Evaluating the Sensitivity of Isolation Forest Parameters in Anomaly Detection

DT Nicolay 26296918
Computer Science Division
Stellenbosch University
Stellenbosch, South Africa
26296918@sun.ac.za

***Abstract*—TODO**

***Index Terms*—TODO**

I. Introduction

II. Background

A. Isolation Forests

The majority of existing model-based approaches to anomaly detection construct a profile of normal instances, then they identify instances that do not conform to this normal profile as anomalies [1]. Liu et al. (2008) proposed a fundamentally different model-based method that explicitly isolates anomalies instead of profiles normal points, Isolation Forests. Here, isolation refers to separating an instance from the rest of the instances. This is ideal for an anomaly detection problem context, since anomalies are by nature sparse and diverse.

Normal profile methods, since not optimised for anomaly detection, often lead to too many false positives or little to no anomalies detected at all. These methods are also constrained to low dimensional data and small data size since they require significant computational power. Isolation Forests on the other hand take advantage of anomaly datasets consisting of fewer observations for the target class, and anomalies having feature values distinct from the rest of the data. Due to the nature of anomaly observations, they are isolated closer to the root of the tree. This forms the foundation of Isolation Trees.

Isolation Forests involve an ensemble of isolation trees where the predicted anomalies are the observations with the shortest average paths across the trees. Using this model for anomaly detection involves two stages. The first stage constructs isolation trees by recursively partitioning. This training stage is described in Algorithm 1 and Algorithm 2 [2].

Algorithm 1 iForest(X, t, ψ)

```
Require: $X$ – input data, $t$ – number of trees, $\psi$ – subsampling size
Ensure: A set of $t$ isolation trees
1: Initialize Forest $\leftarrow \emptyset$
2: for $i = 1$ to $t$ do
3:     $X' \leftarrow$ sample$(X, \psi)$
4:     Forest $\leftarrow$ Forest $\cup$ iTree$(X')$
5: end for
6: return Forest
```

The iTree algorithm recursively partitions the dataset by randomly selecting a feature and a random split value until the data can no longer be divided. The resulting tree structure isolates individual points, where anomalies tend to have shorter average path lengths because they are easier to isolate.

Algorithm 2 iTree(X')

```
Require: $X'$ – input data
Ensure: an isolation tree
1: if $X'$ cannot be divided then return exNode{Size $\leftarrow |X'|$}
2: else
3:     Let $Q$ be the list of attributes in $X'$
4:     Randomly select an attribute $q \in Q$
5:     Randomly select a split point $p$ between the max and min values of attribute $q$ in $X'$
6:     $X_l \leftarrow$ filter$(X', q < p)$
7:     $X_r \leftarrow$ filter$(X', q \geq p)$ return inNode{
8:      Left $\leftarrow$ iTree$(X_l)$,
9:      Right $\leftarrow$ iTree$(X_r)$,
10:     SplitAtt $\leftarrow q$,
11:     SplitValue $\leftarrow p$}
12: end if
```

B. Control Parameters

There are five control parameters to consider namely: the number of estimators, the maximum samples, the contamination, the maximum features, and whether to first bootstrap sample.

The number of trees parameter determines the number of base estimators in the ensemble. This parameter is denoted by t in Algorithm 1. The performance of Isolation Forests converges quickly with a very small number of trees [1].

The maximum samples describes the number of sample observations to draw from the training data for each base estimator. The subsampling size is denoted by ψ in Algorithm 1. Only a small sampling size is required to achieve high detection performance with high efficiency [1]. Setting ψ to 256 often suffices for anomaly detection across a wide range of data [2].

Contamination refers to the proportion of anomalies present in the dataset. It is defined as the number of anomalies divided by total number of observations. When set to a specific value between 0 and 0.5, it determines the threshold on the anomaly scores such that approximately that fraction of the training samples are labelled as outliers. In the original paper, the threshold is automatically fixed at an offset of -0.5, following the original Isolation Forest formulation, where inliers typically yield scores near 0 and outliers near -1, allowing the model to separate them without prior knowledge of the true contamination level. Since this parameter directly influences the threshold for anomaly detection, mis-tuning will directly increase false positives or false negative rates.

The maximum features describes how many features are selected at random before tree construction to train each base estimator. Introducing randomness can help with with high-dimensional data, but may increase instability if set too low.

