A Further Exposition on Usage of AGTRs

We will use this first section of our appendix to provide further exposition as to how we envision AGTRs to be used, as well as their constraints. To be clear, our proposed AGTR does not alleviate the need for labeled data, or avoid all possible biases. Indeed by constructing a mechanism that produces AGTRs, assuming it does not produce a complete graph, there must necessarily be some form of bias that exists via the selection of the AGTR production itself. Not only is labeled data needed to produce the initial model being evaluated, but labeled data is necessary to produce the estimate of the error rate $\hat{\epsilon}$ of the AGTR construction. These are important nuances to keep in mind through usage.

Our belief is that there are many domains where constructing an AGTR, with confidently small $\hat{\epsilon}$, is possible. Our example application has been for malware analysis as it is the area of our expertise. For example we see potential application areas in chemistry, physics, optics, and others where obtained experimental ground truth results is highly expensive and time consuming, but sophisticated simulations may be able to group different scenarios together with high fidelity. That is that the simulation's result may not produce the exactly correct prediction, but could be used to group different scenarios being evaluated into groups highly probably to have outcomes of the same fundamental nature. This is extrapolation on our part since we do not have the requisite background to delve deeply into these other areas, but captures our primary hypothesis: many domains have significant domain knowledge that could produce viable AGTRs.

Once an AGTR is produced, it provides a valuable means of improving bench-marking, confidence in results, and testing of nuanced changes. Designing good benchmarks if of critical importance, and an AGTR can be an effective litmus test for any benchmark creation. It allows one to cheaply use a much larger set of unlabeled data to determine if the benchmark itself has become the subject of over-fitting. We provided an example of this with the AVClass tool in our paper, where our bounds indicate AVClass's day may be over-fit because their empirical recall is higher than our upper bounded recall. This does not necessarily mean the results are actually over-fit, because the datasets evaluated are not the same (in AVClass' cases, not available to us). But it does provide a strong indicator that something is amiss, and provides the created of a benchmark the signal needed to more thoroughly and rigorously investigate their data.

This utility of AGTRs in benchmarking is simultaneously macro and micro in nature. THe prior paragraph has described the macro nature, that we can perform litmus testing against a created benchmark to detect overfitting against a larger unlabeled corpus. The next level down that one would natural want to pursue is to compare two different models using an AGTR, but we must caution the reader into doing such a comparison. When two models are being evaluated and of different mechanical natures, they will naturally correlate to different degrees with the biases of the AGTR construction itself. This means one method could appear to have better predicted bounds by our AGTR by only being successful on the data for which the AGTR itself is able to operate, while potentially being errant on all other data and would not be realized. Any other more reasonable model, that simply lacks the bias to the AGTR in use, would then appear worse when it may in fact be superior.

This is the reason why we recommend to only use AGTR to compare models that are fundamentally very similar, like changing small parameters of a single approach. This is reasonable because the underlying mechanism of prediction remains the same, and allows us to better quantify the impacts of minute changes that would be difficult to estimate on small static corpora. This assumes that the parameters themselves do not have an out-sized impact on the fundamental nature, which we believe is reasonable but should be made explicit.

This highlights another limitation of our approach, which is the number of epistemic uncertainties involved in its use. It will depend on the beliefs and confidence of the domain experts that the models being compared are "similar enough" that using an AGTR to compare changes is valid, and that the error rate bound $\hat{\epsilon}$ is sufficient. While more concrete answers to these are desirable, we recognize them as important items of future work. As we have developed the AGTR approach thus far, it provides a powerful means of better leveraging unlabelled data in the evaluation of a ML model, which we aruge is aditive to the tools in use today.

B Applications of Approximate Ground Truth Refinements

As long as an AGTR can be constructed from a dataset, that dataset can be used for partial evaluation of a clustering algorithm or multiclass classifier - even if the dataset does not have reference labels. This is notable because it allows larger, more representative datasets without reference labels to be used during the evaluation process. The ability to compute bounds on precision, recall, and accuracy without reference labels is valuable in its own right, but we have found two additional ways in which AGTRs can be used to great effect. First, the bounds from an AGTR can be used as a litmus test for detecting biased evaluation results produced using a substandard reference dataset. Second, AGTRs can be used for evaluating modifications to a clustering algorithm or multiclass clasifier.

