Transcend: Detecting Concept Drift in Malware Classification Models

Roberto Jordaney[+], Kumar Sharad [*§], Santanu Kumar Dash [*‡], Zhi Wang [*†], Davide Papini [*•], Ilia Nouretdinov[+], and Lorenzo Cavallaro[+]

[+]Royal Holloway, University of London
[§]NEC Laboratories Europe
[‡]University College London
[†]Nankai University
[•]Elettronica S.p.A.

Abstract

Building machine learning models of malware behavior is widely accepted as a panacea towards effective malware classification. A crucial requirement for building sustainable learning models, though, is to train on a wide variety of malware samples. Unfortunately, malware evolves rapidly and it thus becomes hard—if not impossible—to generalize learning models to reflect future, previously-unseen behaviors. Consequently, most malware classifiers become unsustainable in the long run, becoming rapidly antiquated as malware continues to evolve. In this work, we propose Transcend, a framework to identify aging classification models *in vivo* during deployment, much before the machine learning model's performance starts to degrade. This is a significant departure from conventional approaches that retrain aging models retrospectively when poor performance is observed. Our approach uses a statistical comparison of samples seen during deployment with those used to train the model, thereby building metrics for prediction quality. We show how Transcend can be used to identify concept drift based on two separate case studies on Android and Windows malware, raising a red flag before the model starts making consistently poor decisions due to out-of-date training.

1 Introduction

Building sustainable classification models for classifying malware is hard. Malware is mercurial and modeling its behavior is difficult. Codebases of commercial significance, such as Android, are frequently patched against vulnerabilities and malware attacking such systems evolve rapidly to exploit new attack surfaces. Consequently, models that are built through training on older malware often make poor and ambiguous decisions when faced with modern malware—a phenomenon commonly known as *concept drift*. In order to build sustainable models for malware classification, it is important to identify when the model shows signs of aging whereby it fails to recognize new malware.

Existing solutions [12, 15, 23] aim to periodically retrain the model. However, if the model is retrained too frequently, there will be little novelty in the information obtained to enrich the classifier. On the other hand, a loose retraining frequency leads to periods of time where the model performance cannot be trusted. Regardless, the retraining process requires manual labeling of all the processed samples, which is constrained by available resources. Once the label is acquired, traditional metrics such as precision and recall are used to retrospectively indicate the model performance. However, these metrics do not assess the decision of the classifier. For example, hyperplane-based learning models (e.g., SVM) only check the side of the hyperplane where the object lies while ignoring its distance from the hyperplane. This is a crucial piece of evidence to assess non-stationary test objects that eventually lead to concept drift.

A well known approach for qualitative assessment of decisions of a learning model is the probability of fit of test object in a candidate class. Previous work has relied on using fixed probability thresholds to identify best matches [19]. Standard algorithms compute the probability of a sample fitting into a class as a by-product of the classification process. However, since probabilities need to sum up to 1.0, it is likely that for previously unseen test objects which do not belong to any of the classes, the probability may be artificially skewed. To mitigate this issue, Deo et al. propose ad-hoc metrics derived from the two probabilities output by Venn-Abers Predictors (VAP) [5], one of which is perfectly calibrated. Although promising, the approach is unfortunately still in its infancy and does not reliably identify drifting objects (as further elaborated in § 6).

The machine learning community has developed tech-

[*]Research carried out entirely while Post-Doctoral Researchers at Royal Holloway, University of London.

niques that look at objects statistically rather than probabilistically. For example, Conformal Predictor [20] makes predictions with statistical evidence. However, as discussed by Fern and Dietterich[1], this method is not tailored to be used in presence of concept drift.

Nevertheless, statistical assessments seem to overcome the limitations of probabilistic approaches, as outlined in § 2. Still, there are two key issues that need to be addressed before statistical assessments can be used to detect concept drift. First, the assessments have to be agnostic to the algorithm used to build the learning model. This is non-trivial as different algorithms can have different underlying classification mechanisms. Any assessment has to abstract away from the algorithm and identify a universal criteria that treats the underlying algorithm as a black box. Second, and more importantly, auto-computation of thresholds to identify an aging model from an abstract assessment criteria requires a brute force search among scores for the training objects.

In this work, we address both these issues by proposing both meaningful and sufficiently abstract assessment metrics as well as an assessment criteria for interpreting the metrics in an automated fashion. We propose Transcend—a fully parametric statistical framework for assessing decisions made by the classifier to identify concept drift. Central to our contribution, is the translation of the decision assessment problem to a constraint optimization problem which enables Transcend to be parametric with diverse operational goals. It can be bootstrapped with pre-specified parameters that tune its sensitivity to varying levels of concept drift. For example, in applications of critical importance, Transcend can be pre-configured to adopt a strict filtering policy for poor and unreliable classification decisions. While previous work has looked at decision assessment [4, 19], this is the first work that looks at identifying untrustworthy predictions using decision assessment techniques. Thereby, Transcend can be deployed in existing detection systems with the aim of identifying aging models and ameliorating performance in the face of concept drift.

In a nutshell, we make the following contributions:

- We propose conformal evaluator (CE), an evaluation framework to assess the quality of machine learning tasks (§ 2). At the core of CE is the definition of non-conformity measure derived from the ML algorithm under evaluation (AUE) and feature set (§ 2.1). This measure builds statistical metrics to quantify the AUE quality and statistically support goodness of fit of a data point into a class according to the AUE (§ 2.4).
- We build assessments on top of CE's statistical metrics to evaluate the AUE design and understand statistical distribution of data to better capture AUE's generalization and class separations (§ 3).
- We present Transcend, a fully tunable classification system that can be tailored to be resilient against concept drift to varying degrees depending on user specifications. This versatility enables Transcend to be used in a wide variety of deployment environments where the cost of manual analysis is central to classification strategies. (§ 3.3)
- We show how CE's assessments facilitate Transcend to identify suitable statistical thresholds to detect decay of ML performance in realistic settings (§ 4). In particular, we support our findings with two case studies that show how Transcend identifies concept drift in binary (§ 4.1) and multi-class classification (§ 4.2) tasks.

[1] A. Fern and T. Dietterich. "Toward Explainable Uncertainty". https://intelligence.org/files/csrbai/fern-slides-1.pdf

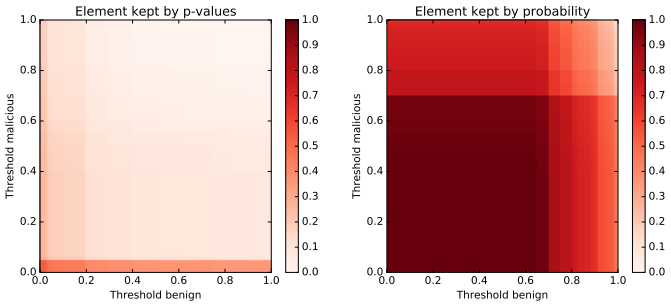
2 Statistical Assessment: Why and How?