The bootstrap parameter controls the manner in which sample observations are drawn for each tree. This determines whether sampling is done with or without replacement. That is, sub-sample X' is randomly sampled with or without replacement in Algorithm 1 from X. Bootstrap resampling can lead to marginal improvements across classification metrics [3].

C. Evaluation Stage

For this stage, a single path length $h(\boldsymbol{x})$ is computed by counting the number of edges from the root node to an external node as an instance $\boldsymbol{x}$ traverse each iTree. There is a predefined height limit, *hlim*, that when reached, the algorithm returns the value e plus an adjustment $c(Size)$. This process is described in Algorithm 3 [2]. The worst case time complexity of the evaluation stage for a dataset size n is $O(nt\psi)$.

Algorithm 3 PathLength($\boldsymbol{x}, T, h_{lim}, e$)

```
Require: $\boldsymbol{x}$ – an instance; $T$ – an iTree; $h_{lim}$ – height limit;
    $e$ – current path length (initialized to 0 when first called)
Ensure: Path length of $\boldsymbol{x}$
 1: if $T$ is an external node or $e \geq h_{lim}$ then
 2:     return $e + c(T.\text{size})$ ▷ $c(\cdot)$ is defined in Equation (1)
 3: else
 4:     $a \leftarrow T.\text{splitAtt}$
 5:     if $x_a < T.\text{splitValue}$ then
 6:         return PathLength($\boldsymbol{x}, T.\text{left}, h_{lim}, e+1$)
 7:     else
 8:         return PathLength($\boldsymbol{x}, T.\text{right}, h_{lim}, e+1$)
 9:     end if
10: end if
```

The anomaly score $s(\boldsymbol{x}, \psi)$ for instance $\boldsymbol{x}$ is computed using

$$s(\boldsymbol{x}\psi) = 2^{-\frac{E(h(\boldsymbol{x}))}{c(\psi)}}, \qquad (1)$$

where $c(\psi)$ is the average path length of unsuccessful searches and $E(h(\boldsymbol{x}))$ is the average depth from the forest of isolation trees. If the score of an observation s is close to 1, the observation is an anomaly. If an observation has a score much smaller than 0.5, then it is regarded as a normal instance. However, if all observations return a score of approximately 0.5, then there are no distinct anomalies.

III. Implementation

A. Tools and Libraries

This implementation was created using Python. NumPy [4] is used for numerical computations. Pandas [5] is used for data manipulation and analysis. Matplotlib [6] is used for data visualisation and plotting. Scikit-learn [7] was used extensively fo modelling, evaluation, and the evaluation metrics.

B. Isolation Forest Algorithm

The Scikit-learn library [7] implementation of the Isolation Forest algorithm described in the background section is used to detect anomalies across three varied datasets. This implementation returns the anomaly score for each observation. The path length to an observation in the a tree, averaged over a forest of isolation trees, is used as a measure of normality and the decision function. When the Isolation Forest collectively produces shorter path length for a particular observation, it is likely to be an anomaly. The maximum depth of each tree is $\lceil log_2(n) \rceil$ where n is the number of samples used to build the tree.

A custom framework is built around the base Isolation Tree implementation to track evaluation metric means and their standard deviations over a specified number of runs. The control parameter comparisons are split into two sections. First, the analysis is conducted on each dataset individually and all results are saved. Plots and statistical tests are performed for the individual datasets to compare the control parameters. Second, the results from the individual datasets tests are compared across datasets separately to analyse the data-dependent effects.

C. Experimental Setup

A random seed of 12 is used for all stochastic procedures to ensure reproducibility. All tests are conducted on a Fedora Linux 41 system with a 13th Gen Intel Core i3-13100 CPU with 16 GiB of memory. The computationally significant portions of the algorithm are running the algorithm with a large ensemble size and also a large parameter grid for some parameter combinations. The code for this paper is available on GitHub [8].

IV. Empirical Process

The datasets and their rationale are described in this section. This is followed by an explanation of the evaluation metrics used along with the experimental design. Finally, the overall analysis framework is outlined.

A. Datasets

The datasets used for control parameter evaluation are described in Table I and are all sourced from ADBench [9]. These datasets provide a good variety in the number of observations, features, anomaly rate, and the difficulty of the anomaly detection problem. This enables a thorough and balanced representation of the performance of the control parameters in different problem settings.

