CSC730: Assignment 7
Isolation Forests - Anomaly Detection by Isolating Anomalies

Kristophor Ray Jensen
Electrical Engineering and Computer Science
South Dakota School of Mines and Technology
Rapid City, United States
0009-0001-7344-349X

I. Introduction

Anomaly detection is a crucial task in various domains, from fraud detection in financial transactions to identifying rare diseases in medical diagnosis. Traditional anomaly detection methods often focus on profiling normal instances and identifying instances that deviate from this normal profile. However, a novel approach called Isolation Forest (iForest) takes a fundamentally different approach by explicitly isolating anomalies instead of profiling normal points [1].

The requirements for this assignment are as follows[2].

1) Get the provided dataset from D2L.
2) Write your own version of isolation forest code.
3) Run your code on the dataset to obtain anomalousness scores for each point for your chosen parameter settings.
4) Sort the points by anomalousness scores and generate a precision-recall curve.
5) Generate the equivalent of the following figure from your forest.

In this assignment, we implement the Isolation Forest algorithm from scratch and evaluate its performance on a dataset provided by the instructor. The dataset contains normal and anomalous instances, and our goal is to assess the effectiveness of the Isolation Forest algorithm in detecting anomalies.

II. Methodology

A. Dataset

The dataset used in this assignment was provided by the instructor. It consists of a set of instances, each labeled as either normal or anomalous. The specific details of the dataset, such as the number of features and the distribution of normal and anomalous instances, were not disclosed.

Figure 1 shows a random selection of 16 nominal images from the training set, while Figure 2 displays a random selection of 16 anomalous images from the training set. Similarly, Figure 3 and Figure 4 present random selections of nominal and anomalous images from the test set, respectively.

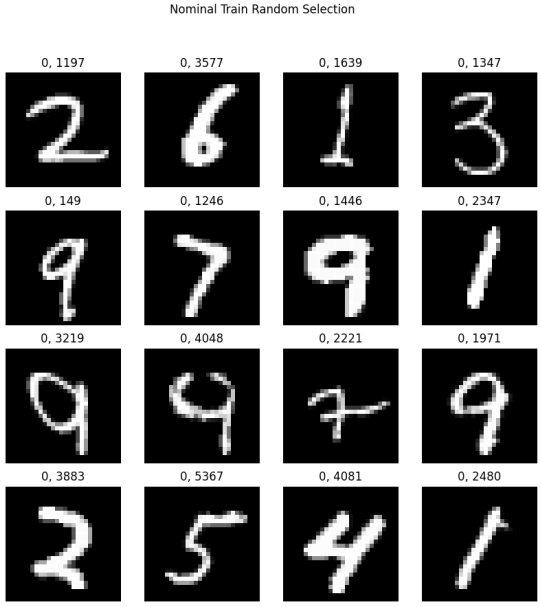

Figure 1. Random selection of nominal images from the training set

B. Isolation Forest Implementation

We implement the Isolation Forest algorithm from scratch using Python, Jupyter notebook, and VS code. The algorithm consists of two main components: Isolation Tree (iTree) and Isolation Forest (iForest). There is an additional class called isolation_tree_node that contains the detailed data of each node in the iTree.

An iTree is a binary tree structure that recursively partitions the data based on randomly selected features and split points until instances are isolated or a maximum tree height is reached. The key idea is that anomalies require fewer partitions on average to be isolated compared to normal instances.

