# Isolation Forest

Based on the work published here: Article Link

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In *2008 eighth ieee international conference on data mining* (pp. 413-422). IEEE.

# Introduction

- **Anomalies** are data patterns that have different data characteristics from normal instances.
- Most model-based anomaly detection approaches:
  - Construct a profile of normal instances.
  - Identify anomalies as instances that do not conform to this profile.
- **Drawbacks of existing methods**:
  - The anomaly detector is optimized for normal instances but not for detecting anomalies, leading to high false alarms.
  - Many methods are computationally expensive, making them suitable only for low-dimensional and small datasets.

# Introduction

- The method introduced is called **Isolation Forest** (iForest), suggested by Liu et al.
- iForest takes a unique approach by isolating anomalies directly instead of profiling normal data.
- No reliance on distance or density measures, eliminating computational costs associated with distance calculations.
- Linear time complexity with low memory requirements.
- Scalable to large datasets and high-dimensional problems with irrelevant attributes.
- Assumptions:
  - 1. Anomalies are the minority and consist of fewer instances.
  - 2. Anomalies have attribute values that are significantly different from normal instances.

# Basic Idea

- Construct an unsupervised decision tree which randomly partition the sample space with binary split until every single instance is isolated. Call this tree as **iTree**.
- Because of their susceptibility to isolation, anomalies will be isolated closer to the root of the tree; whereas normal points will be isolated at the deeper end of the tree.
- Build an **iForest** which is an ensemble of iTrees for sub-sampled, bootstrapped without replacement, data sets.
- Calculate the average path length of each data point and get anomaly score by using some metric.
- Identify data points as anomalies which have high anomaly score.

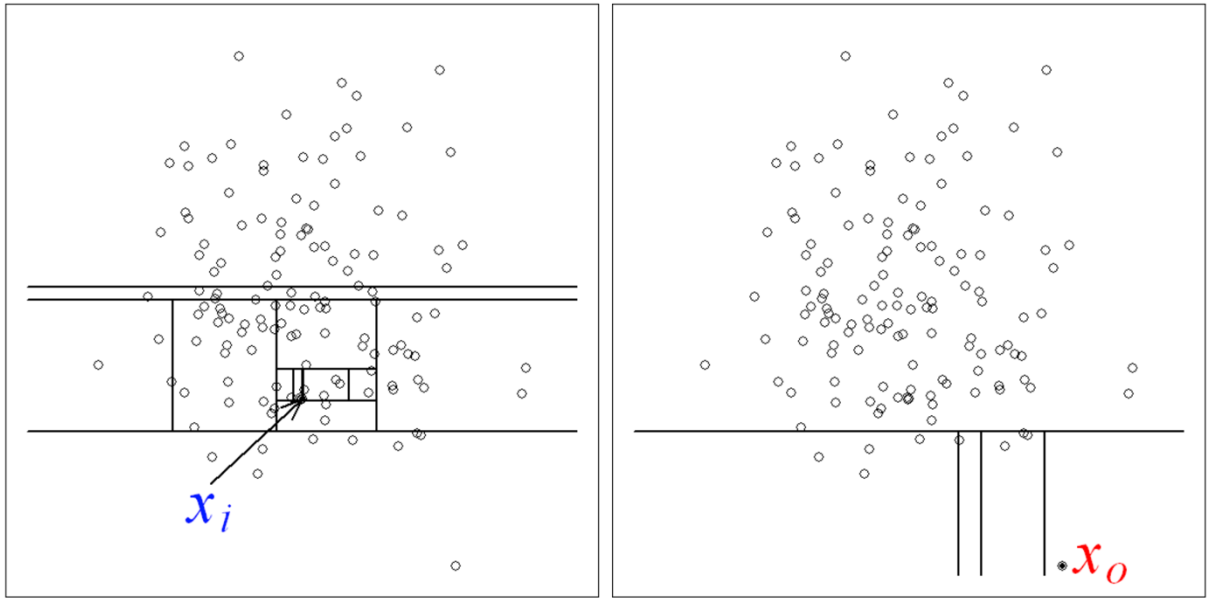
# Basic Algorithm

- Randomly selects a feature and splits on a random value.
- Repeats the process recursively, creating a tree structure.
- Calculates the path length for each data point.
- Averages path lengths across many trees to determine anomaly scores.
- Isolation Trees:
  - Random partitions of the data.
  - Nodes represent splits, leaves represent isolated points.
- Anomaly Score:
  - Points requiring fewer splits have shorter paths and higher anomaly scores.
- Key Parameters:
  - n_estimators: Number of trees.
  - max_samples: Number of samples per tree.
  - contamination: Proportion of expected anomalies.