TABLE I: Summary of Datasets Used

Dataset	**Observations**	**Features**	**Anomaly Rate (%)**	**Domain**
Shuttle	49,097	9	7.15	Astronautics
Campaign	41,188	62	11.27	Finance
Fraud	284,807	29	0.17	Finance

B. Evaluation Metrics

Given the nature of anomaly detection as an imbalanced classification problem, metrics must account for the typically small proportion of anomalies in real-world datasets, as evidenced by the 0.17% anomaly rate in the *fraud* dataset compared to 11.27% in *campaign*. The F1-score served as the primary performance indicator, providing a balanced measure between precision and recall. Precision quantified the proportion of correctly identified anomalies among all flagged instances, which is critical in applications where false alarms incur significant investigation costs. Recall measured the proportion of true anomalies successfully detected, representing the algorithm's sensitivity to anomalous patterns. Both metrics are essential as they capture the trade-off inherent in anomaly detection: aggressive detection (high recall) risks overwhelming analysts with false positives, while conservative detection (high precision) may miss critical anomalies.

ROC-AUC (Receiver Operating Characteristic Area Under Curve) and PR-AUC (Precision-Recall Area Under Curve) evaluated the model's ability to rank anomalies correctly across different decision thresholds. ROC-AUC assesses overall discriminative power, while PR-AUC is particularly informative for imbalanced datasets as it focuses on performance in the positive (anomaly) class. These metrics are computed using the anomaly scores from Isolation Forest's `score_samples()` method, which provides a continuous ranking rather than binary classifications.

Since Isolation Forests are stochastic in nature with regards to feature selection during tree construction, performance does vary between runs.The Jaccard similarity coefficient was used to measure the performance consistency between runs. Training time and memory usage were also tracked to asses the scalability and computational resource management. The average path length statistic was also monitored.

C. Experimental Design

1) Single-Parameter Analysis: Each control parameter was individually varied whilst holding all the others at their base parameters. These base parameters are 100 estimators, 256 maximum samples, the actual contamination percentage of the dataset, all features included and bootstrap sampling disabled. For each configuration, 10 runs were performed with the random seed incremented for each run. The mean, minimum, maximum and standard deviation of all the performance metrics were recorded.

For the number of estimators in the ensemble, the range considered was $n_{\text{estimators}} \in \{1, 2, ..., 25, 50, 75, 100, 125, 175, 200\}$. Originally, a much broader range with larger intervals up to 2000 was considered. However, convergence analysis revealed that the ensemble often converged before 100/200 estimators, therefore, a narrower set of values was considered to more closely monitor trends. The range for the maximum samples parameter was $n_{\text{max samples}} \in \{4, 8, 12, 16, 20, 24, 28, 32, 64, 128, 256\}$. The smaller interval points captured more specific information about where the model converged, and the larger gaps considered the larger parameter values to obtain a broader view of the performance. The contamination parameter was considered over the values contamination $\in \{0.01, 0.02, 0.03, 0.05, 0.07, 0.10, 0.15, 0.20, 0.25, 0.30\}$. The contamination parameter is obviously problem dependent, and, in a true unsupervised learning scenario, unknown. The maximum features parameter was varied linearly from 1 up to the total number of available features (capped at 20) using integer increments. Bootstrap sub-sampling was also considered.

Three or four plots were created for each parameter experiment. Most notably, the convergence plot including F1-score, precision, and recall against the parameter value was created. This plot included the standard deviation in addition to the mean to illustrate the variability over the 10 runs. The training time scaling was plotted by considering the training time against the parameter plotted on a log-log scale. For parameters affecting the precision-recall balance (such as contamination and maximum samples), precision-recall curves were constructed to visualize the trade-offs inherent in different parameter configurations.

The coefficient of variation was computed for the F1-score across the 10 runs to quantify stability, with lower CV values indicating more consistent performance. For the contamination parameter specifically, calibration curves were generated to assess whether the specified contamination level reliably controlled the proportion of instances flagged as anomalies, providing insight into the parameter's predictive calibration.

2) Multi-Parameter Interaction Analysis: Interactions between the maximum features and number of estimator parameters were considered by running all the combinations of the previously described maximum feature range and the number of estimators in intervals of three from one to 25. For example, for the *shuttle* dataset, $9 \times 9 = 81$ combinations were considered. The training time and F1-scores were plotted in a heat map to investigate any correlations.

