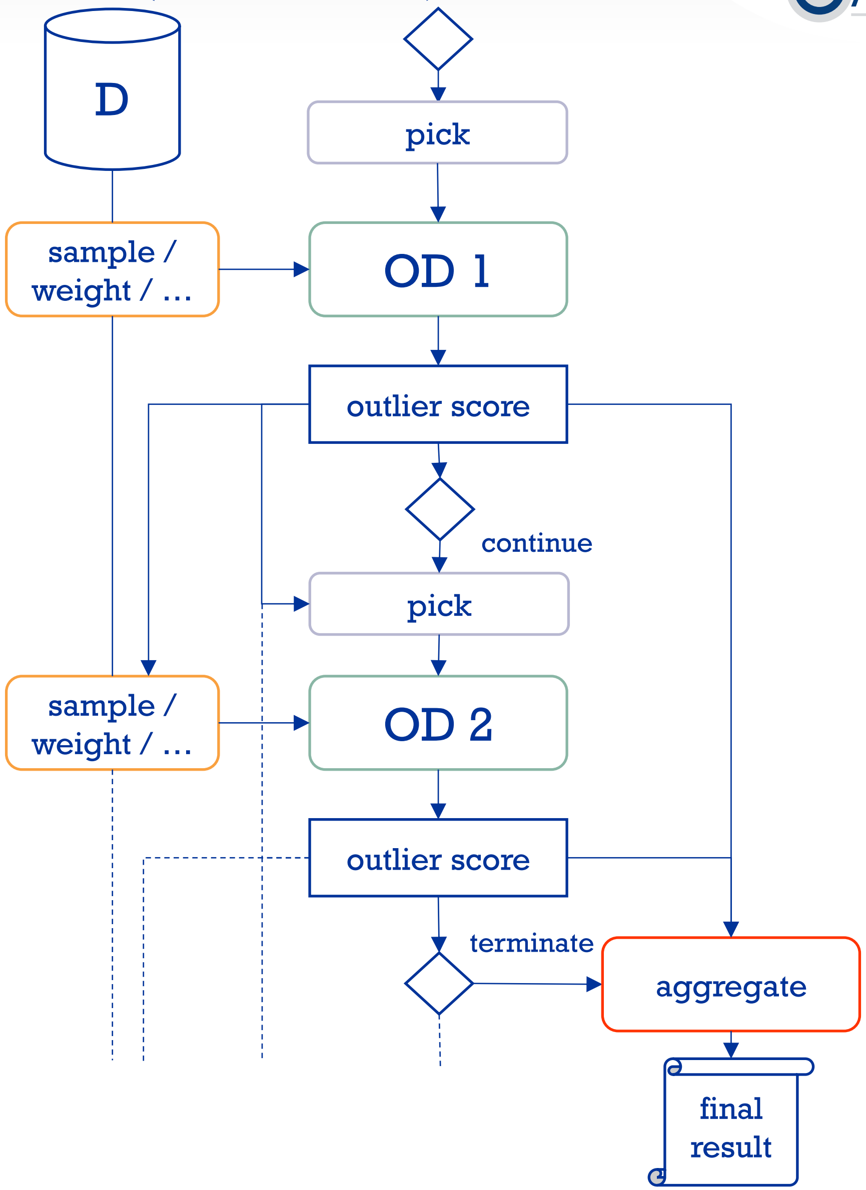
Outlier Ensembles (cont.)

```python
def SequentialEnsemble(D, A):
    j = 0
    past_results = []
    while not termination_condition(j, D):
        Aj = pick_algorithm(j, past_results)
        fj_D = create_new_dataset(D, past_results)

        results = apply_algorithm(Aj, fj_D)
        past_results.append(results)

        j += 1

    outliers = report_outliers(past_results)
    return outliers
```



Sequential Outlier Example:
http://localhost:8888/notebooks/AD03-S119-Sequential_Outlier_Example.ipynb

Outlier Ensembles (cont.)

- Meta algorithms that combine the outputs of multiple algorithms
- Examples from classification: boosting, stacking, bagging, subspace sampling

- **Independent ensembles** (e.g. bagging)
  - Algorithms are applied independently on subsets or subspaces of data
  - Result can be weighted combination of algorithms, average, majority vote etc.

Outlier Ensembles (cont.)

```
Algorithm IndependentEnsemble(Data Set: D,
                 Base Algorithms: A1 . . . Ar )
begin
    j = 1;
    repeat
        Pick an algorithm Aj ;
        Create a new data set fj(D) from D;
        Apply Aj to fj(D);
        j = j + 1;
    until(termination);
    report outliers based on combinations of
        results from previous executions;
end
```

```python
def IndependentEnsemble(D, A):
    j = 0
    results = []
    while not termination_condition(j, D):
        Aj = pick_algorithm(j)
        fj_D = create_new_dataset(D, j)

        results = apply_algorithm(Aj, fj_D)
        results.append(results)

        j += 1

    outliers = report_outliers(results)
    return outliers
```