The iForest is an ensemble, a forest of trees, of iTrees. It

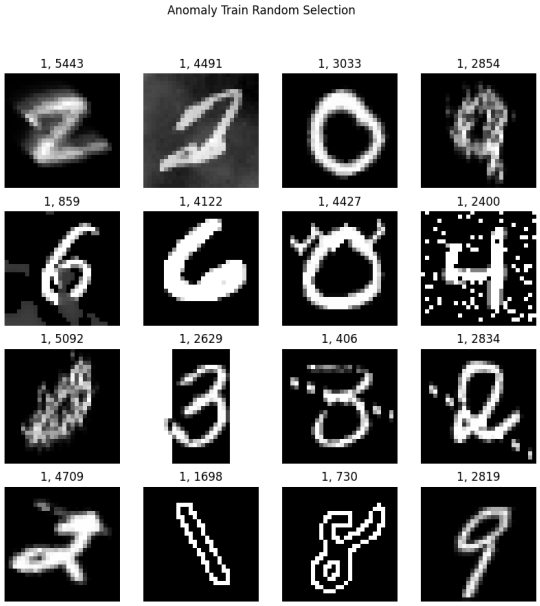

Figure 2. Random selection of anomalous images from the training set

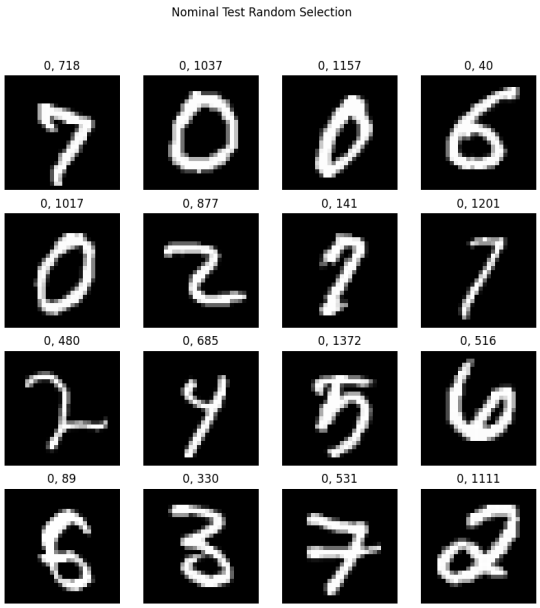

Figure 3. Random selection of nominal images from the test set

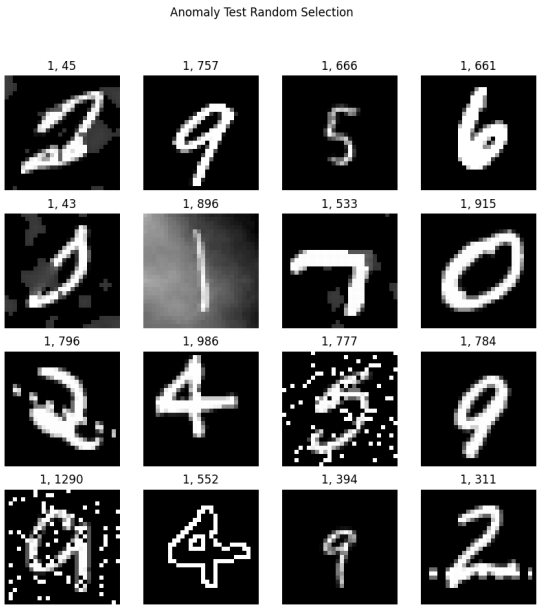

Figure 4. Random selection of anomalous images from the test set

constructs multiple iTrees using subsamples of the training data and aggregates the anomaly scores obtained from each iTree to make the final prediction.

The anomaly score of an instance is calculated based on the average path length it takes to be isolated in the iForest. Anomalies tend to have shorter average path lengths compared to normal instances.

Additionally, a visualization of an iTree is shown in Figure **??**. Each node represents a partition of the data based on a randomly selected feature and split point, with leaf nodes representing the isolated instances. Anomalies are typically isolated closer to the root of the tree.

This visualization was implemented using the Graphviz library to generate a graphical representation of the iTree structure.

C. Evaluation Metrics

To evaluate the performance of the Isolation Forest algorithm, we use the area under the Receiver Operating Characteristic (ROC) curve and the Precision-Recall (PR) curve. The ROC curve plots the true positive rate against the false positive rate at various threshold settings, while the PR curve plots precision against recall. Higher areas under these curves indicate better anomaly detection performance.

III. Results and Discussion

A. Anomaly Detection Performance

After changing to using `isolation_forest.anomaly_score()` instead of `.predict()`, our scratch implementation of the Isolation Forest algorithm achieved a PR curve area of 0.43 and

an ROC-AUC of 0.673. In comparison, the scikit-learn implementation yielded a PR curve area of 0.51 and an ROC-AUC of 0.72.