B.1 Testing Suspect Evaluation Results

We have found that AGTRs can be used to detect misleading results produced from low quality reference datasets. The following five steps describe how to test suspect evaluation results:

1. Compute the precision, recall, and/or accuracy of a clustering algorithm or a multiclass classifier using a substandard reference dataset.
2. Obtain C by applying the clustering algorithm or multiclass classifier to a large, diverse dataset.
3. Construct an AGTR $\hat{R}$ from the dataset in step 2.
4. Select a value of $\hat{\epsilon}$ that is believed to be greater than the number of errors in $\hat{R}$.
5. Compute *Precision(C,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m}$ and *Recall(C,* $\hat{R}$*)* $+ \frac{\hat{\epsilon}}{m}$. Test whether all bounds hold for the evaluation results found during step 1.

Note that if testing a multiclass classifier, C is obtained during step 2 by using the classifier to predict the class label for each data point in the dataset and then clustering all data points that share the same predicted class label. Subsection 2.4 describes the qualities an ideal AGTR should possess for best evaluation results. Refer to subsection 2.5 for discussion about how to select an appropriate value of $\hat{\epsilon}$ during step 4.

Because the dataset used to construct the AGTR during step 3 is larger and more diverse than the reference dataset from step 1, we assume that the AGTR dataset is a better exemplar of the overall problem space. Although the metric bounds found using the AGTR dataset do not necessarily hold for the reference dataset, it would still be irregular for any evaluation results from the reference dataset to violate them. Therefore, if *Precision(C,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m}$ is greater than the precision computed using the reference dataset, or if *Recall(C,* $\hat{R}$*)* $+ \frac{\hat{\epsilon}}{m}$ is less than the recall or accuracy computed using the reference dataset, it is reasonable to conclude that the evaluation results found using the reference dataset are misleading.

B.2 Comparing Similar Models

An important component of cluster and classifier evaluation is the ability to compare models against each other. Suppose we wish to determine which of two clustering algorithms has a higher precision, but we do not have access to a satisfactory reference dataset. We use the two models to predict clusterings C_1 and C_2 from an unlabeled dataset and we construct an AGTR $\hat{R}$ from the same dataset. The precision lower bounds of the two clustering algorithms are given by *Precision(*C_1*,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m}$ and *Precision(*C_2*,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m}$. Although we can apply Theorem 5 to show *Precision(*C_1*,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m} \leq$ *Precision(*C_1*, D)* and *Precision(*C_2*,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m} \leq$ *Precision(*C_2*, D)*, we cannot prove any relationship between *Precision(*C_1*, D)* and *Precision(*C_2*, D)*.

Unfortunately, evaluation metric bounds cannot be used to provably determine whether one clustering algorithm or multiclass classifier has a higher precision, recall, or accuracy than another. However, in specific cases higher evaluation metric bounds may indicate that one model has a higher performance than another. Two conditions must be met in order for this approach to be used. First, the clustering algorithms or classifiers being compared must be intrinsically similar, such as two different versions of the same classifier. Second, one of the clustering algorithms or classifiers must be tested to ensure

that changes in performance are strongly correlated to changes in evaluation metric bounds. The steps of the test are as follows:

1. Obtain C by applying the clustering algorithm or multiclass classifier to a large, diverse dataset.
2. Construct an AGTR $\hat{R}$ from the dataset in step 1.
3. Incrementally shuffle the cluster membership of each data point in C.
4. Compute *Precision(C,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m}$ and *Recall(C,* $\hat{R}$*)* $+ \frac{\hat{\epsilon}}{m}$ at regular intervals of the shuffle.
5. Compute correlation between shuffle percentage and the evaluation metric bounds. Test that a strong negative correlation between the two exists.