3) Cross-Dataset Comparison: The convergence curves along with the training times further were analysed across the various datasets in the same plots. This enabled the identification of data specific trends. The variation across runs was also plotted across the various datasets in order to compare the stability of the application to the different datasets.

D. Analysis Framework

V. Results & Discussion

The observed effects of the control parameters are described in this section. An analysis of the plots and experiment results is presented, followed by practical guidelines for the selection of the parameters for various applications.

A. Number of Estimators

1) Performance Convergence: Generally, the performance reported by the F1-score indicated that the model converged with 50 estimators in the ensemble. This is illustrated in Figure 1. The difference in F1-scores after convergence showed the varying complexity of the anomaly detection problems in each of the datasets. Notably, the recall was substantially higher than the precision in the shuttle dataset seen in Figure 6. However, the other two datasets maintained a similar precision and recall value even as the number of estimators were increased. This is observed in Figure 7 and Figure 8. The ensemble misclassifies some normal points as anomalies, but does easily isolate anomalies, leading to higher recall and lower precision. For the fraud dataset, the F1-score only fully plateaus after 125 estimators were used in the ensemble.

It can therefore be concluded that if 125 estimators are used, the model will likely converge. For computationally constrained circumstances, as few as 25 estimators will suffice for most use cases.

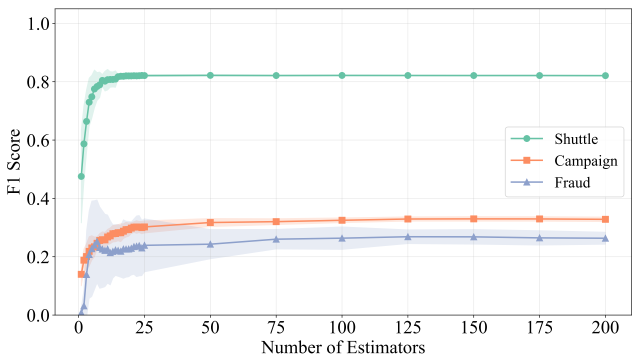

Fig. 1: F1-score performance across datasets for varying numbers of estimators.

2) Stability and Variance Reduction: The coefficient of variation calculated by dividing the mean F1-score by the F1-score standard deviation only fully stabilised at 125 estimators for the campaign and fraud datasets, but before 25 for the shuttle dataset. This stability is shown in in Figure 2 for the campaign dataset and Figure 9 and Figure 10 for the other two.

This behaviour aligns with the Law of Large Numbers: as the number of estimators increases, the random variation in individual model predictions averages out, causing the ensemble's F1-score to converge towards its expected value. Consequently, the variance of the mean decreases approximately as $1/n$, leading to the observed stability beyond 125 estimators. A lower coefficient of variation means the model is more reliable and therefore a higher number of estimators should be used in a case where consistent performance is critical such as fraud detection systems or campaign targeting pipelines where unstable predictions could lead to significant financial or reputational loss.

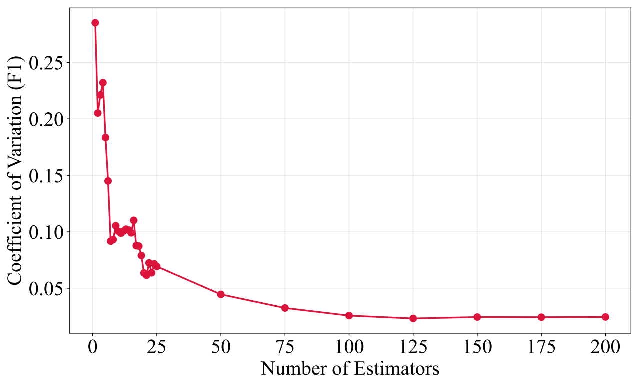

Fig. 2: Stability analysis of campaign dataset.

3) Computational Efficiency: The training time of the model scaled linearly as the number of estimators was increased. This relationship, observed in Figure 3, indicates that the per-estimator training cost is approximately constant across the range of the number of estimators. Also, there's no significant overhead or parallelisation bottleneck as the ensemble grows. As expected, the fraud dataset maintained the highest training time throughout the experiment since it contains the most observations.