In this section we discuss the significance of statistical techniques for decision assessment, which form the core of conformal evaluator. A statistical approach to decision assessment considers each decision in the context of previously made decisions. This is different to a probabilistic assessment where the metric is indicative of how likely a test object is to belong to a class. In contrast, statistical techniques answer the question: *how likely is the test object to belong to a class compared to all of its other members?* The contextual evidence produced by statistical evidence is a step beyond standard probabilistic evidence and typically gives stronger guarantees on the quality of the assessment. Our work dissects Conformal Predictor (CP) [24] and extracts its sound statistical foundations to build conformal evaluator (CE). In the following section, we provide further details, while we forward the reader to § 6 for a full comparison between CP and CE.

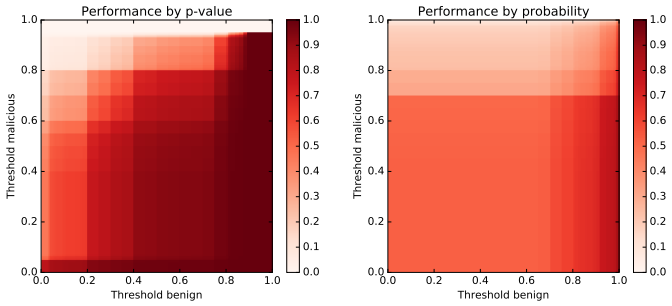
2.1 Non-conformity Measure

Classification is usually based on a *scoring function* which, given a test object z^*, outputs a *prediction score* $F_{\mathcal{D}}(l, z^*)$, where $\mathcal{D}$ is the dataset of training objects and l is a label from the set of possible object labels $\mathcal{L}$.

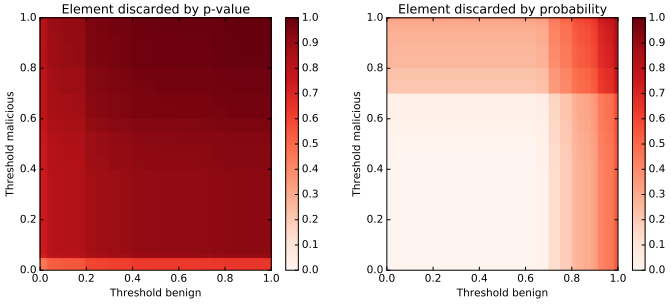
The scoring function can be used to measure the difference between a group of objects belonging to the same class (e.g., malware belonging to the same family) and a new object (i.e., a sample). In Transcend, the *non-conformity measure* (NCM) is computed directly from the scoring function of the algorithm. Thus, conformal



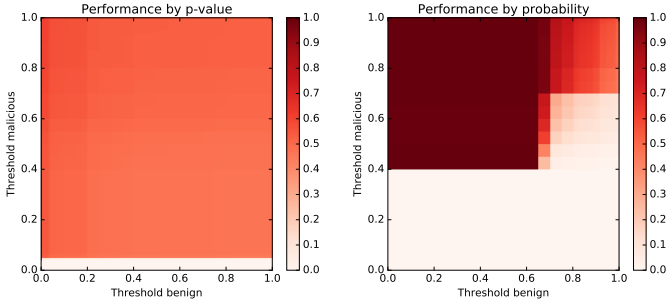
(a) Elements above the threshold



(b) Performance of elements above the threshold



(c) Elements below the threshold



(d) Performance of elements below the threshold

Figure 1: Performance comparison between *p-value* and *probability* for the objects above and below the threshold used to accept the algorithm's decision. The p-values are given by CE with SVM as non-conformity measure, the probabilities are given directly by SVM. As we can see from the graph, p-values tend to contribute to a higher performance of the classifier, identifying those (drifting) objects that would have been erroneously classified.

evaluation is agnostic to the algorithm, making it versatile and compatible with multiple ML algorithms; it can be applied on top of any classification or clustering algorithm that uses a score for prediction.

We note that some algorithms already have built-in quality measures (e.g., the distance of a sample from the hyperplane in SVM). However, these are algorithm specific and cannot be directly compared with other algorithms. On the other hand, Transcend unifies such quality measures through uniform treatment of non-conformity in an algorithm-agnostic manner.

2.2 P-values as a Similarity Metric

At the heart of conformal evaluation is the non-conformity measure—a real-valued function $A_{\mathcal{D}}(C\backslash z, z)$, which tells how different an object z is from a set C. The set C is a subset of the data space of object $\mathcal{D}$. Due to the real-valued range of non-conformity measure, conformal evaluator can be readily used with a variety of machine learning methods such as support-vector machines, neural networks, decision trees and Bayesian prediction [20] and others that use real-valued numbers (i.e., a similarity function) to distinguish objects. Such flexibility enables Transcend to assess a wide range of algorithms.

Conformal evaluation computes a notion of similarity through p-values. For a set of objects $\mathcal{K}$, the p-value $p^C_{z^*}$ for an object z^* is the proportion of objects in class $\mathcal{K}$ that are at least as dissimilar to other objects in C as z^*. There are two standard techniques to compute the p-values from $\mathcal{K}$: *Non-Label-Conditional* (employed by *decision* and *alpha* assessments outlined in § 3.1 and § 3.2), where $\mathcal{K}$ is equal to $\mathcal{D}$, and *Label-Conditional* (employed by the concept drift detection described in § 3.3), where $\mathcal{K}$ is the set of objects C with the same label. The calculations for the non-conformity measures for the test object and the set of objects in $\mathcal{K}$ is shown in equation 1 and 2 respectively. The computation of p-value for the test object is shown in equation 3.

$$\alpha_{z^*} = A_{\mathcal{D}}(C, z^*) \quad (1)$$

$$\forall i \in \mathcal{K}.\alpha_i = A_{\mathcal{D}}(C \setminus z_i, z_i) \quad (2)$$

$$p^C_{z^*} = \frac{|\{j : \alpha_j \geq \alpha_{z^*}\}|}{|\mathcal{K}|} \quad (3)$$

P-values compute an algorithm's credibility and confidence, crucial for decision assessments (§ 2.4).

2.3 P-values vs. Probabilities

One might question the utility of p-value over probability of a test object belonging to a particular class. Probabilities are computed by most learning algorithms as qualitative feedback for a decision. SVM uses Platt's scaling to compute probability scores of an object belonging to a class while a random forest averages the decision of individual trees to reach a final prediction [3]. In this section, we discuss the shortcomings of using probabilities for decision assessment as shown in §4.1.1 and §4.2. Additionally, we also provide empirical evidence in favor of p-values as a building block for decision assessment.