Step 3 is performed randomly sampling a data point in C with no replacement, randomly selecting a cluster in C weighted by the original distribution of cluster sizes, and then assigning the data point to that cluster. The process repeats m times, where m is the number of points in the dataset, after which the entire clustering has been shuffled. Because the dataset has high diversity, the probability that a datapoint is randomly assigned to an incorrect cluster is far greater the probability that it is randomly assigned the correct one. Therefore, shuffling C with this strategy is very likely to produce predicted clusterings that are sequentially worse. Step 5 is performed by computing the Pearson correlation between the shuffle percentage and the evaluation metric bounds. If a very strong negative correlation exists between the two, we conclude that small modifications to a clustering algorithm or classifier that improve its performance will be reflected by a higher evaluation metric bounds, and modifications that lower its performance will be reflected by lower bounds.

C Malware Label and Dataset Challenges

The task of labeling malware data is expensive and error prone, more so than most standard ML applications areas. This forces datasets to be either small and thus non-representative of the large and diverse malware ecosystem, or large but noisily labeled or biased which again limits conclusions. Because of these issues, we believe that malware family classification is an ideal problem space for demonstrating the AGTR evaluation framework. We will quickly review the challenges in labeling malware and how it has caused a lack of quality datasets.

Ideally a malware reference dataset would be constructed using *manual labeling* to determine the malware family of each file. Manual analysis is not perfectly accurate, but the error rate is considered negligible enough that labels obtained via manual analysis are considered to have ground truth confidence [13]. A professional analyst can take a 10 hours or more to fully analyze a single file [14, 15]. This level of analysis is not always needed to determine the family of a malware sample, but it exemplifies the high human cost of manual labeling that prevents its use in all but very small datasets [1]. One approach to mitigate this is *cluster labeling*, where the dataset is clustered and an exemplar from each cluster is manually labeled. This strategy is highly reliant on the precision of the algorithm used to cluster the reference dataset, which is often custom-made [16, 17]. Furthermore, the scalability of cluster labeling is still limited due to the requirement of manual analysis.

For these reasons larger malware reference datasets tend to use *antivirus labeling*, where an antivirus engine is used to label a corpus. This is easy to implement at scale, but has significant quality issues [18]. Antivirus labels are frequently incomplete, inconsistent or incorrect [19, 20]. Antivirus signatures do not always contain family information [13], and different antivirus engines disagree on the names of malware families [9]. Labels from an antivirus engine can take almost a year to stabilize [18], creating a necessary lag time between data occurrence and label inference. These issues can be partially mitigated by using *antivirus majority voting*, but aggregating the results of multiple antivirus engines produces unlabeled files due to lack of consensus. This then biases the final dataset to only the "easy" samples that are the least interesting and already known by current tools [2].

The aforementioned labeling issues have a significant impact on the datasets that are available. The largest datasets used in malware classifier and clustering research range from a hundred thousand [13, 16] up to one million samples[21], but are private corpora held by corporations that can afford the construction cost and do not want to give away a competitive advantage[1]. Since the data is private

[1]There are also legal concerns for sharing benign applications, but our discussion is focused solely on malware.

the validation of the labeling can not be replicated or investigated, and in most cases the number of families is not fully specified[13, 16]. The vast majority of publicly available datasets that have been used are less than 12,000 samples in size[17, 22–26]. Of these MalGenome is the only fully manually labeled corpus, but also the smallest with only 49 families and 1,260 files[27, 28]. The small size and low diversity of these corpora makes it difficult to make generalizable conclusions about the quality of a malware clustering algorithm or classifier. The largest reference dataset in general use is VX Heavens with 271,092 samples, but it contains very old malware, is labeled using only a single antivirus engine, and the history and composition of the dataset is poorly documented [29]. Additional statistics about notable malware reference datasets are listed in Appendix E.

Given the issues in curating these datasets, we have identified three common three common traits that negatively impact evaluation. All of the datasets we just referenced possess one or more of these undesirable traits.

1) Reference labels without ground truth confidence: Because producing ground truth reference labels for a large corpus of malware samples is infeasible, it is common practice to use malware reference labels without ground truth confidence [1]. Many prior malware classifiers have been evaluated using non-ground truth reference labels without the quality of those labels having been assessed [9], which may result in overoptimistic or misleading results [2].