# Each iTrees is built using sub-sampled, bootstrapped without replacement, data sets

- Sub-sampled Data:
  - Each tree in the forest is trained on a randomly selected subset (sub-sample) of the full dataset.
  - Sub-sampling ensures that each tree focuses on different portions of the data, adding diversity to the model.
- Bootstrapping Without Replacement:
  - When creating the sub-sampled dataset for a tree, instances are selected without replacement from the main dataset.
  - This means that a data point, once selected, is not included again in the same sample, ensuring the sub-sample is unique.
- This technique reduces the size of the dataset for each tree, making the process computationally efficient.
- It also helps isolate anomalies effectively by focusing on subsets of the data, as anomalies are more likely to stand out in smaller samples.

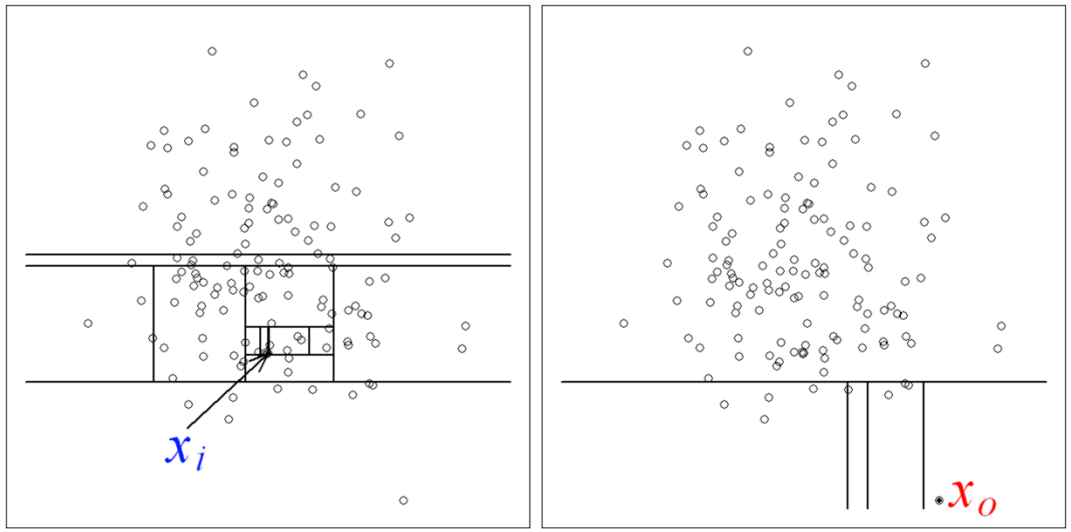
# The number of partitions required to isolate a normal point is greater than an anomaly