P-values offer a significant advantage over probabilities when used for decision assessment. Let us assume that the test object z^* has p-values of $p^1_{z^*}, p^2_{z^*} \cdots p^k_{z^*}$ and probability of $r^1_{z^*}, r^2_{z^*} \cdots r^k_{z^*}$ of belonging to classes $l_1, l_2 \cdots l_k$ (which is the set of all classes in $\mathcal{L}$). In the case of probabilities, $\Sigma_i r^i_{z^*}$ must sum to 1.0. Now, lets consider a 2-class problem. If z^* does not belong to either of the classes, and the algorithm computes a low probability score $r^1_{z^*} \sim 0.0$, then $r^2_{z^*}$ would artificially tend to 1.0. In other words, if we use probabilities for decision assessment it is likely that we might reach an incorrect conclusion for previously unseen samples. P-values on the other hand are not constrained by such limitations. It is possible for both $p^1_{z^*}$ and $p^2_{z^*}$ to be a low value for the case of a previously unseen sample. This is true also when p-values are built using probability as NCM. To calculate the probability of a test sample, only information belonging to the test samples are used (e.g., distance to the hyperplane in the case SVM or ratio of decisions for one class in the case of random forest). Instead, a p-value is computed comparing the scores of all the samples in a class (see equation 1 and 2).

We further elaborate on this by training an SVM classifier with Android malware objects from the Drebin dataset [2] and by testing it using objects from a drifted dataset (the Marvin dataset [14], see § 4 for details). Then, we apply a threshold to accept the decision of the classifier only if a certain level of certainty is achieved. Figure 1 shows the average of F1-score for malicious and benign classes after the application of the threshold for the objects that fall above (Figure 1b) and below it (Figure 1d). Figure 1 also shows the ratio of objects retained (Figure 1a) and rejected (Figure 1c). Figure 1b shows that the use of p-values produces better performance as it identifies more objects to reject than probabilities (Figure 1a). Here, filtering out a high number of objects is correct as they are drifting from the trained model. Keeping them would degrade the performance of the algorithm (Figures 1c and 1d). The threshold is applied to the testing objects; we present case studies in § 4.1, which show how to derive it from the training dataset.

2.4 Statistical Decision Assessment

This section introduces and discusses CE metrics used to assess the classification decisions. The techniques for interpreting these metrics are discussed in § 3.

Algorithm Credibility. The first evaluation metric for assessing classification decision on a test object is *algorithm credibility*. $A_{Cred}(z^*)$ is defined as the p-value for the test object z^* corresponding to the label chosen by the algorithm under analysis. As discussed, the p-value measures the fraction of objects within $\mathcal{K}$, that are at least as different from the set of objects C as the new object z^*. A high credibility value means that z^* is very similar to the objects in the class chosen by the classifier. Although credibility is a useful measure of classification quality, it only tells a partial story. There may potentially be high p-values for multiple labels indicating multiple matching labels for the test object which the classification algorithm has ignored. On the other hand, a low credibility value is an indicator of either z^* being very different from the objects in the class chosen by the classifier or the object being poorly identified. These two observations show that credibility alone is not sufficient for reliable decision assessment. Hence, we introduce another measure to gauge the non-performance of the classification algorithm—*algorithm confidence*.

Algorithm Confidence. For a given choice (e.g., assigning z to a class l_i), confidence tells how certain or how committed the evaluated algorithm is to the choice. Formally, it measures how distinguishable is the new object $z^* \in l_i$ from other classes l_j with $j \neq i$. We define the algorithm confidence as 1.0 minus the maximum p-value among all p-values except the *p-value* chosen by the algorithm (i.e., algorithm credibility): $A_{Conf}(z^*) = 1 - \max(P(z^*) \setminus A_{Cred}(z^*))$ where, $P(z^*) = \{p^{l_i}_{z^*} : l_i \in \mathcal{L}\}$

$P(z^*)$ is the set of p-values associated to the possible choices for the new object z^*. The highest value of confidence is reached when the algorithm credibility is the highest p-value. It may happen that the choice made by the algorithm is not attached to the highest p-value, suggesting that the confidence is sub-optimal. Results in § 4 show that this provides valuable insights, especially when the method under assessment makes choices with low values of confidence and credibility. Low algorithm confidence indicates that the given object is similar to other classes as well. Depending on the algorithm credibility, this indication may imply that the decision algorithm is not able to uniquely identify the classes or, that the new object looks similar to two or more classes.

Finally, we note that algorithm confidence and credibility are not biased by the number of classes in a dataset as popular measures, such as precision and recall [13]. Thus CE's findings are more robust to dataset changes.

3 Framework Description

Previous section introduced conformal evaluation along with the two metrics that we use for decision assessment: algorithm confidence and algorithm credibility. Transcend uses two techniques to evaluate the quality of an algorithm employed on a given dataset: (*i*) *Decision assessment*—evaluates the robustness of the predictions made by the algorithm; and (*ii*) *Alpha assessment*—evaluates the quality of the non-conformity measure. We combine these assessments to enable the detection of concept drift (§3.3).

3.1 Decision Assessment

Conformal evaluator qualitatively assesses an algorithm's decision by assigning a class $l \in \mathcal{L}$ as predicted by the algorithm to each new object z^* and computing its algorithm credibility and confidence.

Hence, four possible scenarios unfold: (*i*) High algorithm confidence, high algorithm credibility—the best situation, the algorithm is able to correctly identify a sample towards one class and one class only. (*ii*) High algorithm confidence, low algorithm credibility—the algorithm is not able to correctly associate the sample to any of the classes present in the dataset. (*iii*) Low algorithm confidence, low algorithm credibility—the algorithm gives a label to the sample but it seems to be more similar to another label. (*iv*) Low algorithm confidence, high algorithm credibility—according to the algorithm, it seems that the sample is similar to two or more classes.

The measures are then grouped into two sets —correct or wrong— which represents values for correctly and wrongly classified objects. Subsequently, values are averaged and their standard deviation is also computed, this is done for every class $l \in \mathcal{L}$, to study whether the algorithm works consistently for all classes or if there are difficult classes that the algorithm has trouble dealing with. This assessment, performed during the design phase of the algorithm, helps us to decide the cutoff threshold for a deployed scenario to separate the samples with enough statistical evidence of correctness.