2) Insufficient size or diversity: Due to the enormous number of families in existence, malware reference datasets must be both large and diverse in order to be representative of the malware ecosystem. Using a reference dataset with a small number, or imbalanced set, of families lowers the significance of evaluation results [2].

3) Outdated malware samples: The ecosystem of malware is constantly changing as categories of malware, malware families, and other tradecraft rise to prominence or fall out of favor. Evaluating a clustering algorithm or multiclass classifier using a dataset of outdated malware may produce results that do not translate to present day malware. All of the discussed datasets contain malware samples from 2015 and earlier, failing to represent the last half decade of malware development.

D Proof Details

Theorem 1. *Precision(C, R) ≤ Precision(C, D)*

Proof of Theorem 1. Suppose some i s.t. $1 \leq i \leq c$. Since R is a refinement of D, $\exists D_j \in D$ s.t. $R_{f'(i)} \subseteq D_j$. Because $R_{f'(i)} \subseteq D_j$, it must be that $|C_i \cap R_{f'(i)}| \leq |C_i \cap D_j|$. By definition, $f(i) = \underset{j}{\operatorname{argmax}} |C_i \cap D_j|$. Therefore, $|C_i \cap R_{f'(i)}| \leq |C_i \cap D_j| \leq |C_i \cap D_{f(i)}|$. We can simply sum over this inequality to obtain:

$$\frac{1}{m} \sum_{i=1}^{c} |C_i \cap R_{f'(i)}| \leq \frac{1}{m} \sum_{i=1}^{c} |C_i \cap D_{f(i)}|$$

By Definition 1, $Precision(C, R) \leq Precision(C, D)$. □

Theorem 2. *Recall(C, R) ≥ Recall(C, D)*

Proof of Theorem 2. Suppose some j s.t. $1 \leq j \leq d$. Because R is a refinement of D, $\exists Q_j = \{Q_{j\ell}\}_{1 \leq \ell \leq q_j}$ s.t. $Q_{j\ell} \in R$ and $D_j = \bigcup_{\ell=1}^{q_j} Q_{j\ell}$. We can say that $|D_j| = \sum_{\ell=1}^{q_j} |Q_{j\ell}|$ and furthermore that $|C_{g(j)} \cap D_j| = \sum_{\ell=1}^{q_j} |C_{g(j)} \cap Q_{j\ell}|$. We know that $\forall Q_{j\ell} \in Q_j$, $\exists R_k \in R$ s.t. $Q_{j\ell} = Q_k$. By definition $g'(k) = \underset{i}{\operatorname{argmax}} |C_i \cap R_k|$, and $R_k = Q_{j\ell}$, so $|C_{g'(k)} \cap R_k| \geq |C_{g(j)} \cap Q_{j\ell}|$. Since by Property 1 the sets R_k are in bijection with the sets $Q_{j\ell}$, we can sum over this inequality to obtain:

$$\frac{1}{m} \sum_{k=1}^{r} |C_{g'(k)} \cap R_k| \geq \frac{1}{m} \sum_{j=1}^{d} \sum_{\ell=1}^{q_j} |C_{g(j)} \cap Q_{j\ell}|$$

$$\frac{1}{m}\sum_{j=1}^{d}\sum_{\ell=1}^{q_j}|C_{g(j)}\cap Q_{j\ell}| = \frac{1}{m}\sum_{j=1}^{d}|C_{g(j)}\cap D_j|$$

By Definition 2, *Recall(C, R)* $\geq$ *Recall(C, D)*. □

Corollary 2.1. *Recall(C, R)* $\geq$ *Accuracy(C, D)*

Proof of Corollary 2.1. Using Theorem 2, *Recall(C, R)* $\geq$ *Recall(C, D)*. It must be the case that *Recall(C, D)* where $g(j) = \underset{i}{\text{argmax}}\,|C_i \cap D_j|$ $\geq$ *Recall(C, D)* where g is the identity function. Using Definition 3, *Recall(C, D)* = *Accuracy(C, D)* when g is the identity function. Therefore, *Recall(C, R)* $\geq$ *Accuracy(C, D)*. □