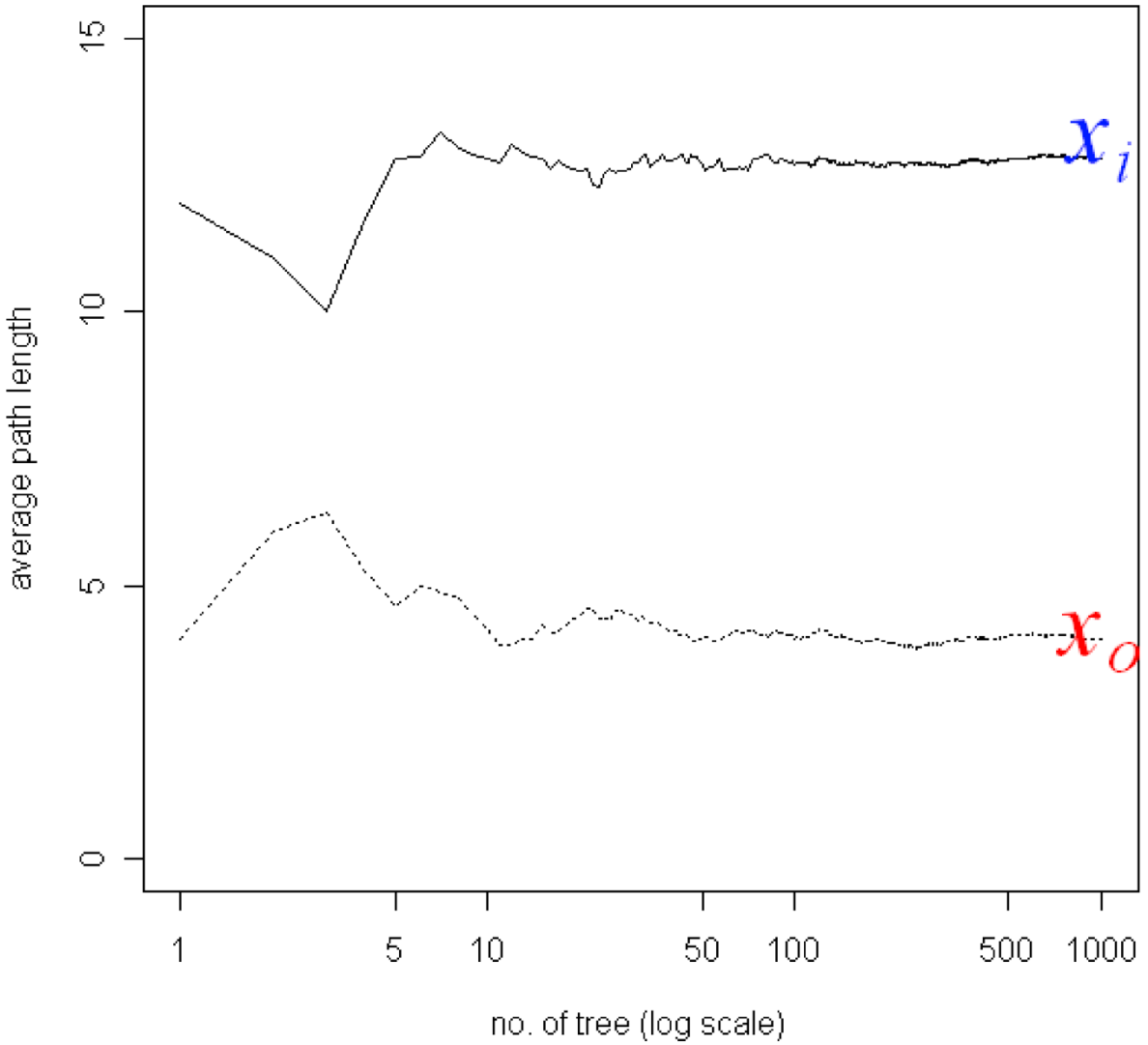

(a) Isolating $x_i$ (b) Isolating $x_o$

- Anomalies are more susceptible to isolation and hence have short path lengths
  - $x_i$ is a normal point and $x_o$ is an anomaly
- $x_i$ requires 12 random partitions to be isolated
- Whereas $x_o$ requires only 4 partitions to be isolated
- The number of partitions is equivalent to the path length from the root node to a terminal node in the tree