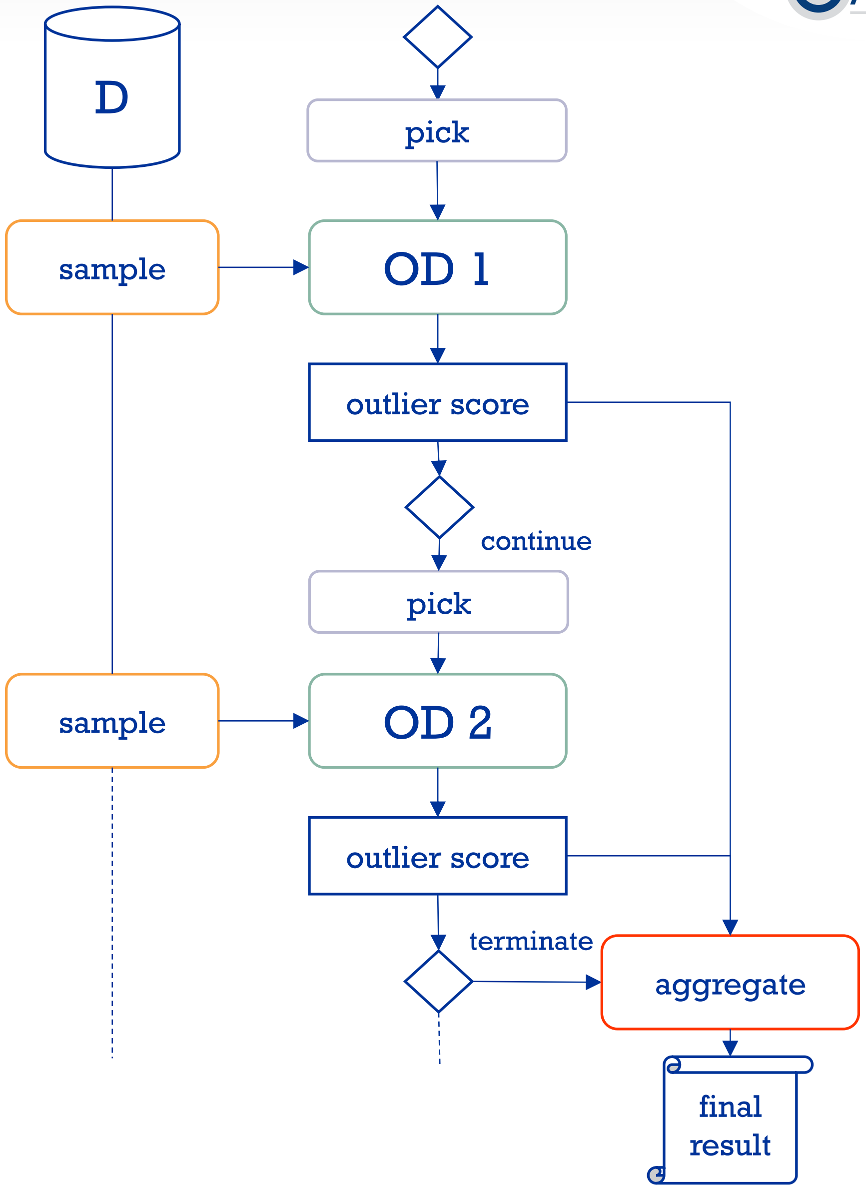
Outlier Ensembles (cont.)

```python
def IndependentEnsemble(D, A):
    j = 0
    results = []
    while not termination_condition(j, D):
        Aj = pick_algorithm(j)
        fj_D = create_new_dataset(D, j)

        results = apply_algorithm(Aj, fj_D)
        results.append(results)

        j += 1

    outliers = report_outliers(results)
    return outliers
```



High-Dimensional Outlier Detection

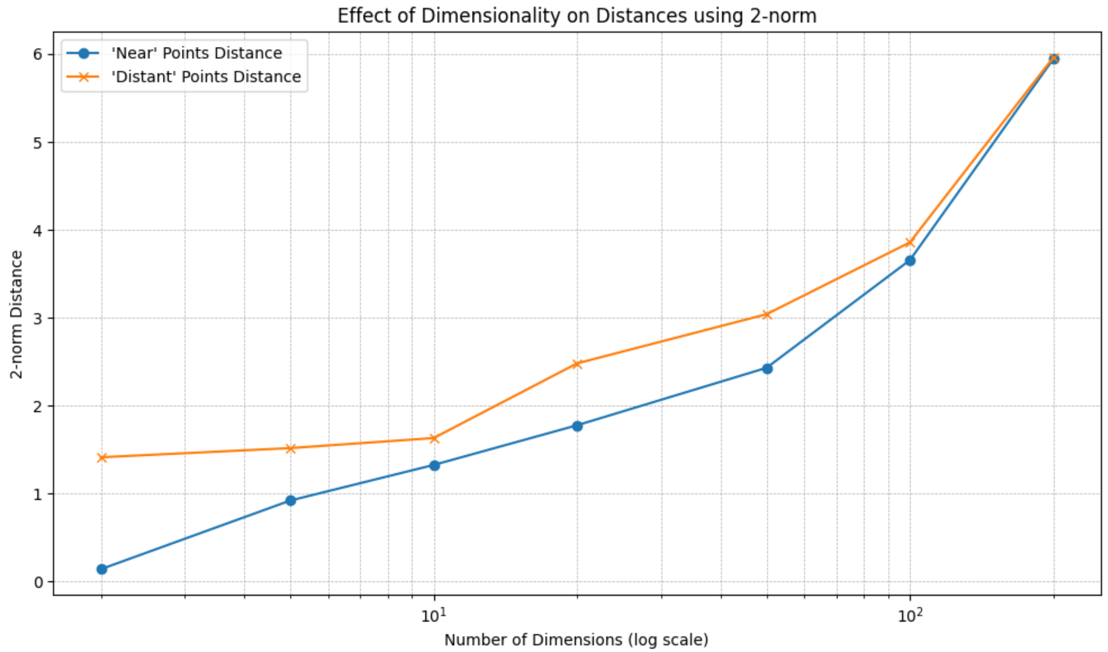
- Challanging as dimensions can be noisy and irrelevant
- Pairwise distances tend to become more similar (curse of dimensionality)
- In high-dimensional space all pairs of points become almost equidistant