Theorem 3. $|$*Precision(C, S)* $-$ *Precision(C,* $\hat{S}$*)*$| \leq \frac{1}{m}$

Proof of Theorem 3. Let $S = \{S_t\}_{1\leq t\leq s}$ be an arbitrary partition of M. Let the label translation function $f : \{1...c\} \mapsto \{1...s\}$ be defined as $f(i) = \underset{t}{\text{argmax}}\,|C_i \cap S_t|$. Suppose some $M_n \in M$, some $S_x \in S$ s.t. $M_n \in S_x$, and some S_y s.t. $S_y \in S$ or $S_y = \{\emptyset\}$. Furthermore, suppose some $C_a, C_b \in C$ s.t. $M_n \in C_a$ and $b = f(y)$. Let $\hat{S} = \{\hat{S}_{\hat{t}}\}_{1\leq\hat{t}\leq\hat{s}}$ be a clustering identical to S except for one cluster label change, which is given by $\hat{S}_x = S_x - \{M_n\}$ and $\hat{S}_y = S_y \cup \{M_n\}$. Let the function $\hat{f} : \{1...c\} \mapsto \{1...\hat{s}\}$ be defined as $\hat{f}(i) = \underset{\hat{u}}{\text{argmax}}\,|C_i \cap \hat{S}_{\hat{u}}|$. At minimum $|C_a \cap \hat{S}_{\hat{f}(a)}| = |C_a \cap S_{f(a)}| - 1$ and at maximum $\hat{S}_{\hat{f}(a)}| = |C_a \cap S_{f(a)}|$. Similarly, at minimum $|C_b \cap \hat{S}_{\hat{f}(b)}| = |C_b \cap S_{f(b)}|$ and at maximum $|C_b \cap \hat{S}_{\hat{f}(b)}| = |C_b \cap S_{f(b)}| + 1$. Finally, because all elements in M are unique and C partitions M, $\forall C_i \in C$ s.t. $i \neq a$ and $i \neq b$, $|C_i \cap \hat{S}_{\hat{f}(i)}| = |C_i \cap S_{f(i)}|$. Therefore, $\sum_{t=1}^{s}|C_{g(t)}\cap S_t| - 1 \leq \sum_{\hat{t}=1}^{\hat{s}}|C_{\hat{g}(\hat{t})}\cap\hat{S}_{\hat{t}}| \leq \sum_{t=1}^{s}|C_{g(t)}\cap S_t| + 1$. By Definition 1, *Precision(C, S)* $-\frac{1}{m} \leq$ *Precision(C,* $\hat{S}$*)* $\leq$ *Precision(C, S)* $+\frac{1}{m}$. We write this as $|$*Precision(C, S)* $-$ *Precision(C,* $\hat{S}$*)*$| \leq \frac{1}{m}$. □

Corollary 3.1. $|$*Precision(C, R)* $-$ *Precision(C,* $\hat{R}$*)*$| \leq \frac{\epsilon}{m}$

Proof of Corollary 3.1. By Theorem 3, $|$*Precision(C, S)* $-$ *Precision(C,* $\hat{S}$*)*$| \leq \frac{1}{m}$ for some arbitrary clustering S and a second clustering $\hat{S}$ that is equivalent to S but with a single data point belonging to a different cluster. Given a GTR R and a corresponding AGTR $\hat{R}$, we can sequentially change the cluster membership of ϵ data points in R to obtain $\hat{R}$. At each step the precision value can change by at most $\pm\frac{1}{m}$. Therefore, $|$*Precision(C, R)* $-$ *Precision(C,* $\hat{R}$*)*$| \leq \frac{\epsilon}{m}$. □

Corollary 3.2. $|$*Precision(C, D)* $-$ *Precision(C,* $\hat{R}$*)*$| \leq \frac{\delta}{m}$