# Average path lengths converge



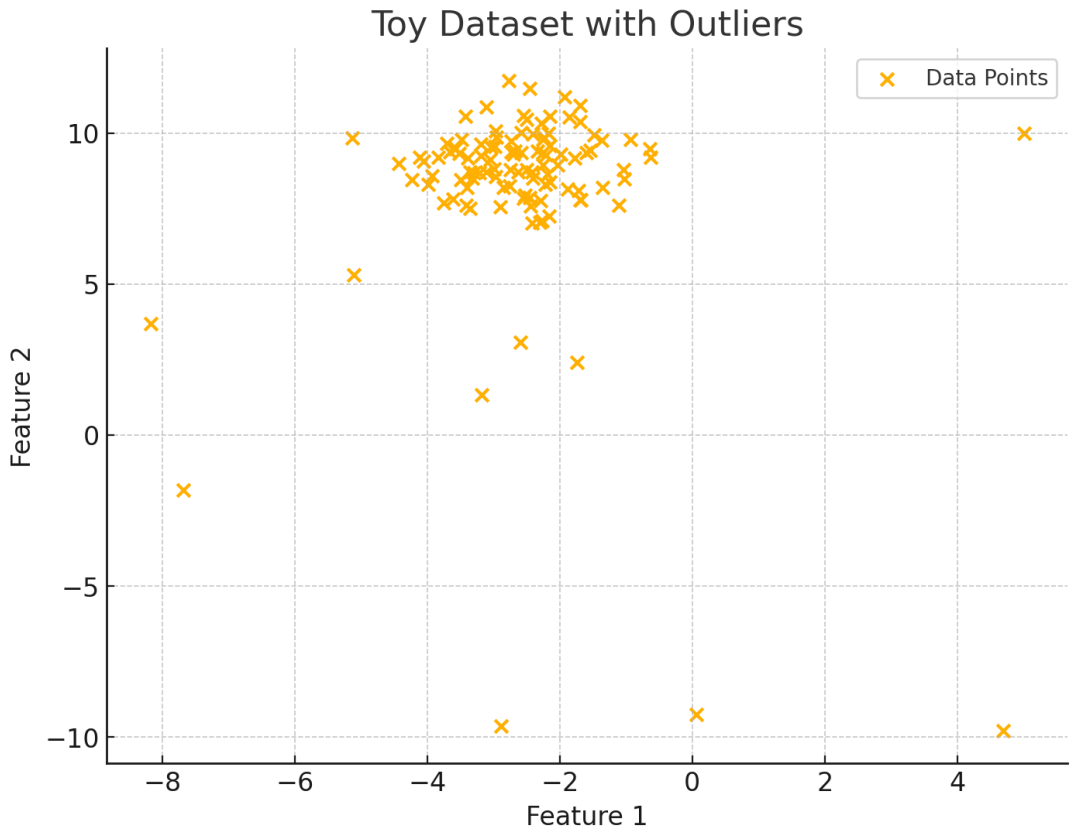

(a) Isolating $x_i$ (b) Isolating $x_o$

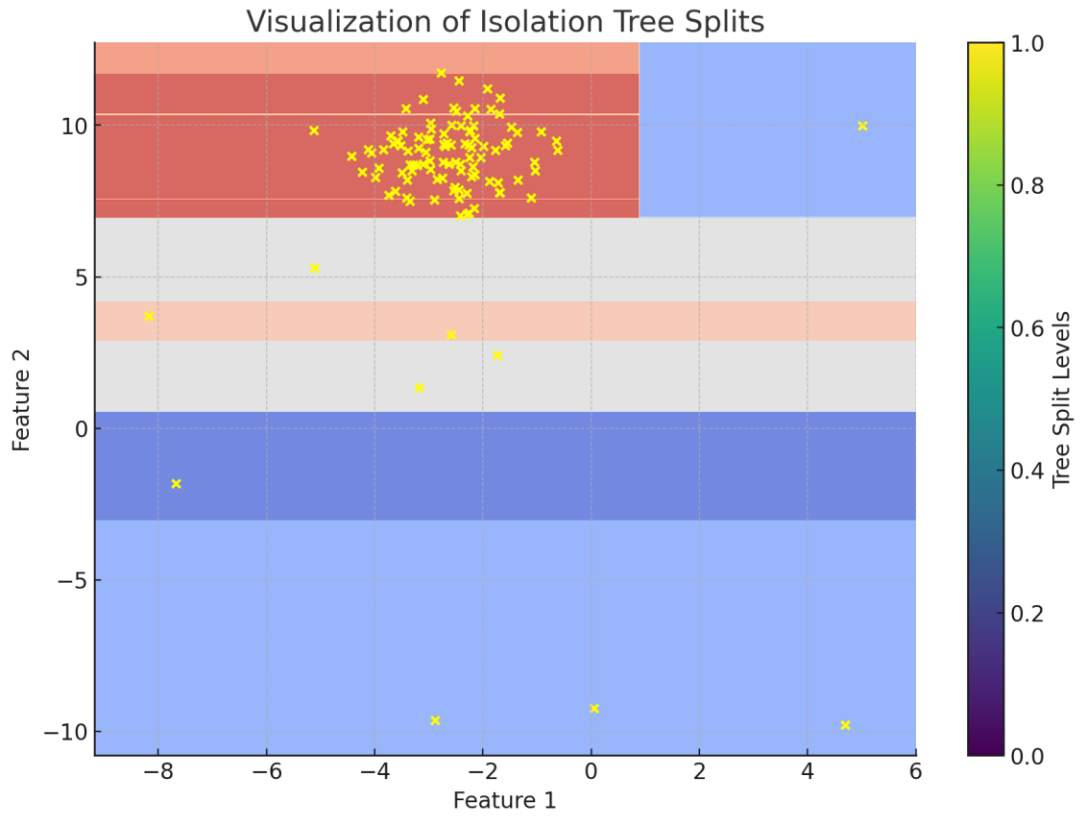


# Run using the sklearn.ensemble library

```
from sklearn.ensemble import IsolationForest

# Sample data
X = [[-1.1], [0.2], [0.5], [100], [0.3]]