Comparing the results obtained for correct and wrong choices produces interesting results. For correct choices it would be desirable to have high credibility and confidence. Conversely, for wrong choices it would be desirable to have low credibility and high confidence. The divergence from these scenarios helps understand whether the algorithm takes strong decisions, meaning that there is a strong statistical evidence to confirm its decisions, or, in contrast, if the decisions taken are easily modified with a minimal modification of the underlying data.

By looking at the outcome of decision assessment, it is possible to understand whether the choices made by an algorithm are supported with statistical evidence. Otherwise, it is possible to get an indication where to look for possible errors or improvements, i.e., which classes are troublesome, and whether further analysis is needed, e.g. by resorting to the alpha assessment.

3.2 Alpha Assessment

In addition to the decision assessment, which evaluates the output of a similarity-based classification/clustering algorithm, another important step in understanding the inner workings and subtleties of the algorithm includes analyzing the data distribution of the algorithm under evaluation. Owing mainly to practical reasons, malware similarity-based algorithms are developed around a specific dataset. Hence there is often the possibility of the algorithm to over-fit its predictions to the dataset. Overfitting results in poor performance when the algorithm analyses new or unknown datasets [13]. Despite employing techniques to avoid over-fitting, the best way to answer this question is to try the algorithm against as many datasets as possible. We show that conformal evaluator can help solve this problem, when no more than one dataset is available.

The *alpha assessment* analysis takes into account how appropriate is the similarity-based algorithm when applied to a dataset. It can detect if the final algorithm results still suffer from over-fitting issues despite the efforts of minimizing it using common and well known techniques (e.g., cross validation).

Furthermore, the assessment enables us to get insights on classes (e.g., malware families), highlighting how the similarity-based method works against them. Researchers may gather new insights on the peculiarities of each class, which may eventually help to improve feature engineering and the algorithm's performance, overall.

First, for each object $z_j \in D$, where l_j is z_j's true class, we compute its p-values against every possible $l \in \mathcal{L}$. We then plot the boxplot [10], containing the p-values for each decision. By aligning these boxplots and grouping them by class/cluster, we can see how much an element of class/cluster j resembles that of another one, allowing for reasoning about the similarity-based algorithm itself.

In § 4 we present case studies where we statistically evaluate the quality behind performances of algorithms within the conformal evaluator framework.

3.3 Concept Drift

We now describe the core of Transcend's concept drift detection and object filtering mechanism. It must be stressed here that we look at concept drift from the perspective of a malware analysis team. Consequently, the

severity of the drift is a subjective issue. For critical applications, even a few misclassifications can cause major issues. Consequently, the malware analysis team would have a high standard for abandoning an aging classification model. Therefore, we make the concept drift detection in Transcend parametric in two dimensions: the desired performance level (ω) and the proportion of samples in an epoch that the malware analysis team is willing to manually investigate (δ). The analyst selects ω and δ as degrees of freedom and Transcend will detect the corresponding concept drift point constrained by the chosen parameters. The goal is to find thresholds that best separate the correct decisions from the incorrect ones based on the quality metrics introduced by our analysis. These thresholds are computed on the training dataset but are enforced on predictions during deployment (for which we do not have labels). The rationale is very simple: predictions with p-values above such thresholds would identify objects that likely fit (from a statistical perspective) in the model; such classifications should be trusted. Conversely, objects out of predictions with p-values smaller than such thresholds should not be trusted as there is lack of statistical evidence to support their fit in the model.

What happens to untrustworthy predictions (and related test—likely drifted—objects) is out of the scope of this work. It is reasonable to envision a pipeline that would label drifted objects to retrain the machine learning model, eventually. While this raises several challenges (e.g., how many objects need to be labeled, how much resources can be invested in the process), we would like to remark the fact that is only possible once concept drift is detected: the goal of this research. Not only, Transcend plays a fundamental role in the identification of drifting objects and thus in the understanding of when a prediction should be trusted or not, but its metrics can also aid in selecting what drifted objects should be labeled first (e.g., those with low p-values as are the one that have drifted the most from the trained model).

The following discussion assumes two classes of data, malicious and benign, but it is straightforward to extend it to a multiclass scenario.

We define the function $f : \mathbb{B} \times \mathbb{M} \to \Omega \times \Delta$ that maps a pair of thresholds in the benign and malicious class and outputs the performance achieved and the number of decisions accepted. Here, the number of decisions accepted refers to the percentage of the algorithm outputs with a p-value (for benign or malicious classes, depending on the output itself) greater than the corresponding threshold; performance means the percentage of correct decisions amongst the accepted ones. $\mathbb{B}$, $\mathbb{M}$, Ω and Δ are the domains of the possible thresholds on benign samples, malicious samples, desired performance and classification decisions accepted, respectively. During training of our classifier, we iterate over all values of the benign threshold t'_b and the malicious threshold t'_m, at a pre-specified level of granularity, in the domain of $\mathbb{B}$ and $\mathbb{M}$, respectively. Let us assume f gives the output $f{:}f(t'_b,t'_m) = (\omega',\delta')$

To detect concept drift during deployment with a prespecified threshold of either ω or δ, we need to define an inverse of f which we call $f^{-1} : \Lambda \to \mathbb{B} \times \mathbb{M}$ where $\Lambda = \Omega \cup \Delta$. When supplied with either ω or δ, f^{-1} would give us two thresholds t_b and t_m which would help Transcend decide when to accept the classifier's decision and when to ignore it. Notice that with a conjoined domain Λ, which only accepts either ω or δ, it is not trivial to reconstruct the values of t_b and t_m. For every value of ω, there could be multiple values for δ. Therefore, we adopt a simple heuristic to compute t_b and t_m whereby we maximize the second degree of freedom given the first. For example, given ω, we find t_b and t_m for every possible value of δ and pick the t_b and t_m that maximizes δ. The formulation is exactly the same when δ is used as an input. The formal equations for the inverse functions are:

$$\begin{aligned} \Gamma &= \{x : x \in \forall t'_b \forall t'_m . f(t'_b,t'_m))\} \\ f^{-1}(\omega) &= \{(t_b,t_m) : \delta \in f(t_b,t_m) = max(\forall \delta' \in \Gamma)\} \\ f^{-1}(\delta) &= \{(t_b,t_m) : \omega \in f(t_b,t_m) = max(\forall \omega' \in \Gamma)\} \end{aligned}$$

Comparison with Probability. The algorithm used as inner non-conformity measure (NCM) in CE may have a pre-defined quality metric to support its own decision-making process (e.g., probability). Hence, we also compare the ability of detecting concept drift of the algorithm's internal metric with CE metrics. The thresholds are extracted from the true positive samples, because we expect the misclassified samples to have a lower value of the quality metric: it seems rather appropriate to select a higher threshold to highlight decisions the algorithm would likely make wrong. We compare our metrics with probability metrics derived from two different algorithms for our case studies. In the first case study (see, § 4.1), we compare our metrics with SVM probabilities derived from Platt's scaling [17]; on the other hand, the second case study (see, § 4.2) uses the probabilities extracted from a random forest [3] model. This comparison shows the general unsuitability of the probability metric to detect concept drift. For example, the threshold obtained from the first quartile of the true positive p-value distribution is compared with that of the first quartile of the true positive probability distribution, and so forth.