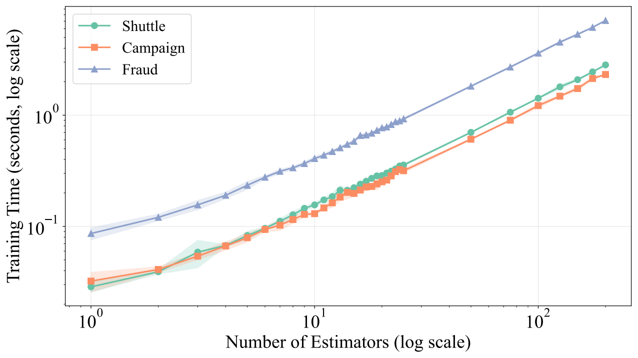

Fig. 3: Training time across increasing number of estimators.

B. Maximum Samples

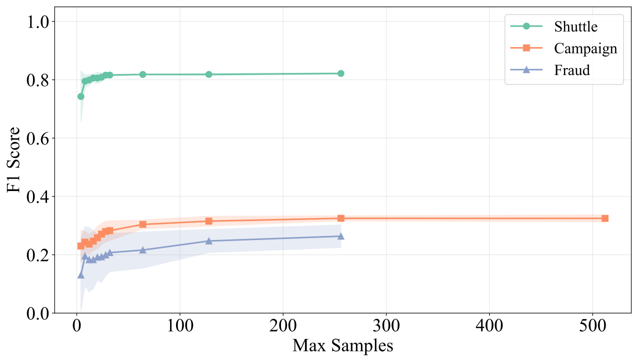

Fig. 4: Performance with increasing values of maximum samples parameter.

1) Performance vs. Subsample size:

2) Swamping Effect Analysis:

3) Training Time Complexity:

4) Emperical Rule Derivation:

C. Contamination Parameter

1) Robustness to Misspecification: Since this parameter is directly used by the algorithm to determine the threshold at which to classify an observation as an anomaly, the F1-score peaks at the contamination value that matches the correct percentage of anomalies in the dataset. For example, in Figure 5 the F1-score is greatest for the shuttle dataset at approximately 0.0715 which matches the anomaly rate of the dataset of 7.15%.

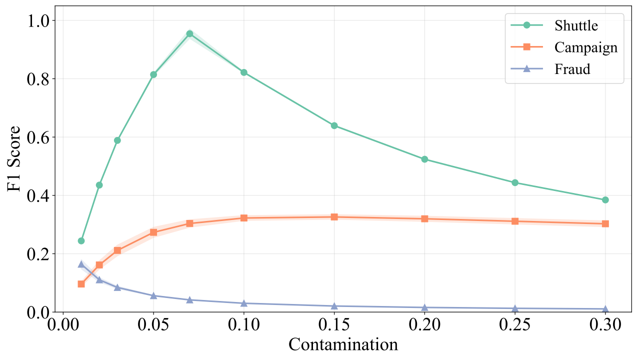

Fig. 5: Performance with increasing values of the contamination parameter.

2) Precision-Recall Trade-off:

3) Calibration Analysis:

D. Maximum Features

1) Performance:

2) Efficiency-Performance Trade-off:

3) Interaction with Ensemble Size:

E. Bootstrap Parameter

1) Stability Comparison:

2) Performance Metrics:

3) Interaction with Contamination:

F. Practical Guidelines

VI. Conclusions

References

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

[2] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, Mar. 2012. [Online]. Available: https://doi.org/10.1145/2133360.2133363

[3] H. Choi and K. Jung, "Impact of data distribution and bootstrap setting on anomaly detection using isolation forest in process quality control," *Entropy*, vol. 27, no. 7, p. 761, July 2025.

[4] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[5] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[6] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] D. Nicolay, "Ml441 assignments," https://github.com/Voltzz9/ml441_assignments, 2025.

[9] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," in *Neural Information Processing Systems (NeurIPS)*.

Appendix A
Additional Results

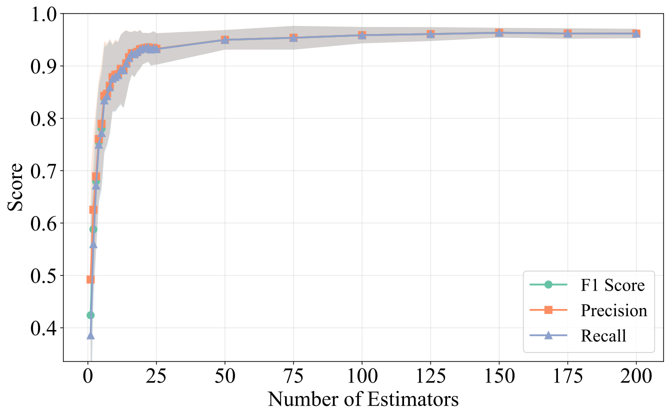

Fig. 6: Convergence curve for shuttle dataset.