Experiment:

- Two near points, two distant points in $\mathbb{R}^2$
- Add 2, 5, 10, 20, 50, 100, 200 dims. with random values
- Observe Minkowski metric

$$d(x,y) \coloneqq \|x-y\|_p = \sqrt[p]{\sum_{i=1}^{n}(x_i-y_i)^p}$$

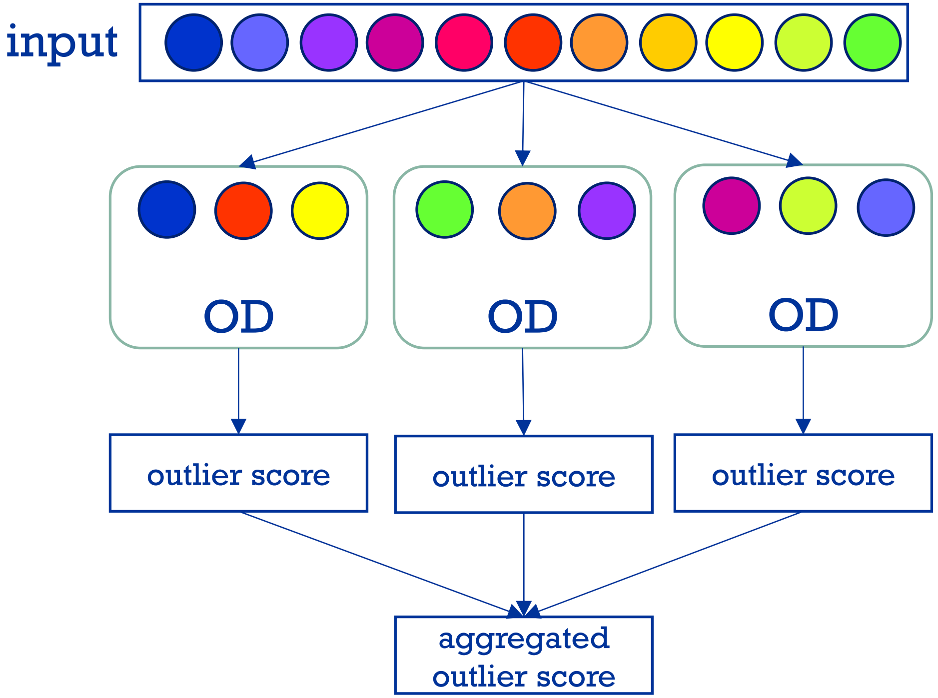
(Manhattan p=1, Euclidean p=2, ... Maximum p=∞)



High-Dimensional Outlier Detection

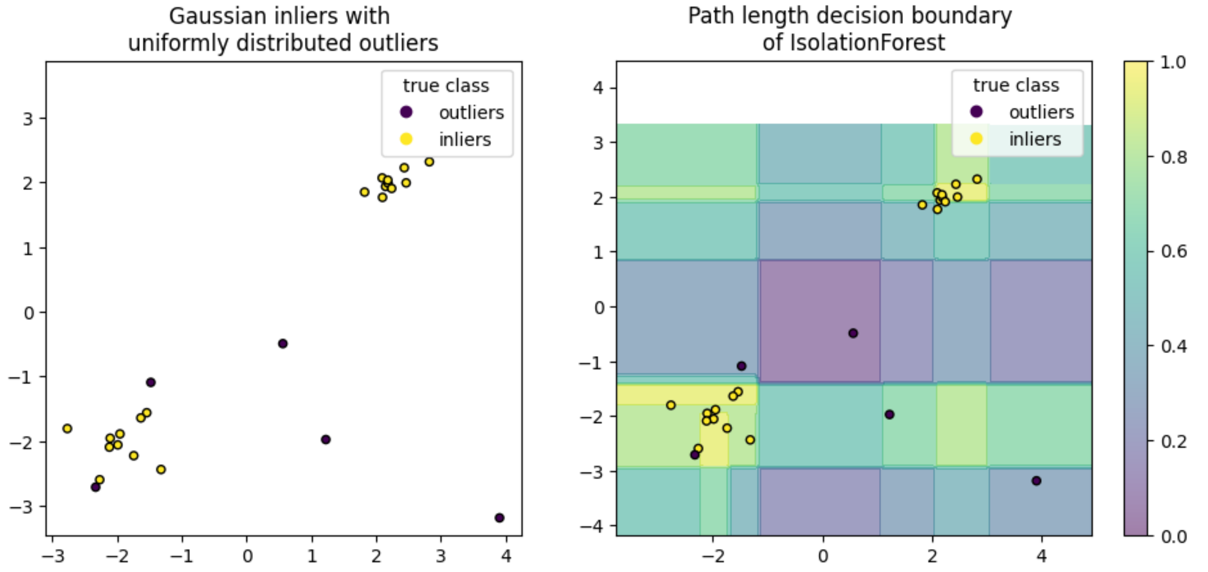
Solution:

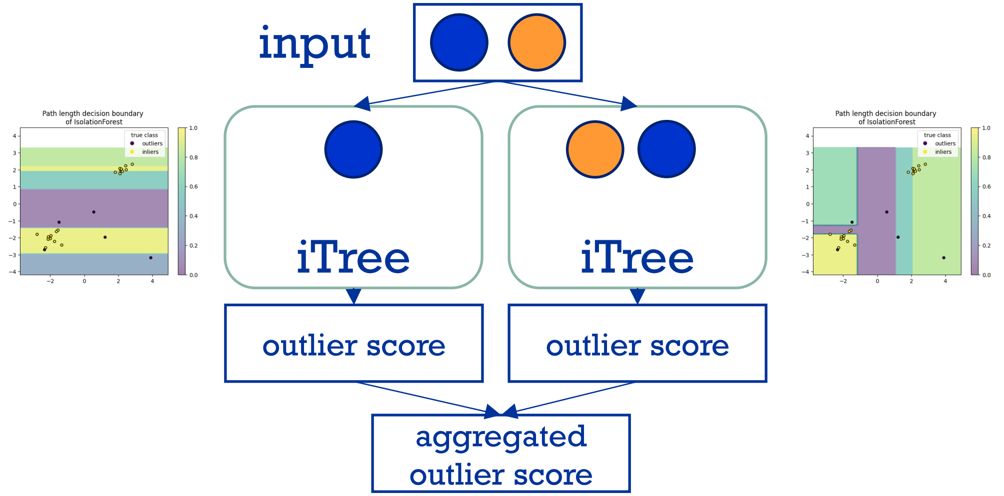
- Find lower-dimensional local subspace of relevant attributes (subspace outlier detection)
- Assumption: Outliers often hidden in unusual local behavior of low-dimensional subspaces
- Identify multiple relevant subspaces and combine predictions from these
- Closely related to outlier ensembles (bootstrap aggregation, "**bagging**")



High-Dimensional Outlier Detection (cont.)

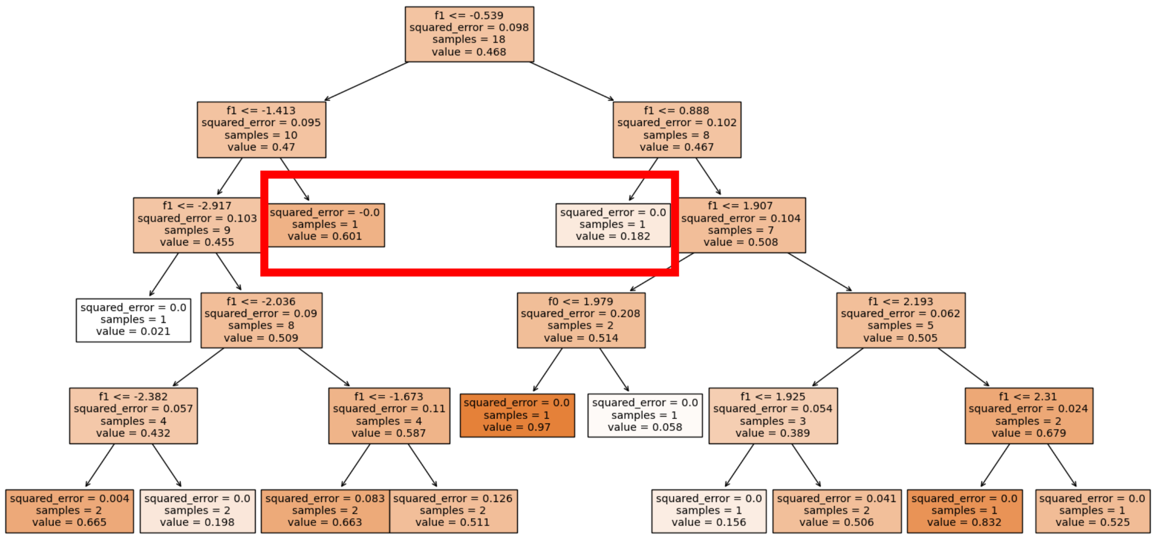
- Isolation Forest
  - Idea: Adapt random forest to AD
  - Construct random trees:
    Randomly select feature and split values until height limit is reached or all samples are isolated in leafs
  - Anomalies are isolated earlier in leafs than normal samples
  - Pathlength from root to respective leaf as anomaly score
  - Averaged over multiple trees
- Example:

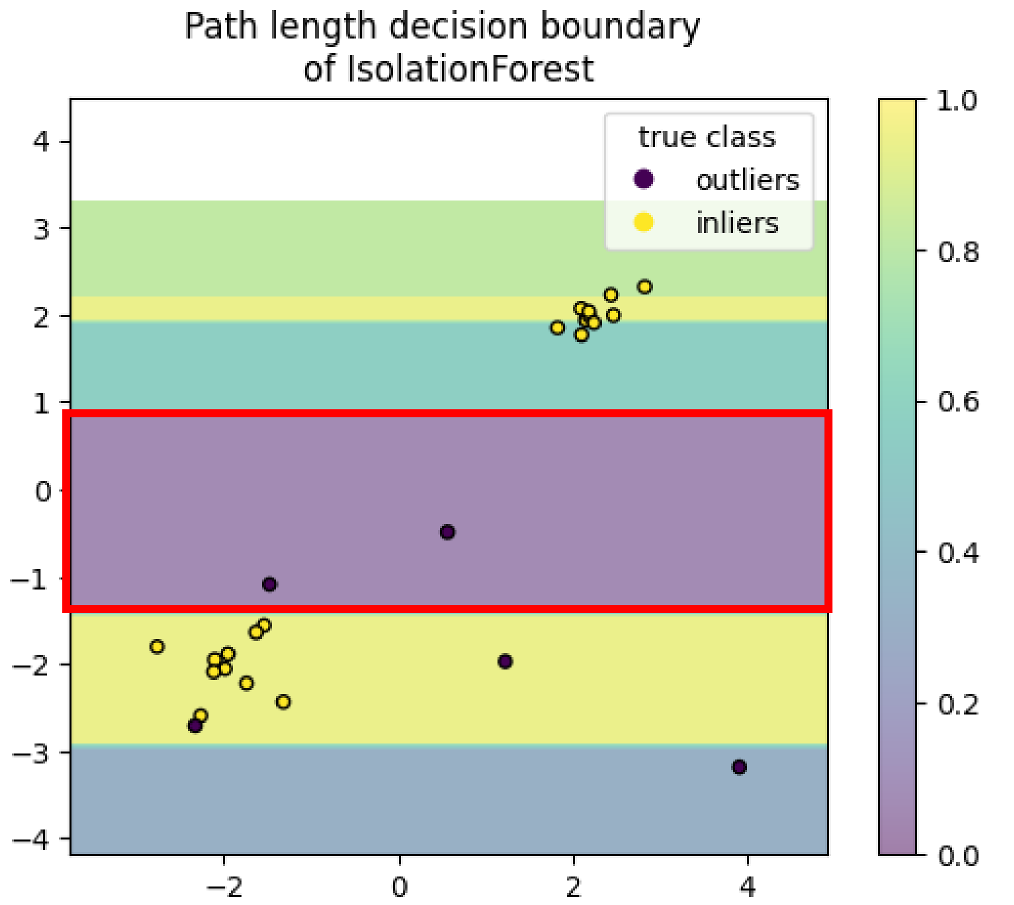




High-Dimensional Outlier Detection (cont.)