Figure 5 shows the PR curve for the scikit-learn implementation, while Figure 6 displays the corresponding ROC curve. Similarly, Figure 7 presents the PR curve for our scratch implementation, and Figure 8 shows the ROC curve.

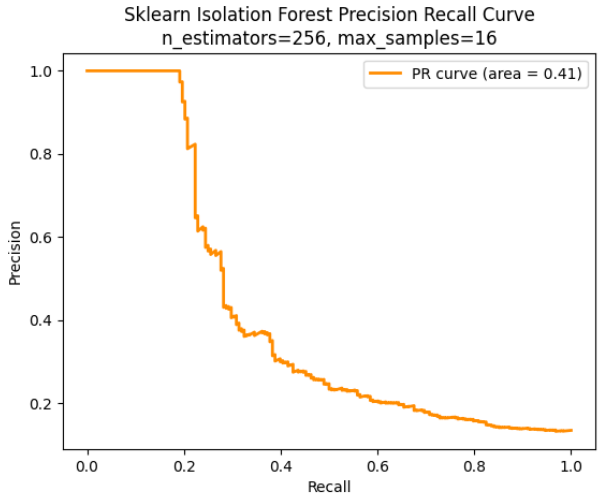

Figure 5. PR curve for the scikit-learn implementation

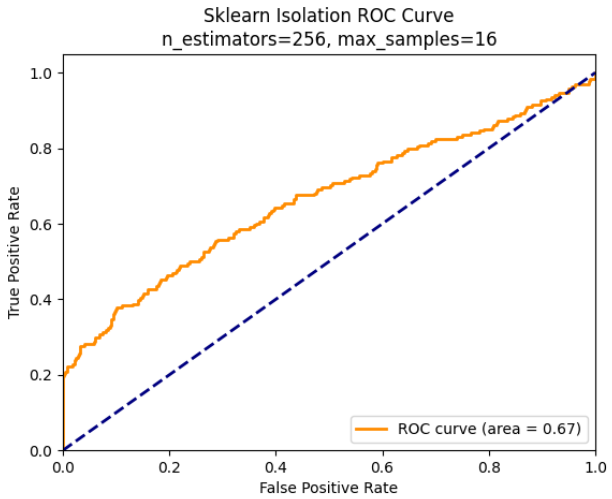

Figure 6. ROC curve for the scikit-learn implementation

B. Visualization of Isolation Forest

Figure 15 depicts an example of an iTree, where each node represents a partition of the data based on a randomly selected feature and split point. The leaf nodes represent the isolated instances, with anomalies typically being isolated closer to the root of the tree.

The visualization helps understand how the Isolation Forest algorithm partitions the data and isolates anomalies.