Proof of Corollary 3.2. By Theorem 3, $|$*Precision(C, S)* $-$ *Precision(C,* $\hat{S}$*)*$| \leq \frac{1}{m}$ for some arbitrary clustering S and a second clustering $\hat{S}$ that is equivalent to S but with a single data point belonging to a different cluster. Given a ground truth reference clustering D and a corresponding AGTR $\hat{R}$, we can sequentially change the cluster membership of δ data points in D to obtain $\hat{R}$. At each step the precision value can change by at most $\pm\frac{1}{m}$. Therefore, $|$*Precision(C, D)* $-$ *Precision(C,* $\hat{R}$*)*$| \leq \frac{\delta}{m}$. □

Theorem 4. $|$*Recall(C, S)* $-$ *Recall(C,* $\hat{S}$*)*$| \leq \frac{1}{m}$

Proof of Theorem 4. Let $S = \{S_t\}_{1 \leq t \leq s}$ be an arbitrary partition of M. Let the label translation function $g : \{1...s\} \mapsto \{1...c\}$ be defined as $g(t) = \underset{i}{\operatorname{argmax}} |C_i \cap S_t|$. Suppose some $M_n \in M$, some $S_x \in S$ s.t. $M_n \in S_x$, and some S_y s.t. $S_y \in S$ or $S_y = \{\emptyset\}$. Let $\hat{S} = \{\hat{S}_{\hat{t}}\}_{1 \leq \hat{t} \leq \hat{s}}$ be a clustering identical to S except for one cluster label change, which is given by $\hat{S}_x = S_x - \{M_n\}$ and $\hat{S}_y = S_y \cup \{M_n\}$. Let the function $\hat{g} : \{1...\hat{s}\} \mapsto \{1...c\}$ be defined as $\hat{g}(\hat{t}) = \underset{i}{\operatorname{argmax}} |C_i \cap \hat{S}_{\hat{t}}|$. At minimum $|C_{\hat{g}(x)} \cap \hat{S}_x| = |C_{g(x)} \cap S_x| - 1$ and at maximum $|C_{\hat{g}(x)} \cap \hat{S}_x| = |C_{g(x)} \cap S_x|$. Similarly, at minimum $|C_{\hat{g}(y)} \cap \hat{S}_y| = |C_{g(y)} \cap S_y|$ and at maximum $|C_{\hat{g}(y)} \cap \hat{S}_y| = |C_{g(y)} \cap S_y| + 1$. Because each other clusters in $\hat{S}$ identical to some cluster in S, $\sum_{t=1}^{s} |C_{g(t)} \cap S_t| - 1 \leq \sum_{\hat{t}=1}^{\hat{s}} |C_{\hat{g}(\hat{t})} \cap \hat{S}_{\hat{t}}| \leq \sum_{t=1}^{s} |C_{g(t)} \cap S_t| + 1$. Using Definition 2, we obtain *Recall(C, S)* $- \frac{1}{m} \leq$ *Recall(C,* $\hat{S}$*)* $\leq$ *Recall(C, S)* $+ \frac{1}{m}$. We write this as $|$*Recall(C, S)* $-$ *Recall(C,* $\hat{S}$*)*$| \leq \frac{1}{m}$. □

Corollary 4.1. $|$*Recall(C, R)* $-$ *Recall(C,* $\hat{R}$*)*$| \leq \frac{\epsilon}{m}$

Proof of Corollary 4.1. By Theorem 4, $|$*Recall(C, S)* $-$ *Recall(C,* $\hat{S}$*)*$| \leq \frac{1}{m}$ for some arbitrary clustering S and a second clustering $\hat{S}$ that is equivalent to S but with a single data point belonging to a different cluster. Given a GTR R and a corresponding AGTR $\hat{R}$, we can sequentially change the cluster membership of ϵ data points in R to obtain $\hat{R}$. At each step the recall value can change by at most $\pm\frac{1}{m}$. Therefore, $|$*Recall(C, R)* $-$ *Recall(C,* $\hat{R}$*)*$| \leq \frac{\epsilon}{m}$. □

Corollary 4.2. $|$*Recall(C, D)* $-$ *Recall(C,* $\hat{R}$*)*$| \leq \frac{\delta}{m}$