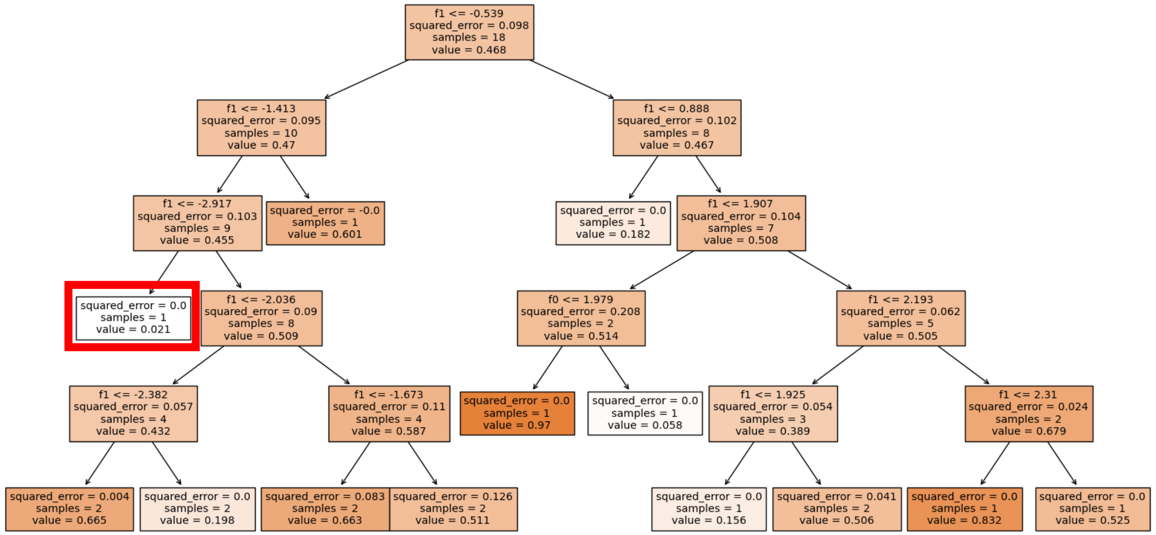
Tree 1:
Branch: -1.413 < f1<= 0.888

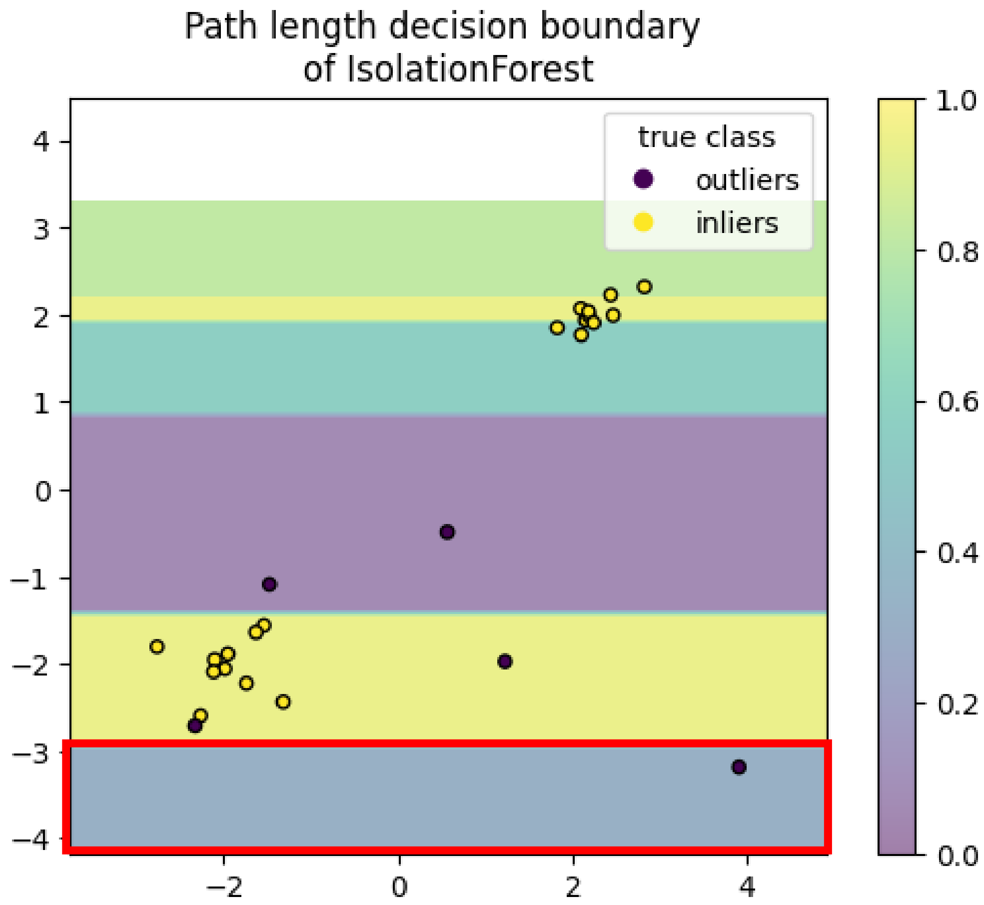




High-Dimensional Outlier Detection (cont.)