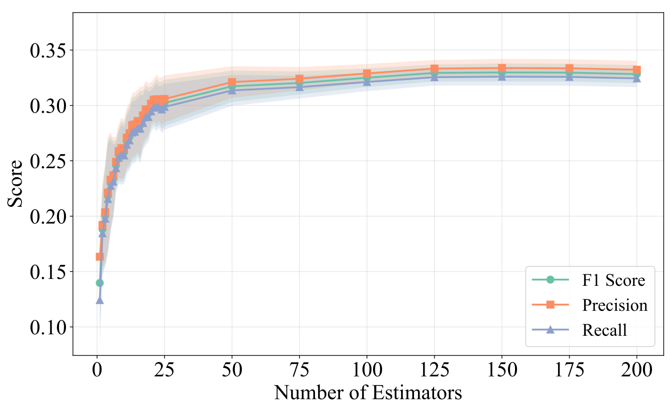

Fig. 7: Convergence curve for campaign dataset.

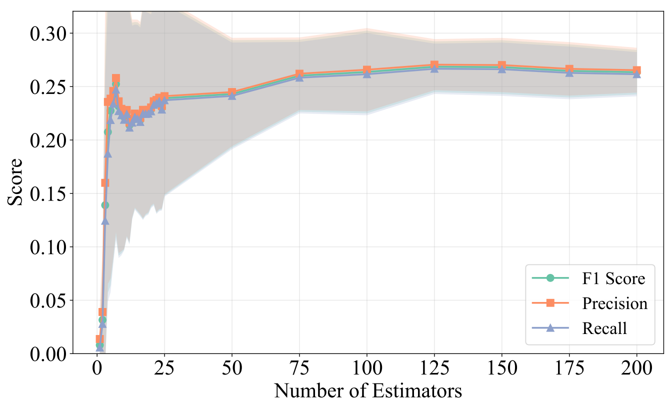

Fig. 8: Convergence curve for fraud dataset.

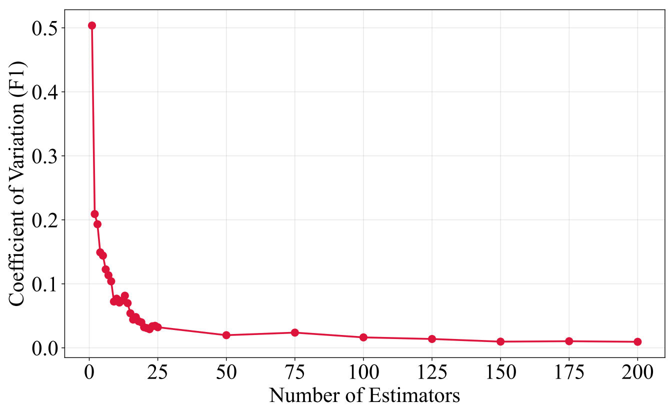

Fig. 9: Stability analysis of shuttle dataset.

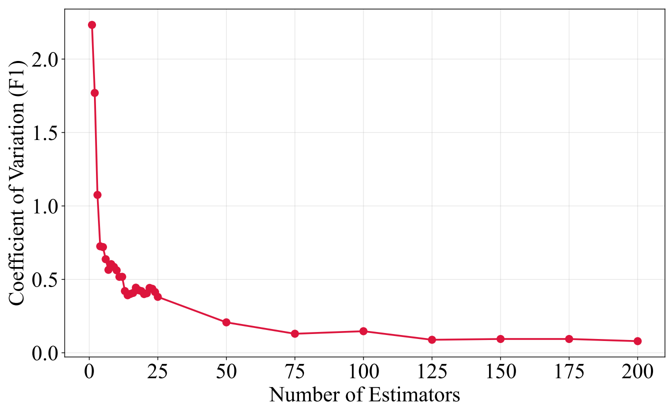

Fig. 10: Stability analysis of fraud dataset.