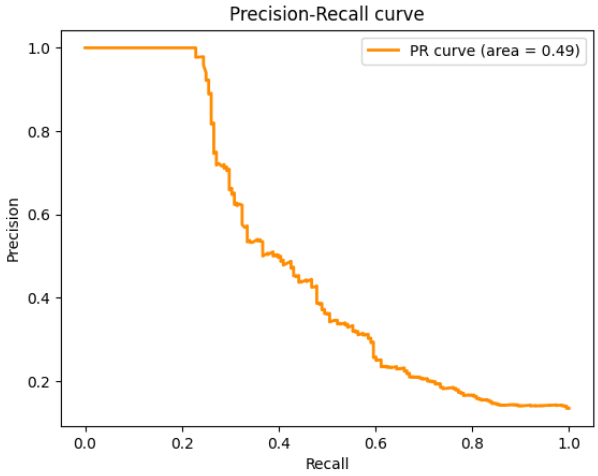

Figure 7. PR curve for our scratch implementation

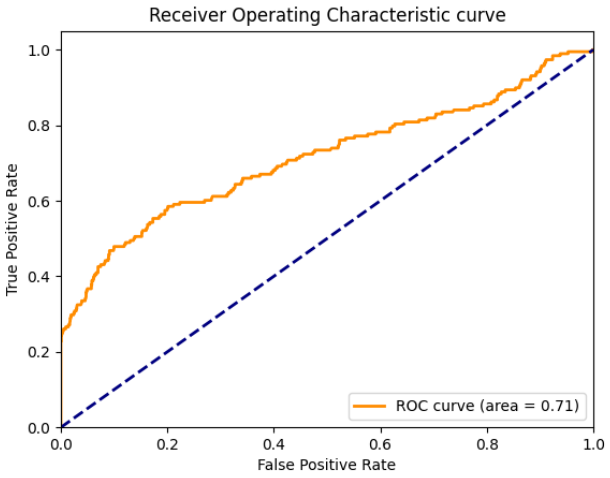

Figure 8. ROC curve for our scratch implementation

C. Dimensionality Reduction

To visualize the anomaly scores in a 2-dimensional space, we applied dimensionality reduction techniques such as t-SNE and PCA to the dataset prior to fitting the Isolation Forest algorithm.

With t-SNE using 2 components, the resulting tree heights were 6. The PR curve area was 0.33, and the ROC-AUC was 0.46. The performance was poor, and the contour plot (Figure 12) shows that the nominal and anomaly points are mixed together.

With PCA using 2 components, the resulting tree heights were approximately 10. The PR-AUC was 0.34, and the ROC-AUC was 0.62. The contour plot (Figure 16) shows mixing of nominal and anomaly points, but there is some distinction.

The dimensionality reduction techniques did not significantly improve the anomaly detection performance of the Isolation Forest algorithm on this dataset.