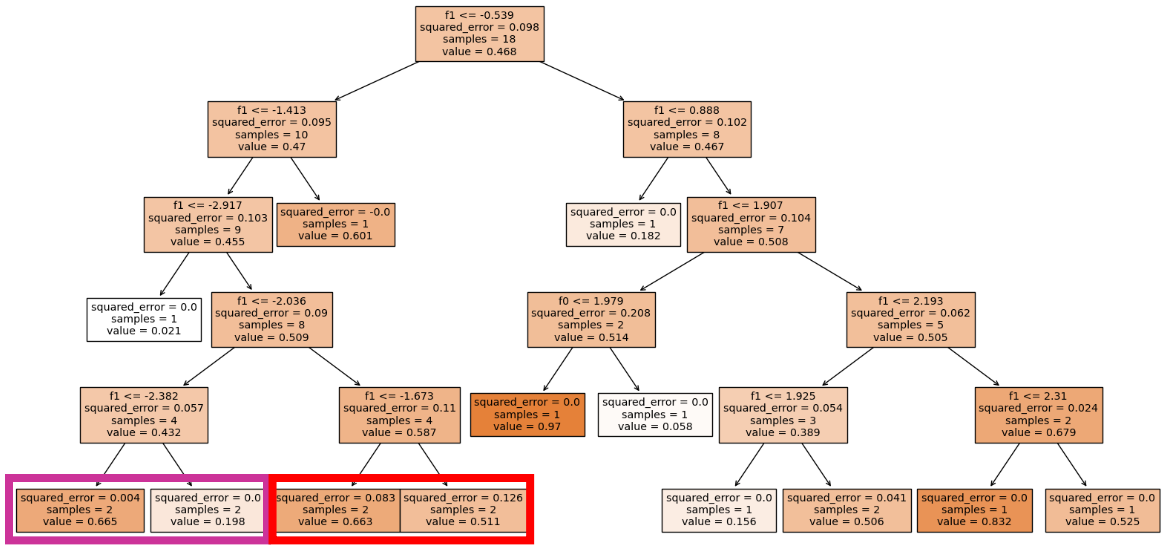
Tree 1:
Branch: f1 <= -2.917





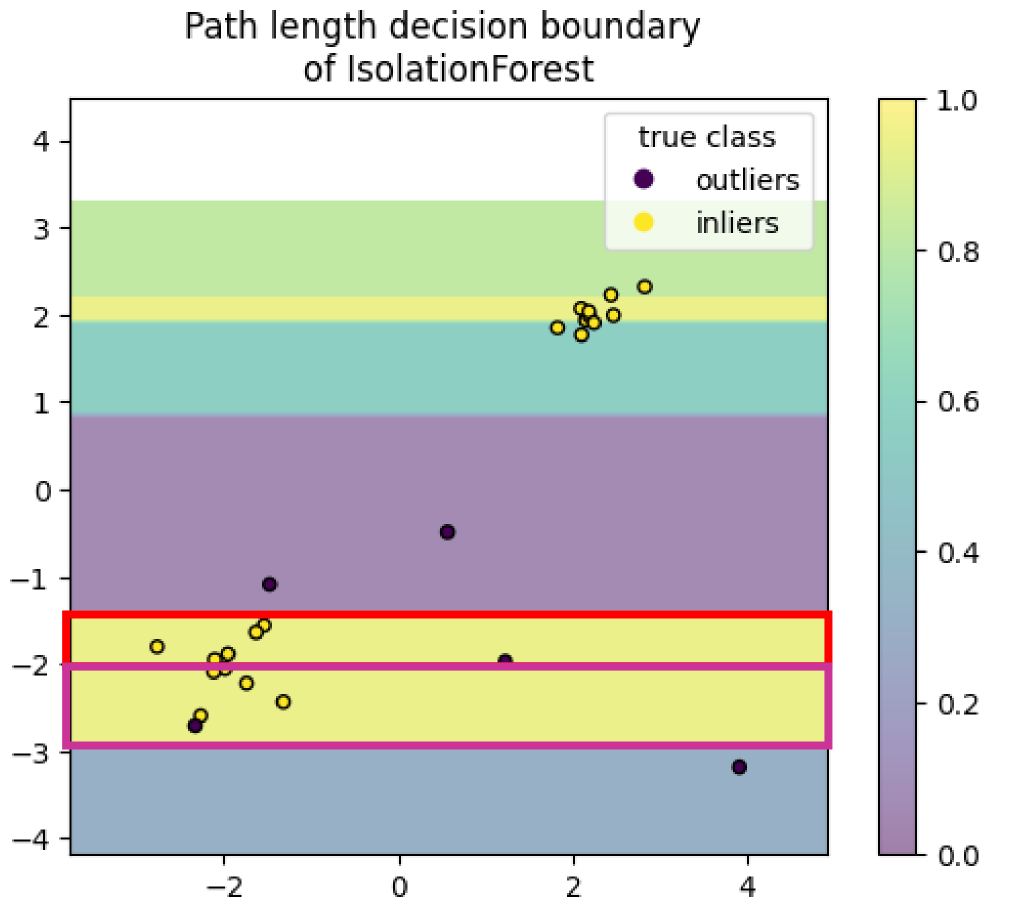
High-Dimensional Outlier Detection (cont.)

