**A hybrid of Isolation Forest algorithm and C4.5 Decision Tree algorithm for Supervised Anomaly Detection**

Dharavath Ramesh\*, Aarush Juneja and Pratyush Kesarwani

Department of Computer Science and Engineering

Indian Institute of Technology (Indian School of Mines), Dhanbad - 826004, India

**Email**: drramesh@iitism.ac.in\*, aarush.15je001424@cse.iitism.ac.in, pratyush.15je1100@cse.ism.ac.in

**Abstract:** Anomalies can be defined as patterns or data points that do not conform to a well-defined notion of normal behaviour.Anomaly detection is a very popular research problem which caters the interest of a large amount of research scientists. Anomaly detection is a very important step in every good Data Mining framework. Several techniques involving one or more of the following fields, namely, Statistical Analysis, Machine Learning, Soft Computing, Deep Learning, Information Theory etc. have been used for making better anomaly detection systems. Anomaly detection finds its applications in various fields such as detecting malicious behaviour in online social media networks, detecting fraud in credit card transactions, fault detection systems etc.

SHIFT THIS TO INTRODUCTION KA LAST. (Section 2 contains this, Section 3 contains kar kar ke)

/\*\*In this paper, we

describe a hybrid algorithm that uses Isolation Forests and C4.5 decision trees to efficiently detect anomalies. We then demonstrate the algorithm by detection fraudulent transactions amongst a huge dataset of transactions. We then compare the hybrid algorithm with the standard

Isolation Forest algorithm and the standard C4.5 Decision Tree Algorithm separately. We describe how the performance of Isolation Forest is improved on combining the concept of Isolation Trees and C4.5 Decision Trees. \*\*/

**Keywords:** Anomaly detection, ensemble learning, supervised machine learning, isolation forest, decision tree.

// Make Introduction 1.5 page long atleast

**1. Introduction:** Outliers are extreme values that deviate from other observations on data , they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample. There are two kinds of outliers, Univariate and Multivariate. Univariate outliers refers to looking for outliers in a single feature distribution and Multivariate outliers refers to looking for outliers in a n-feature distribution. Common cause of outliers include data entry errors, measurement errors, data processing errors, intentional errors etc. Fraud detection is a topic applicable to many industries including banking and financial sectors, insurance, government agencies . Fraud are on a rise in recent years, making fraud detection more important than ever. Despite efforts on the part of the affected institutions, hundreds of millions of dollars are lost to fraud every year. Since relatively few cases show fraud in a large population, detecting fraud is tricky. Isolation forest is an effective method for fraud detection. Isolation forest’s basic principle is that outliers are few and are far from the rest of the observations. Isolation Forest explicitly prunes the underlying isolation tree once the anomalies are identified. We have observed that C4.5 Decision Trees perform better than Isolation Trees while using the underlying expected value mathematics of the Isolation Forests in the ensemble built using C4.5 Decision Trees.

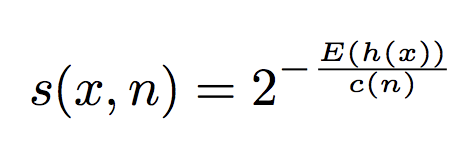
2. Related Work

1. Isolation Forest ka paper
2. Isolation Forest se anomaly detection ka 2012 ka paper
3. J48 trees se anomaly detection ka paper (probably 2015)

**3. Isolation Forest Algorithm:** // explain the standard algorithm and write pseudocode here

Isolation Forest is a supervised learning algorithm for ensemble based anomaly detection.