Proof of Corollary 4.2. By Theorem 4, *Recall(C, S)* $- \frac{1}{m} \leq$ *Recall(C,* $\hat{S}$*)* $\leq$ *Recall(C, S)* $+ \frac{1}{m}$ for some arbitrary clustering S and a second clustering $\hat{S}$ that is equivalent to S but with a single data point belonging to a different cluster. Given a ground truth reference label clustering D and a corresponding AGTR $\hat{R}$,we can sequentially change the cluster membership of ϵ data points in D to obtain $\hat{R}$. At each step the recall value can change by at most $\pm\frac{1}{m}$. Therefore, $|$*Recall(C, D)* $-$ *Recall(C,* $\hat{R}$*)*$| \leq \frac{\delta}{m}$. □

Theorem 5. *If* $\hat{e} \geq e$ *then Precision(C,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m} \leq$ *Precision(C, D)*

Proof of Theorem 5. By Corollary 3.1, $|$*Precision(C, R)* $-$ *Precision(C,* $\hat{R}$*)*$| \leq \frac{\epsilon}{m}$. This can be written as *Precision(C,* $\hat{R}$*)* $- \frac{\epsilon}{m} \leq$ *Precision(C, R)*. By applying Theorem 1, *Precision(C,* $\hat{R}$*)* $- \frac{\hat{\epsilon}}{m} \leq$ *Precision(C,* $\hat{R}$*)* $- \frac{\epsilon}{m} \leq$ *Precision(C, R)* $\leq$ *Precision(C, D)*. □

Theorem 6. *If* $\hat{e} \geq e$ *then Recall(C,* $\hat{R}$*) +* $\frac{\hat{\epsilon}}{m} \geq$ *Recall(C, D)*

Proof of Theorem 6. By Corollary 4.1, $|$*Recall(C, R)* $-$ *Recall(C,* $\hat{R}$*)*$| \leq \frac{\epsilon}{m}$. This can be written as *Recall(C,* $\hat{R}$*)* $+ \frac{\epsilon}{m} \geq$ *Recall(C, R)*. Therefore, by Theorem 2, *Recall(C,* $\hat{R}$*)* $+ \frac{\hat{\epsilon}}{m} \geq$ *Recall(C,* $\hat{R}$*)* $+ \frac{\epsilon}{m} \geq$ *Recall(C, R)* $\geq$ *Recall(C, D)*. □

Corollary 6.1. *If* $\hat{e} \geq e$ *then Recall(C,* $\hat{R}$*) +* $\frac{\hat{\epsilon}}{m} \geq$ *Accuracy(C, D)*

Proof of Corollary 6.1. By Corollary 4.1, $|$*Recall(C, R)* $-$ *Recall(C,* $\hat{R}$*)*$| \leq \frac{\epsilon}{m}$. This can be written as *Recall(C,* $\hat{R}$*)* $+ \frac{\epsilon}{m} \geq$ *Recall(C, R)*. Using Corollary 2.1, *Recall(C,* $\hat{R}$*)* $+ \frac{\hat{\epsilon}}{m} \geq$ *Recall(C,* $\hat{R}$*)* $+ \frac{\epsilon}{m}$ $\geq$ *Recall(C, R)* $\geq$ *Accuracy(C, D)*. □

E Dataset Tables

Table 5: Notable Private Malware Reference Datasets

Name	Samples	Families	Platform	Collection Period
Malsign	142,513	?	Windows	2012 - 2014
MaLabel	115,157	?	Windows	Apr 2015 or earlier
MtNet	1.3 million	98	Windows	Jun 2016 or earlier

Table 6: Notable Public Malware Reference Datasets

Name	Samples	Families	Platform	Collection Period
VX Heavens	271,092	137	Windows	?
Malheur	3,133	24	Windows	2006 - 2009
MalGenome	1,260	49	Android	Aug 2010 - Oct 2011
Drebin	5,560	179	Android	Aug 2010 - Oct 2012
Malicia	11,363	55	Windows	Mar 2012 - Mar 2013
Kaggle	10,868	9	Windows	Feb 2015 or earlier