# Train Isolation Forest
iso_forest = IsolationForest(contamination=0.2)
iso_forest.fit(X)

# Predict anomalies
predictions = iso_forest.predict(X)
print(predictions)  # Output: [1, 1, 1, -1, 1]
```

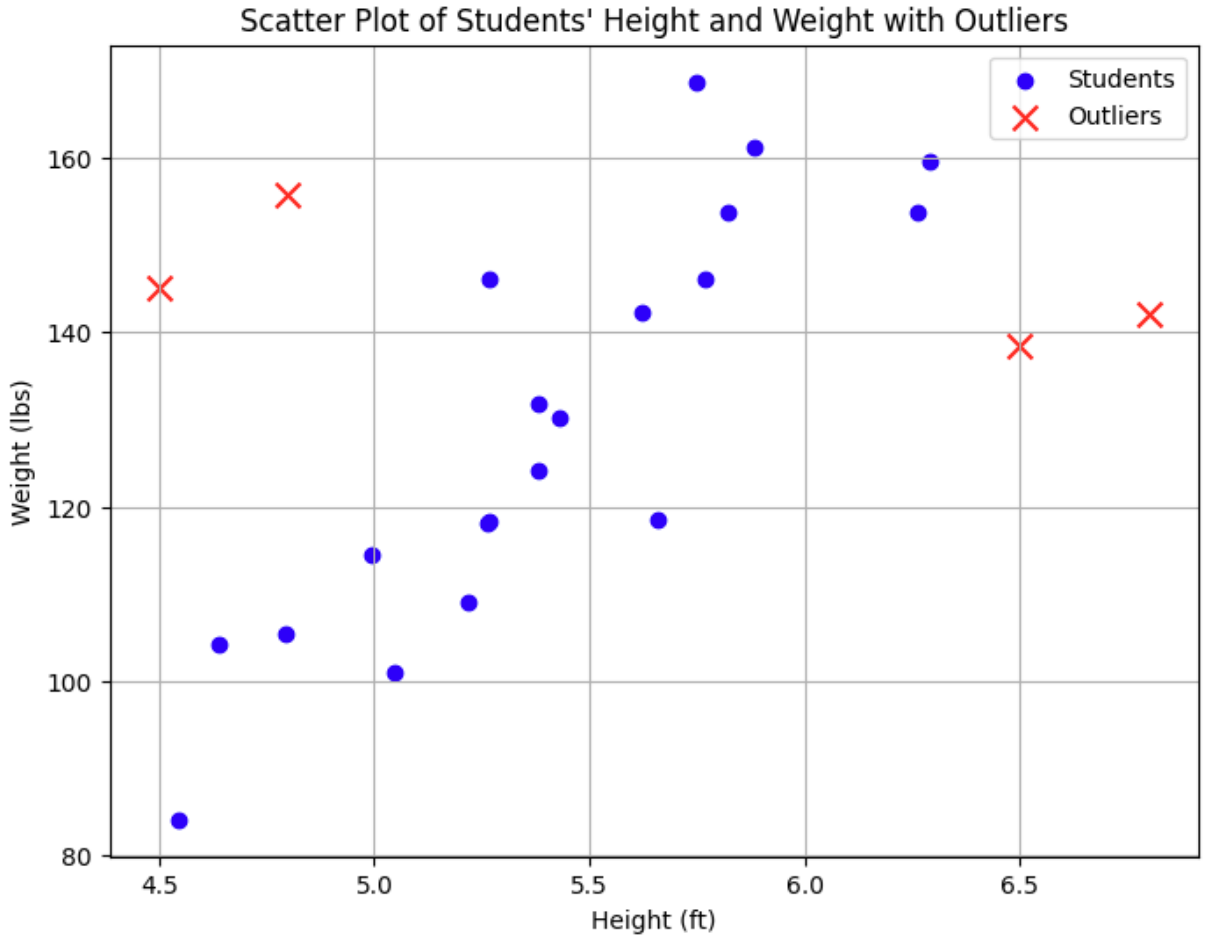
# Splits required to isolate are higher for outliers