The reasoning outlined above still holds when a given algorithm, adapted to represent the non-conformity measure, uses raw score as its decision-making criteria. For instance, the transformation of a raw score to a probability value is often achieved through a monotonic transformation (e.g., Platt's scaling, for SVM) that does not affect the p-value calculation. Such algorithms do not

provide a raw score for representing the likelihood of an alternative hypothesis (e.g., that the test object does not belong to any of the classes seen in the training). Moreover, a threshold built from a raw score lacks context and meaning; conversely, combining raw scores to compute p-values provides a clear statistical meaning, able of quantifying the observed drift in a normalized scale (from 0.0 to 1.0), even across different algorithms.

CE can also provide quality evaluation that allows switching the underlying ML-based process to a more computationally intensive one on classes with poor confidence [4]. Our work details the CE metrics used by Dash et al. [4] and extends it to identify concept drift.

4 Evaluation

To evaluate the effectiveness of Transcend, we introduce two case studies: a binary classification to detect malicious Android apps [2], and a multi-class classification to classify malicious Windows binaries in their respective family [1]. The case studies were chosen to be representative of common supervised learning settings (i.e., binary and multi-class classification), easy to reproduce[2], and of high quality[3].

Binary Classification Case Study. In [2], Arp et al. present a learning-based technique to detect malicious Android apps. The approach, dubbed Drebin, relies on statically extracting features, such as permissions, Intents, APIs, strings and IP addresses, from Android applications to fuel a linear SVM. Hold-out validation results (66-33% split in training-testing averaged over ten runs) reported TPR of 94% at 1% FPR. The Drebin dataset was collected from 2010 to 2012 and the authors released the feature set to foster research in the field.

To properly evaluate a drifting scenario in such settings, we also use Marvin [14], a dataset that includes benign and malicious Android apps collected from 2010 and 2014. The rationale is to include samples drawn from a timeline that overlaps with Drebin as well as newer samples that are likely to drift from it (duplicated samples were removed from the Marvin dataset to avoid biasing the results of the classifier). Table 1 provides details of the datasets.

Section 4.1 outlines this experiment in detail; however, without any loss of generality, we can say models are trained using the Drebin dataset and tested against the Marvin one. In addition, the non-conformity measure we instantiate CE with is the distance of testing objects from the SVM hyperplane, as further elaborated in § 4.1.1.

Multiclass Classification Case Study. Ahmadi et al. [1] present a learning-based technique to classify Windows malware in corresponding family of threats. The approach builds features out of machine instructions' opcodes of Windows binaries as provided by Microsoft and released through the Microsoft Malware Classification Challenge competition on Kaggle [11]—a well-known platform that hosts a wide range of machine learning-related challenges. Ahmadi et al. rely on eXtreme Gradient Boosting (XGBoost) [21] for classification. It is based on gradient boosting [18] and, like any other boosting technique, it combines different weak prediction models to create a stronger one. In particular, the authors use XGBoost with decision trees.

Table 2 provides details of the Microsoft Windows Malware Classification Challenge dataset. To properly evaluate a drifting scenario we omit the family Tracur from the training dataset, as further elaborated in § 4.2. In this setting, a reasonable conformity measure that captures the likelihood of a test object o to belong to a given family $l \in \mathcal{L}$ is represented by the probability p that o belongs to $l \in \mathcal{L}$, as provided by decision trees. We initialize conformal evaluator with $-p$ as non-conformity measure, because it captures the dissimilarities. Please note we do not interpret $-p$ as a probability anymore (probability ranges from 0 to 1), but rather as a (non-conformity) score CE builds p-values from (see § 2).

We would like to remark that these case studies are chosen because they are general enough to show how concept drift affects the performance of the models. This is not a critique against the work presented in [1, 2]. Rather, we show that even models that perform well in closed world settings (e.g., k-fold cross validation), eventually decay in the presence of non-stationary data (concept drift). Transcend identifies *when* this happens in operational settings, and provides indicators that allow to establish whether one should *trust* a classifier decision or not. In absence of retraining, which requires samples relabeling, the ideal net effect would then translate to having high performance on non-drifting objects (i.e., those that fit well into the trained model), and low performance on drifting ones.

In a nutshell, our experiments aim to answer the following research questions:

RQ1: *What insights do CE statistical metrics provide?* Intuitively, such metrics provide a quantifiable level of quality of the predictions of a classifier.

RQ2: *How can CE statistical metrics detect concept drift in binary and multiclass classification?* Intuitively, we can interpret quality metrics as thresholds: predictions of tested objects whose quality fall below such

[2]The work in [2] released feature sets and details on the learning algorithm, while we reached out to the authors of [1], which shared datasets and the learning algorithm's implementation with us.

[3]The work in [2] was published in a top-tier venue, while the work in [1] scored similar to the winner of the Kaggle's Microsoft Malware Classification Challenge [11].

DREBIN DATASET		MARVIN DATASET	
Type	Samples	Type	Samples
Benign	123 435	Benign	9 592
Malware	5 560	Malware	9 179

Table 1: Binary classification case study datasets [2].

MICROSOFT MALWARE CLASSIFICATION CHALLENGE DATASET			
Malware	Samples	Malware	Samples
Ramnit	1 541	Obfuscator.ACY	1 228
Lollipop	2 478	Gatak	1 013
Kelihos˙ver3	2 942	Kelihos˙ver1	398
Vundo	4 75	Tracur	751

Table 2: Multiclass classification case study datasets [1].

thresholds should be marked as untrustworthy, as they drift away from the trained model (see §3.3).

We elaborate this further in § 4.1 and § 4.2 for binary and multiclass classification tasks, respectively.

4.1 Binary Classification Case Study

This section assesses the quality of the predictions of Drebin[4], the learning algorithm presented in [2]. We reimplemented Drebin and achieved results in line with those reported by Arp et al. in absence of concept drift (0.95 precision and 0.92 recall, and 0.99 precision and 0.99 recall for malicious and benign classes, respectively on hold out validation with 66-33% training-testing Drebin dataset split averaged on ten runs).