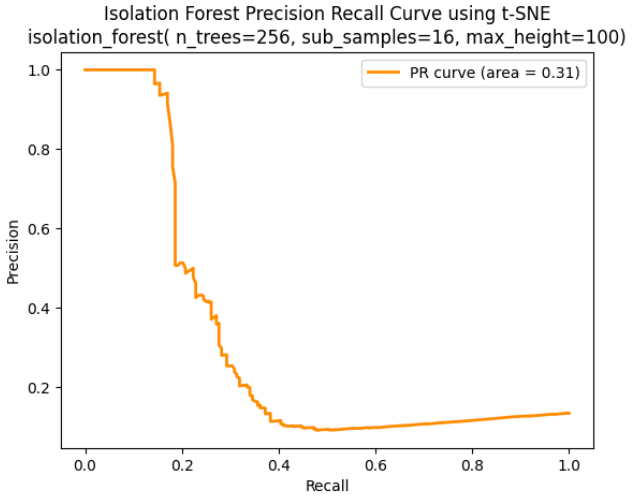

Figure 9. PR curve for t-SNE with 2 components

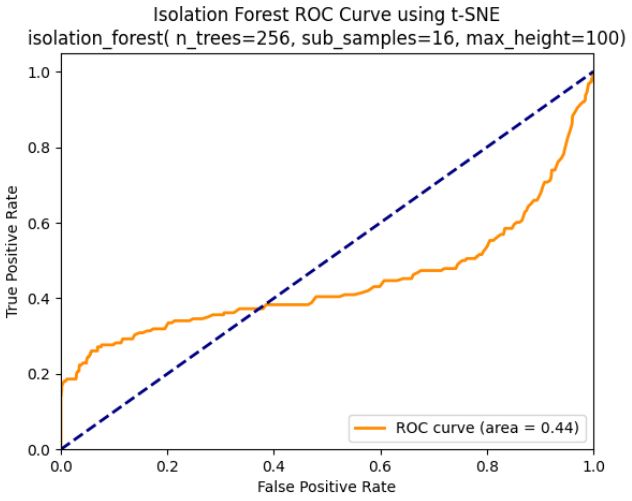

Figure 10. ROC curve for t-SNE with 2 components

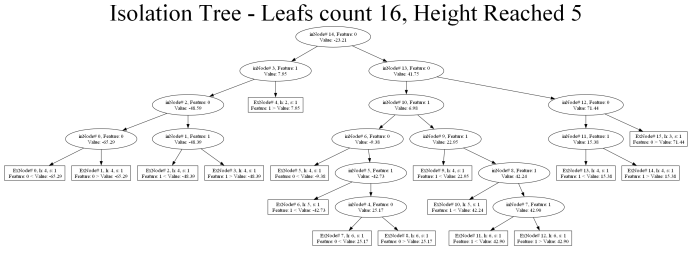

Figure 11. Visualization of an iTree with t-SNE

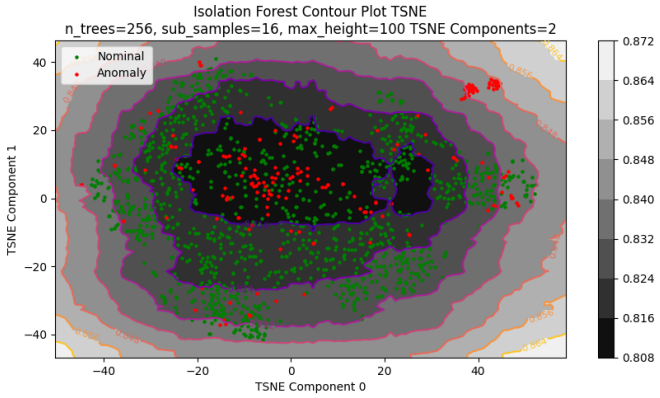

Figure 12. Contour plot of anomaly scores with t-SNE

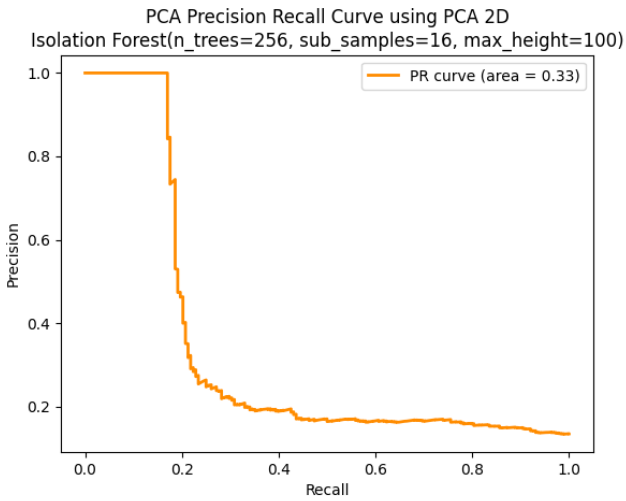

Figure 13. PR curve for PCA with 2 components

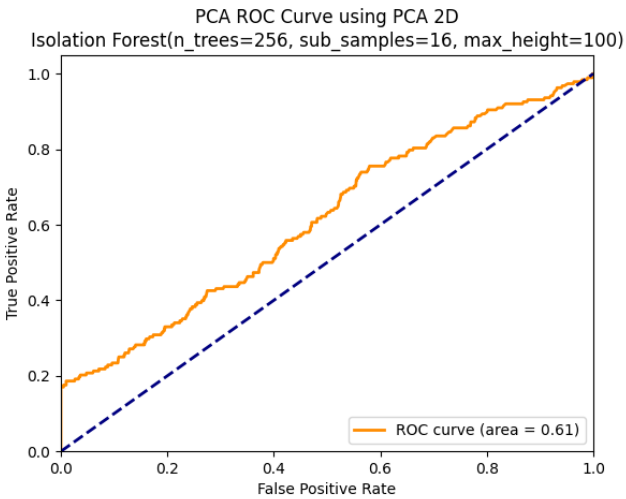

Figure 14. ROC curve for PCA with 2 components

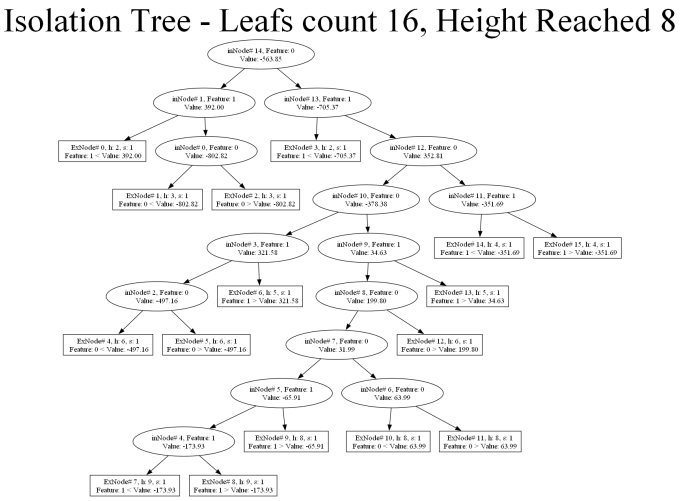

Figure 15. Visualization of an iTree with PCA

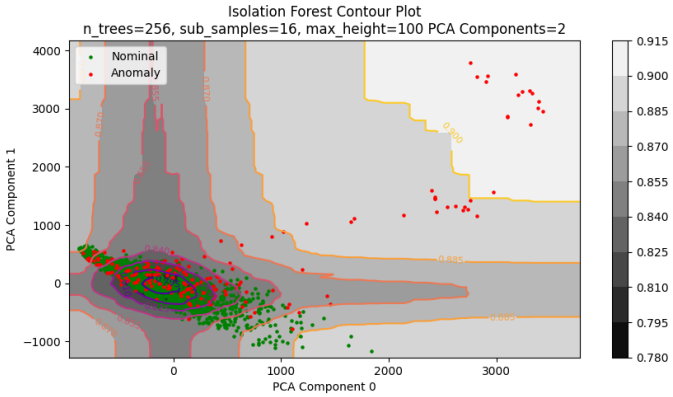

Figure 16. Contour plot of anomaly scores with PCA

IV. Conclusion

In this assignment, we implemented the Isolation Forest algorithm from scratch and evaluated its performance on a dataset provided by the instructor. The results showed that our scratch implementation achieved a PR curve area of 0.43 and an ROC-AUC of 0.673, which was slightly lower than the scikit-learn implementation.

We also explored dimensionality reduction techniques, such as t-SNE and PCA, to visualize the anomaly scores in a 2-dimensional space. However, these techniques did not significantly improve the anomaly detection performance of the Isolation Forest algorithm on this dataset.

The visualization of the Isolation Forest provided insights into how the algorithm partitions the data and isolates anomalies. The contour plots of the anomaly scores with t-SNE and PCA showed mixing of nominal and anomaly points, indicating the challenges in effectively separating them in the reduced-dimensional space.

Future work could involve exploring alternative anomaly detection techniques or investigating ways to preprocess the data to improve the performance of the Isolation Forest algorithm. Additionally, a more in-depth analysis of the dataset characteristics and the specific challenges it poses for anomaly detection could provide valuable insights for developing more robust and effective anomaly detection methods.

References

[1] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[2] R. Loveland, "Assignment_7.pdf," From SDSMT D2L Website, 2024.