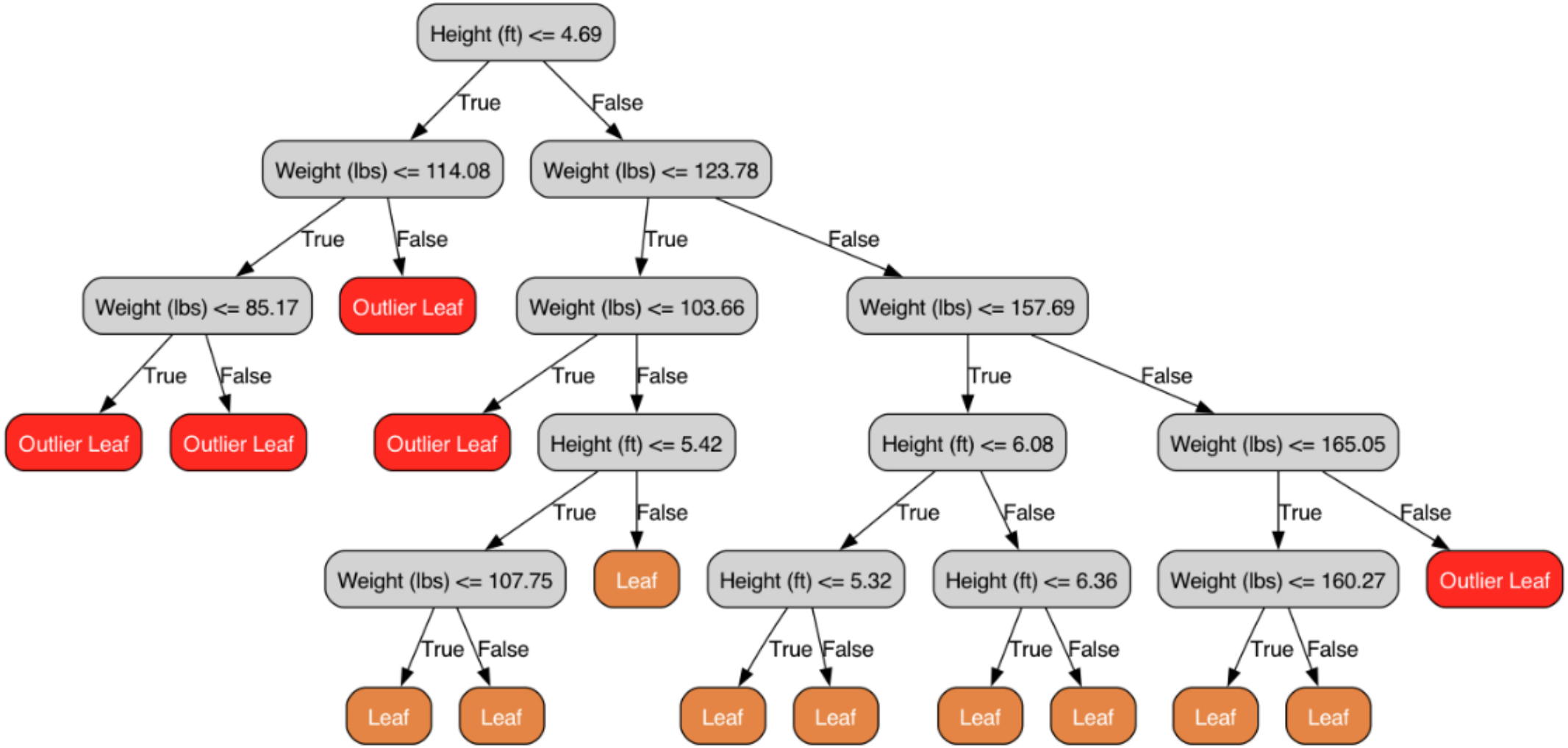



# Example with toy data

- We plot the height and weight of 24 students.
- This data was randomly generated



# Example iTree



# Example iForest