Figure 2a shows how CE's decision assessment supports such results. In particular, the average algorithm credibility and confidence for the correct choices are 0.5 and 0.9, respectively. This reflects a high prediction quality: correctly classified objects are very different (from a statistical perspective) to the other class (and an average p-value of 0.5 as algorithm credibility is expected due to mathematical properties of the conformal evaluator). Similar reasoning applies for incorrect predictions, which are affected by a poor statistical support (average algorithm credibility of 0.2).

Figure 2b shows CE's alpha assessment of Drebin. We plot this assessment as a boxplot to show details of the p-value distribution. The plot shows that the p-value distribution for the wrong predictions (i.e., second and third column) is concentrated in the lower part of the scale (less than 0.1), with few outliers; this means that, on average, the p-value of the class which is not the correct one, is much lower than the p-value of the correct predictions. Benign samples (third and fourth columns) seem more stable to data variation as the p-values for benign and malicious classes are well separated. Conversely, the p-value distribution of malicious samples (first and second columns) is skewed towards the bottom of the plot; this implies that the decision boundary is loosely defined, which may affect the classifier results in the presence of concept drift. A direct evaluation of the confusion matrix and associated metrics does not provide the ability to see decision boundaries nor predictions (statistical) quality.

4.1.1 Detecting Concept Drift

This section presents a number of experiments to show how Transcend identifies concept drift and correctly marks as untrustworthy the decisions the NCM-based classifier predicts erroneously.

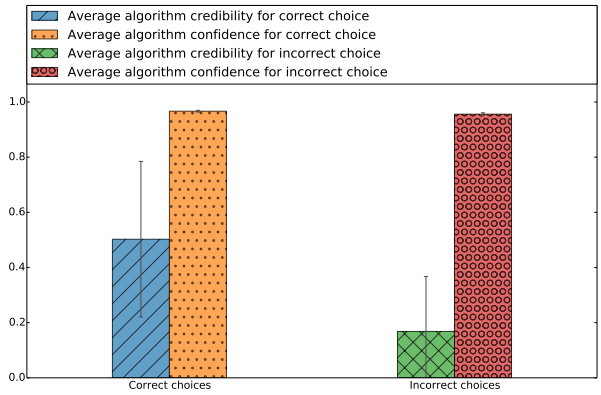
We first show how the performance of the learning model introduced in [2] decays in the presence of concept drift. To this end, we train a model with the Drebin dataset [2] and we test it against 9,000 randomly selected malicious and benign Android apps (with equal split) drawn from the Marvin dataset [14]. The confusion matrix in Table 3a clearly shows how the model is affected by concept drift as it reports low precision and recall for the positive class representing malicious objects[5]. This is further outlined in Figure 3a, which shows how the p-value distribution of malicious objects is pushed towards low values (poor prediction quality).

Table 3b shows how enforcing cut-off quality thresholds affect—by improving—the performance of the same learning algorithm. For this experiment, we divided the Drebin dataset in training and calibration sets with a 90-10% averaged over 10 rounds. This ensures that each object in the dataset has a p-value. We then asked Transcend to identify suitable quality thresholds (cfr § 3.3) with the aim to maximize the F1-score as derived by the calibration dataset, subject to a minimum F1-score of 0.99 and a minimum percentage of kept element of 0.76[6]. It is worth noting that such thresholds are derived from the calibration dataset but are enforced to detect concept drift on a testing dataset. Results show how flagging predictions of testing objects with p-values below the cut-off thresholds as unreliable improves precision and recall for the positive (malicious) class, from 0.61 to 0.89 and from 0.36 to 0.76, respectively.

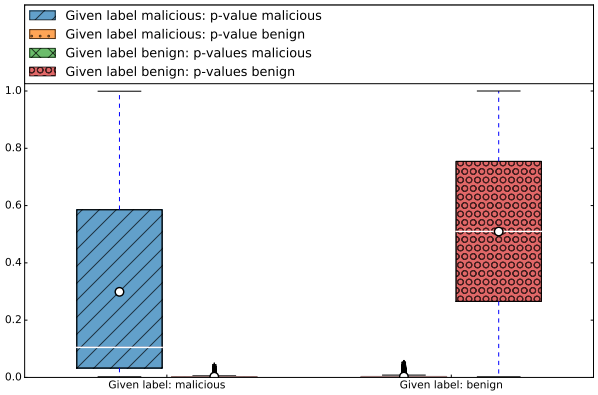
[4]Unless otherwise stated, we refer to Drebin as both the learning algorithm and the dataset outlined in [2].

[5]Drebin spans the years 2010–2012 while Marvin covers from 2010 to 2014. Most of the Drebin's features capture information (e.g., string and IP addresses) that is likely to change over time, affecting the ability of the classifier to identify non-stationary data.

[6]In [2], Arp et al. report a TPR of 94% at a FPR of 1%. Such metrics do not rule out the possibility of having 0.99 as F1-score; if that is a plausible constraint, Transcend's parametric framework will find a suitable solution.



(a) Decision assessment for the binary classification case study (Drebin [2]) with the original dataset. Correct predictions are supported by a high average algorithm credibility and confidence, while incorrect ones have a low and a high algorithm credibility and confidence, respectively. Overall, positive results supported by a strong statistical evidence.



(b) Alpha assessment for the binary classification case study (Drebin [2]) with the original dataset. Benign samples are well separated from malicious ones, especially when the assigned label is benign; this provides a clear statistical support that positively affect the quality of predictions.

Figure 2: Binary Classification Case Study (Drebin [2]): Decision assessment and Alpha assessment.

	Assigned label		
Sample	Benign	Malicious	**Recall**
Benign	4 498	2	1
Malicious	2 890	1 610	0.36
Precision	0.61	1	

(a)

	Assigned label		
Sample	Benign	Malicious	**Recall**
Benign	4 257	2	1
Malicious	504	1 610	0.76
Precision	0.89	1	

(b)

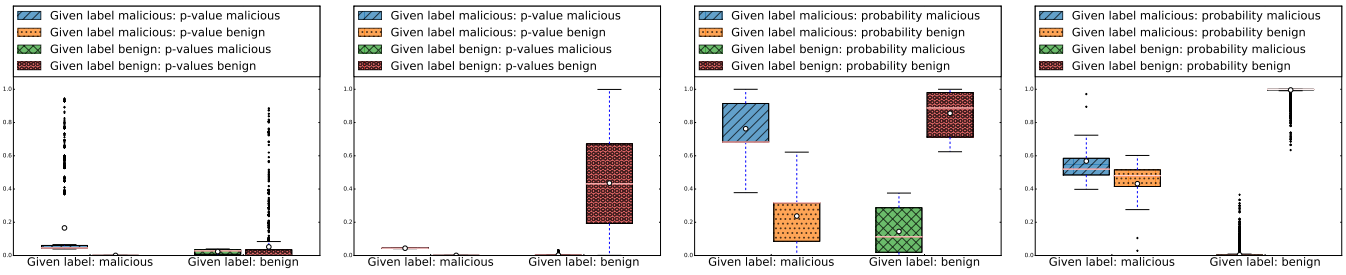
	Assigned label		
Sample	Benign	Malicious	**Recall**
Benign	4 413	87	0.98
Malicious	255	4 245	0.94
Precision	0.96	0.98	