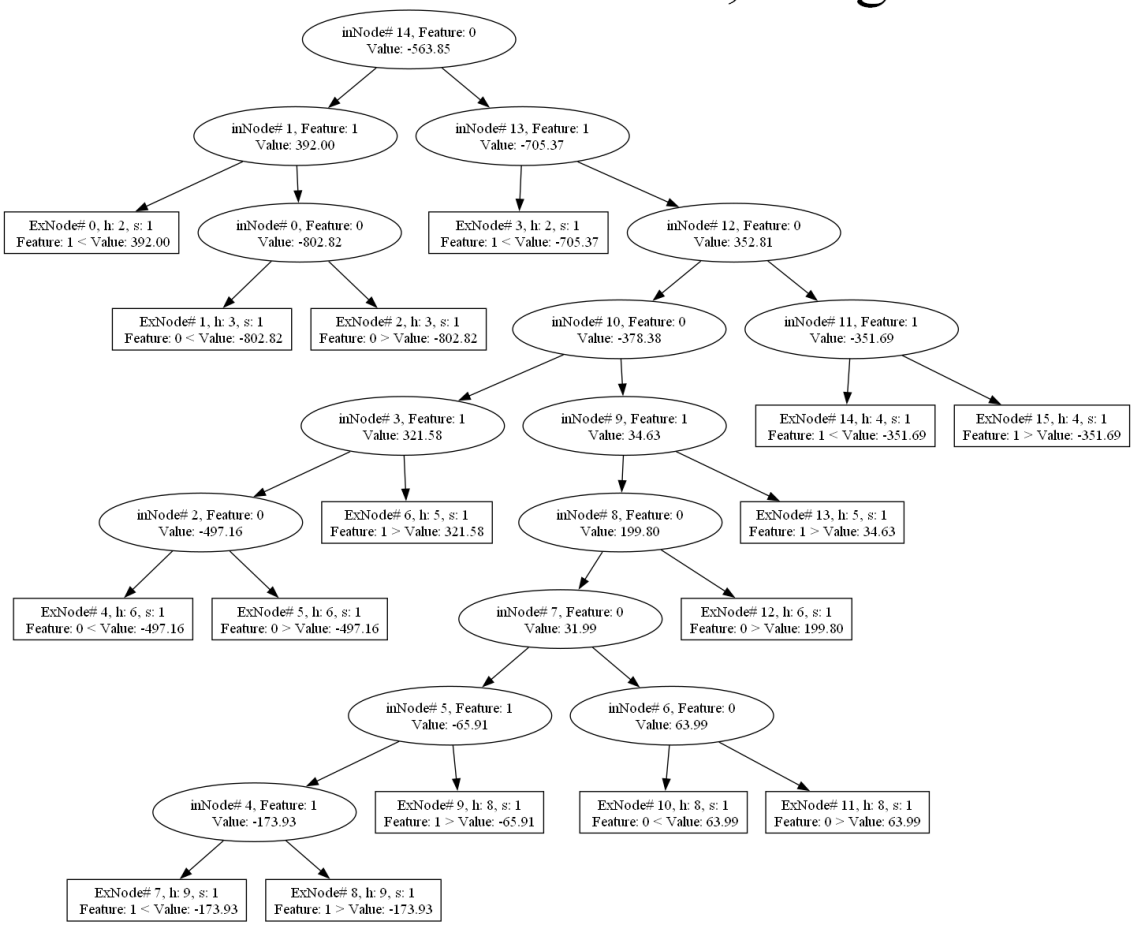

Figure 17. Visualization of an iTree with PCA

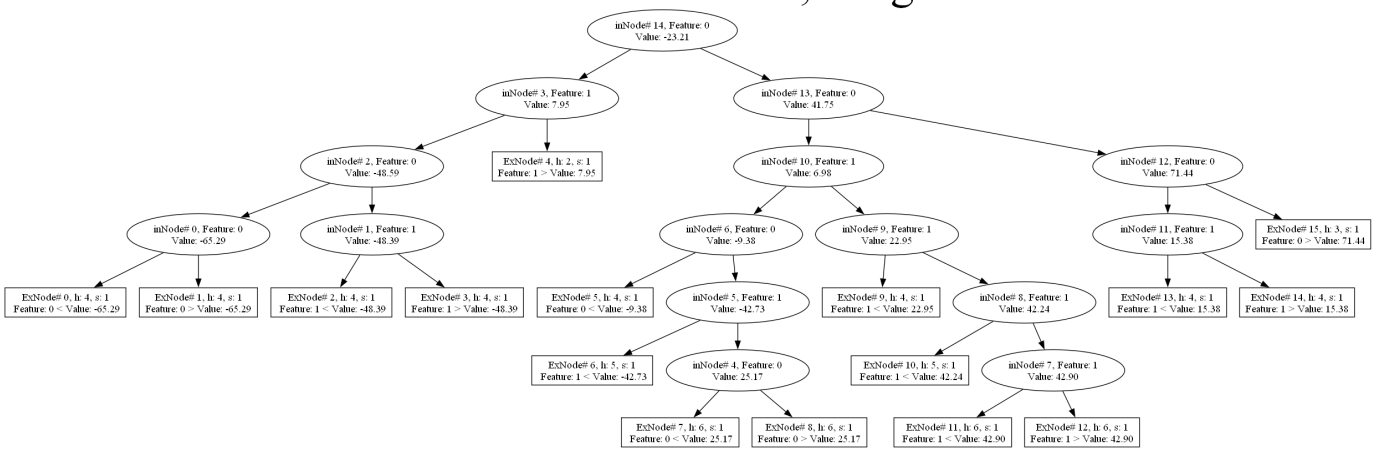

Figure 18. Visualization of an iTree with t-SNE