Tree 1:
Branch: -2.917 < f1 <= -1.413



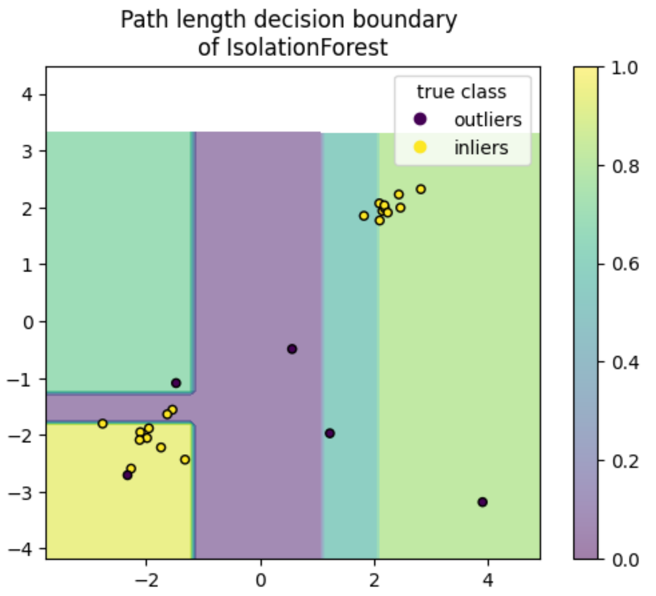
-2.917 <= f1 < 2.036

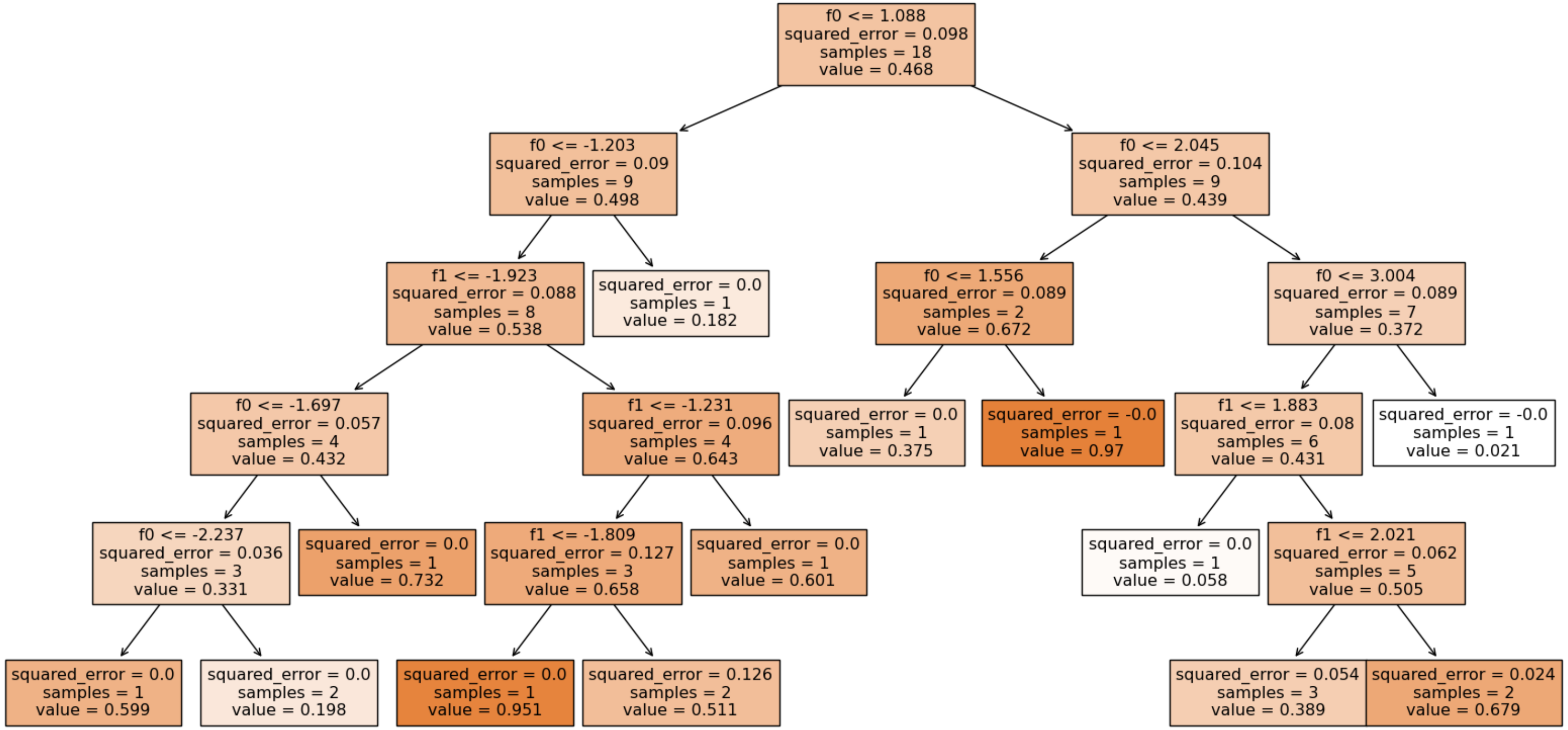
-2.036 <= f1 < -1.413



High-Dimensional Outlier Detection (cont.)

Tree 2:





Isolation Forest: Outlier Score

- Observation: Anomalies are isolated earlier in leafs than normal samples
- Pathlength from root to respective leaf as anomaly score
- Averaged over multiple trees
- Maximum possible hight grows in the order of n (#Samples), average grows in the order of $\log(n)$
- Normalization by average height of leaf (unsuccessful search path length in Binary Search Tree)

$h_i(x)$: height of leaf in which x is located for tree i

Average path length of unsuccessful seach:

$$c(n) = 2H(n-1) - \left(\frac{2(n-1)}{n}\right)$$

$H(i) \approx \ln(i) + 0.55721566$ (Euler's constant)

Outlier score:

$$s(x,n) = 2^{-\frac{\frac{1}{T}\sum_i^T h_i(x)}{c(n)}}$$