(c)

Table 3: Binary classification case study ([2]). Table 3a: confusion matrix when the model is trained on Drebin and tested on Marvin. Table 3b: confusion matrix when the model is trained on Drebin and tested on Marvin with p-value-driven threshold filtering. Table 3c: retraining simulation with training samples of Drebin as well as the filtered out element of Marvin of Table 3b (2386 malicious samples and 241 benign) and testing samples coming from another batch of Marvin samples (4500 malicious and 4500 benign samples). The fate of the drifting objects is out of scope of this paper as that would require to solve a number of challenges that arise *once* concept drift is identified (e.g., randomly sampling untrustworthy samples according to their p-values, effort of relabeling depending on available resources, model retraining). We nonetheless report the result of a realistic scenario in which objects drifting from a given model, correctly identified by Transcend, represent important information to retrain the model and increase its performance (assuming a proper labeling as briefly sketched above).

	TPR of kept elements		FPR of kept elements		TPR of discarded elements		FPR of discarded elements		MALICIOUS kept elements		BENIGN kept elements	
	p-value	probability	p-value	probability	p-value	probability	p-value	probability	p-value	probability	p-value	probability
1st quartile	0.9045	0.6654	0.0007	0.0	0.0000	0.3176	0.0000	0.0013	0.3956	0.1156	0.6480	0.6673
Median	0.8737	0.8061	0.0000	0.0	0.3080	0.3300	0.0008	0.0008	0.0880	0.0584	0.4136	0.4304
Mean	0.8737	0.4352	0.0000	0.0	0.3080	0.3433	0.0008	0.0018	0.0880	0.1578	0.4136	0.7513
3rd quartile	0.8723	0.6327	0.0000	0.0	0.3411	0.3548	0.0005	0.0005	0.0313	0.0109	0.1573	0.1629

Table 4: Binary classification case study ([2]): examples of thresholds. From the results we can see that increasing the threshold will lead to keep only the sample where the algorithm is sure about. The number of discarded samples is very subjective to the severity of the shift in the dataset, together with the performance of those sample it is clear the advantage of the p-value metric compared to the probability one.



(a) (b) (c) (d)

Figure 3: Binary Classification Case Study: p-value and probability distribution for true malicious and benign samples when the model is trained on Drebin dataset and tested on Marvin. Graph (a): p-value distribution for true malicious samples. Graph (b): p-value distribution of true benign samples. Graph (c): probability distribution of true malicious samples. Graph (d): probability distribution of true benign samples.

We would like to remark that drifting objects are still given a label as the output of a classifier prediction; Transcend flags such predictions as untrustworthy, de-facto limiting the mistakes the classifier would likely make in the presence of concept drift. It is clear that one needs to deal with such objects, eventually. Ideally, they would represent an additional dataset that, once labeled properly, would help retraining the classifier to predict similar objects. This opens a number of challenges that are out of the scope of this work; however, one could still rely on CE's metrics to prioritize objects that should be labeled (e.g., those with low p-values as they are the one the drift the most from the model). This might require to randomly sample drifting objects once enough data is available as well as understanding how much resources one can rely on for data labeling. It is important to note that Transcend plays a fundamental role in this pipeline: it identifies concept drift (and, thus, untrustworthy predictions), which gives the possibility of start reasoning on the open problems outlined above.

The previous paragraphs show the flexibility of the parametric framework we outlined in § 3.3, on an arbitrary yet meaningful example, where statistical cut-off thresholds are identified based on an objective function to optimize, subject to specific constraints. Such goals are however driven by business requirements (e.g., TPR vs FPR) and resource availability (e.g., malware analysts available vs number of likely drifting samples—either benign or malicious—for which we should not trust a classifier decision) thus providing numerical example might be challenging. To better outline the suitability of CE's statistical metrics (p-values) in detecting concept drift, we provide a full comparison between p-values and probabilities as produced by Platt's scaling applied to SVM. We summarize a similar argument (with probabilities derived from decision trees) for multiclass classification tasks in § 4.2.

Comparison with Probability. In the following, we compare the distributions of p-values, as derived from CE, and probabilities, as derived from Platt's scaling for SVM, in the context of [2] under the presence of concept drift (i.e., training on Drebin, testing on Marvin as outlined). The goal of this comparison is to understand which metric is better-suited to identify concept drift.

Figure 3a shows the alpha assessment of the classifications shown in Table 3a. The figure shows the distribution of p-values when the true label of the samples is malicious. Correct predictions (first and second columns), reports p-values (first column) that are are slightly higher than those corresponding to incorrect ones (second column), with a marginal yet well-marked separation as compared to the values they have for the incorrect class (third and fourth columns). Thus, when wrong predictions refer to the benign class, the p-values are low and show a poor fit to both classes. Regardless of the classifier outcome, the p-value for each sample is very low, a likely indication of concept drift.

Figure 3b depicts the distribution of p-values when true label of the samples is benign. Wrong predictions (first and second columns) report p-values representing benign (second column) and malicious (first column) classes to be low. Conversely, correct predictions (third and fourth columns) represent correct decisions (fourth column) and have high p-values, much higher compared to the p-values of the incorrect class (third column). This is unsurprising as benign samples have data distributions that do not drift with respect to malicious ones.

A similar reasoning can be followed for Figures 3c and 3d. Contrary to the distribution of p-values, probabilities are constrained to sum up to 1.0 across all the classes; what we observe is that probabilities tend to be skewed towards high values even when predictions are wrong. Intuitively, we expect to have poor quality on *all* the classes of predictions in the presence of a drifting

scenario: while probabilities tend to be skewed, CE's statistical metrics (p-values) seem better-suited at this task.

So far, we have seen how Transcend produces statistical thresholds to detect concept drift driven by predefined goals under specific constraints. In addition, the analysis of p-value and probability distributions highlighted how the former seem to be better-suited than probabilities to identify concept drift. In the following paragraphs, we show how CE's statistical metrics provide thresholds that *always outperform* probabilities in detecting concept drift. Figure 6 in Appendix 7 provides a thorough comparison. For simplicity, here, we focus the attention on the 1st and 3rd quartile, the median and the average of the distribution of p-values and probabilities as potential cut-off thresholds, as shown in Table 4.