The underlying estimator of an Isolation Forest ensemble is Isolation Tree. It is a binary tree such that each node has either 2 children or no child. In these trees, partitions at each node are created by first randomly selecting a feature and then selecting a random split value between the minimum and maximum value of the selected feature. In principle, outliers are less frequent than regular observations and are different from them in terms of values (they lie further away from the regular observations in the feature space). That is why by using such random partitioning they should be identified closer to the root of the tree (shorter average path length, i.e., the number of edges an observation must pass in the tree going from the root to the terminal node), with fewer splits necessary. As with other outlier detection methods, an anomaly score is required for decision making. In case of Isolation Forest it is defined as:

 (1)

where *h(x)* is the path length of observation *x*, *c(n)* is the average path length of unsuccessful search in a Binary Search Tree and *n* is the number of external nodes. E(h(x)) is the average of h(x) in the entire ensemble of Isolation Trees. Each observation is given an anomaly score and the following decision can be made on its basis:

* Score close to 1 indicates anomalies
* Score much smaller than 0.5 indicates normal observations
* If all scores are close to 0.5 than the entire sample does not seem to have clearly distinct anomalies

Given a data set of n instances, Section 10.3.3 of [1] gives the aver-

age path length of unsuccessful search in BST as:

**c(n) = 2H(n − 1) − (2(n − 1)/n) (2)**

where **H(i)** is the harmonic number and it can be estimated

by **ln(i) + 0.5772156649 (Euler’s constant).**

The following represents the algorithm used for building an Isolation Forest ensemble.

**(Detailed explanation is to be added here)**

**Algorithm 1 : iF orest(X, t, ψ)**  
**Inputs:** X - input data, t - number of trees, ψ - sub-sampling size  
**Output:** a set of t iTrees  
1: Initialize Forest  
2: set height limit l = ceiling(log 2 ψ)  
3: for i = 1 to t do  
4:X 0 ← sample(X, ψ)  
5:Forest ← Forest ∪ iTree(X 0 , 0, l)  
6: end for  
7: return F orest

The following represents how an Isolation Tree is formed. **(Detailed explanation is to be added here)**

**Algorithm 2 : iT ree(X, e, l)**

**Inputs:** X - input data, e - current tree height, l - height limit

**Output:** an iTree

1: if e ≥ l or |X| ≤ 1 then

2:return exNode{Size ← |X|}

3: else

4:let Q be a list of attributes in X

5:randomly select an attribute q ∈ Q

6:randomly select a split point p from max and min values of attribute q in X

7:X\_left ← f ilter(X, q < p)

8:X\_right ← f ilter(X, q ≥ p)

9: return inNode{Left ← iT ree(X\_left , e + 1, l),

10: Right ← iT ree(X\_right, e + 1, l),

11: SplitAtt ← q,

12: SplitV alue ← p}

13: end if

The following algorithm defines the procedure to be used for computing the anomaly score of a test instance, based on a single isolation Tree. The average of path lengths given by using the following procedure for each Isolation Tree in the Isolation Forest is E(h(x)). This value when put in equation (1) gives the anomaly score for this instance of data.

**Algorithm 3 : PathLength(x, T, e)**

Inputs : x - an instance, T - an iTree, e - current path length;

to be initialized to zero when first called

Output: path length of x

1: if T is an external node then

2: return e + c(T.size) {c(.) is defined in Equation 1}

3: end if

4: a ← T.splitAtt

5: if x a < T.splitV alue then

6: return P athLength(x, T.lef t, e + 1)

7: else {x a ≥ T.splitV alue}

8: return P athLength(x, T.right, e + 1)

9: end if

**4.** **C4.5 Decision Tree Algorithm: //** explain the entropy based decision tree that works for nominal data as well here, and write pseudocode here. C4.5 Decision Trees are also known as J48 Decision Trees in some literatures.

**5. Hybrid algorithm:**  In this approach, we have improved upon the standard Isolation Forest which is an ensemble of Isolation Trees, by replaced the base estimator of the ensemble with the State Of The Art Decision Trees, namely C4.5 Decision Trees.

6. Comparison: Isolation Forests scikit, C4.5 khud ka, CART scikit, Logistic Regression scikit, Random Forest scikit, Isolation Forest + C4.5 Khud ka.

7. Conclusion and Future work

**References:**

[1] B. R. Preiss. Data Structures and Algorithms with Object-Oriented Design Patterns in Java.

Wiley, 1999.