# Isolation Forest method implementation

**Implementing a method from the literature**
**Algorithmic Data Analysis – Coding Assignment 4**

Giulia Ortolani

March 31, 2024

**Resources**: No resources
**Collaborations**: No collaborations

## Isolation Forest Algorithm

Isolation Forest is an outlier detection method that isolates anomalies in a dataset. The algorithm implemented here is based on the methodology presented in [1]. Unlike methods that adopt density-based or distance-based approaches, the Isolation Forest algorithm works by randomly selecting attributes and split points during training to create partitions in the data.
The assumption is that anomalies are rare and therefore expected to require fewer splits to be isolated from the majority of the data points. When obtaining the tree, anomalies are expected to be closer to the root, while normal data points are deeper within the tree structure. This method uses a forest of trees created in this way.
For a given data point, we can measure the path length to reach it from the root for all trees in the forest and take the average of these values. Points with shorter average path lengths are then classified as anomalies. This process allows for the detection of outliers without the need for explicit labeling of anomalies during training.
The randomness in selecting attributes and split points allows the algorithm to isolate anomalies more efficiently, making it particularly suitable for large-scale datasets with high-dimensional feature spaces.

## Dataset presentation

The Cardiac Arrhythmia Database[1] is a comprehensive collection of data aimed at understanding and classifying various types of cardiac arrhythmias. This dataset serves as a valuable resource for studying the complexities of heart rhythm disorders. Containing a total of 279 attributes and 452 observations, the Cardiac Arrhythmia Database offers a wealth of information crucial for understanding cardiac health. Attributes range from basic demographic data such as age and sex to intricate ECG measurements, providing researchers with a comprehensive view of each patient's condition.
In this analysis, I have decided to focus on a subset of attributes deemed most relevant for our research. These attributes include demographic information such as age, sex, height, and weight, as well as key ECG measurements like QRS duration, P-R interval, and Q-T interval. The selected attributes aim to capture essential features necessary for the accurate classification of arrhythmias.
The dataset includes instances classified into 16 distinct classes of arrhythmia, with varying frequencies among each class. Instances have been anonymized to ensure patient privacy, with missing attribute

[1] `https://archive.ics.uci.edu/dataset/5/arrhythmia`

values marked for clarity. Missing values are a challenge that requires attention, as they can potentially affect the integrity of our analysis. To ensure the smooth functioning of our code and enable preliminary analysis, missing attribute values have been initialized to -1. However, this may not be the best solution, and resolving missing values is a critical point in this code.

## Implementation of the Isolation Forest Algorithm

This code implements an Isolation Forest algorithm for anomaly detection. I've decided to organize the code in the following way. Anomaly detection using iForest is a two-stage process: the first (training) stage builds isolation trees using sub- samples of the training set; the second (testing) stage passes the test instances through isolation trees to obtain an anomaly score for each instance.

### Main

This program reads the dataset, splits it into training and testing sets to perform the training and evaluating stage. The splitting is done by randomly selecting 80% of the instances for the training set and the other 20% for the testing set. Then, it builds the isolation forest using the training set (by calling the `iForest` function) and use the forest to compute anomaly scores for the testing set, returning the top anomalies.
The top 5 anomaly scores obtained for this dataset are:

0.3973
0.3906
0.3729
0.3639
0.3602

Anomaly scores closer to 1 indicate a higher likelihood of being an anomaly, while scores closer to 0 indicate a lower likelihood.

### Training Stage

The `iTree` function builds an isolation tree recursively. It randomly selects an attribute and a split point for each internal node until a maximum tree height (`l`) or a minimum subset size (`e`) is reached. The function `iForest` creates a forest of isolation trees by repeatedly calling `iTree` with random subsets of the training data.

### Evaluating Stage

The `anomaly-score` function computes an anomaly score for each data point based on the average path length in the forest.
After that, the `find-top-anomalies` function sorts the data by anomaly score and returns the top anomaly values.

Following the analysis of [1], I have refined my anomaly search to focus on specific classes. Specifically, I have narrowed down the search to classes 03, 04, 05, 07, 08, 09, 14, and 15. These classes correspond to the following labels: Old Anterior Myocardial Infarction, Old Inferior Myocardial Infarction, Sinus tachycardia, Ventricular Premature Contraction (PVC), Supraventricular Premature Contraction, Left bundle branch block, Left ventricular hypertrophy and Atrial Fibrillation or Flutter.

| Class code | Class | Number of instances |
|---|---|---|
| 03 | Old Anterior Myocardial Infarction | 15 |
| 04 | Old Inferior Myocardial Infarction | 15 |
| 05 | Sinus tachycardia | 13 |
| 07 | Ventricular Premature Contraction (PVC) | 3 |
| 08 | Supraventricular Premature Contraction | 2 |
| 09 | Left bundle branch block | 9 |
| 14 | Left ventricular hypertrophy | 4 |
| 15 | Atrial Fibrillation or Flutter | 5 |

Based on this class distribution, the selected classes (03, 04, 05, 07, 08, 09, 14, and 15) contain a total of 66 instances out of 424 instances in the dataset, accounting for approximately 15.57% of the dataset. These classes have been chosen as potential indicators of anomalies in the dataset.

The paper indicates that these selected classes contain approximately 15% of anomalies. I set this threshold to select a proportionate amount of data from these classes and classify them as anomalies (*this is not implemented yet in the code, I'm not sure that this is the right idea*).

## References

[1] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation-based anomaly detection". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1 (2012), p. 3.