Intuitively speaking, a successful technique not only would achieve high performances on correct predictions, but it would also report poor performances on drifting objects. This is evident from Table 4, where a cut-off threshold at the 1st quartile reports a high performance for the objects that fit the trained model (0.9045 TPR at 0.0007 FPR), and a poor performance for those drifting away (0 TPR and 0 FPR); this means that at this threshold, CE's statistical metrics suggest to consider as untrustworthy only objects the classifier would have predicted incorrectly. Conversely, probabilities also tend to be skewed when predictions are wrong, affecting the ability to rely on such metrics to correctly identify concept drift. Table 4 shows 0.6654 TPR and 0 FPR for objects whose quality fall above the 1st quartile of the probability distribution, and 0.3176 TPR and 0.0013 FPR for those who fall below; this means that probabilities marked as unreliable also make predictions that would have been classified correctly.

As we move up towards more conservative thresholds, CE's statistical metrics start discarding objects that would have been classified correctly. This is unsurprising as we have defined a threshold that is more selective of the desired quality. Regardless, at each point p-values still outperform probabilities (higher TPR and FPR of objects with a quality higher than the cut-off, and lower for those below the threshold). These results further show how relying on detecting concept drift is a challenging problem that cannot be easily addressed by relying on a prefixed 50% threshold [19].

Note that the number of untrustworthy predictions on the testing dataset is a function of the number of drifting objects. If the entire dataset drifts, we would expect Transcend to flag as untrustworthy all (or most of) the predicted objects that do not fit the trained model.

Adapting to Concept Drift. Once drifting objects are identified, the next step would require data relabeling and model retraining, as outlined throughout the paper. Table 3c shows the results of these steps, which take precision for benign samples to 0.89 and recall for malicious ones to 0.76. We would like to remark that this work focuses on the construction of statistical metrics to *identify* concept drift as outlined so far. While relabeling is out of scope for this work, it is clear that an approach that identifies drifting objects is well-suited to address such a challenge in the pipeline as resources can be focused on analyzing samples that do not fit in the trained model.

4.2 Multiclass Classification Case Study

In this section we evaluate the algorithm proposed by Ahmadi et al. [1] as a solution to Kaggle's Microsoft Malware Classification Challenge; the underlying rationale is similar to that outlined in the previous section, thus, we only report insightful information and takeaways. In this evaluation, we train the classifier with seven out of eight available malware families; Trucur, the excluded family, represents our drifting testing dataset.

The confusion matrix reports a perfect diagonal[7]; in this case, the decision assessment gives us no additional information because we cannot analyze the distribution of p-values of incorrect choices. From a quality perspective, drawing upon the alpha assessment of Figure 4, two families, Vundo and Ramnit, have significant differences. The Ramnit family has p-values that are much higher than those of the interfering families. However, for Vundo the p-values of interfering families are closer to the correct ones. These details can be only be observed through the alpha assessment, suggesting that the identification of the Ramnit samples would be more robust when the data distribution changes.

Family Discovery. Below, we show how we identify a new family based on CE's statistical metrics.

The testing samples coming from Tracur are classified as follows: 5 as Lollipop, 6 as Kelihos_ver3, 358 as Vundo and 140 as Kelihos_ver1. Looking at the distribution of probabilities and p-values it is easy to relate to the case of binary classification, i.e., for each family there is only one class with high p-values corresponding to the class of the true label. For the test objects of Tracur, we observe that the p-values for all the classes are close to 0. This is a clear pattern which shows that the samples are coming from an unknown distribution. In a scenario changing gradually, we will observe an initial concept drift (as shown in the binary classification case study in § 4.1.1), characterized by a gradual decrease of the p-values for all the classes, which ends up in a situation where we have p-values very close to 0 as observed here. These results clearly show that even in multiclass classification settings, CE provides metrics that are better-suited

[7]We reached out to the authors who provided us with the dataset and the implementation of the learning algorithm to replicate the results presented in [1].

to identify concept drift than probabilities[8]. The comparison between p-values and probabilities is reported in Figures 7 to 10 in Appendix 7 and follow a reasoning similar to that of the binary classification case study.

5 Discussion

Security community has grappled with the challenge of concept drift for some time now [12, 23, 25]. The problem commonly manifests itself in most malware detection/classification algorithm tasks and models perform poorly as they become dated. Literature [12, 15, 16] recommends retraining the model periodically (see § 6) to get around this. However, retraining periodicity is loosely defined and is an expensive process that leads to sub-optimal results. Consequently, there are periods where the model performance cannot be trusted. The problem is further exacerbated as concept drift is hard to identify without manual intervention. If the model is retrained too frequently, there will be little novelty in information obtained through retraining to enrich the model. Regardless of the periodicity, the retraining process requires manual labeling of all the processed objects. Transcend selectively identifies the drifted objects with statistical significance[9], thus is able to restrict the manual labeling process to the objects that are substantially different than the ones in the trained model (see §3.3 and §4.1.1).

Adversarial ML and Model Fortification. Our work aims to detect concept drift as it occurs in an existing model. Concept drift can occur due to various reasons. Common causes being malware polymorphism or evasion but adversarial data manipulation (adversarial drift) can also be a reason. Approaches have been proposed to fortify models against drift [12, 15, 23], however such solutions deal with specific domains and do not provide a generalized solution. Transcend is agnostic to the machine learning algorithm under consideration. This let us leverage the strength of the algorithm while detecting concept drift. Therefore, if the algorithm is more resilient to concept drift, drift will be detected later on in time. If it is less resilient, drift will be detected as sooner.

Comparison with Probability. Probabilities have been known to work well in some scenarios but as demonstrated in § 4.1.1 and § 4.2 they are not as effective as compared to p-values which are more versatile, especially in the presence of concept drift. When probabilities are reported to be low it is difficult to understand if the sample does not belong to any class or if the sample is actually just *difficult* to classify while still belonging to one of the known classes. In other words, the p-value metric offers a natural *null option* when the p-values calculated for all the classes are low. Instead, as shown in the case of SVM (see, § 4.1.1), the probability metric is bounded to one of the options in the model.

[8]The algorithm in [1] relies on probabilities (decision trees).

[9]The p-value for an object o with label l is the statistical support of the null hypothesis H_0, i.e., that o belongs to l. Transcend finds the significance level (the per-class threshold) to reject H_0 for the alternative hypothesis H_a, i.e., that o does not belong l (p-values for wrong hypotheses are smaller than those for correct ones, e.g., Figure 2b).

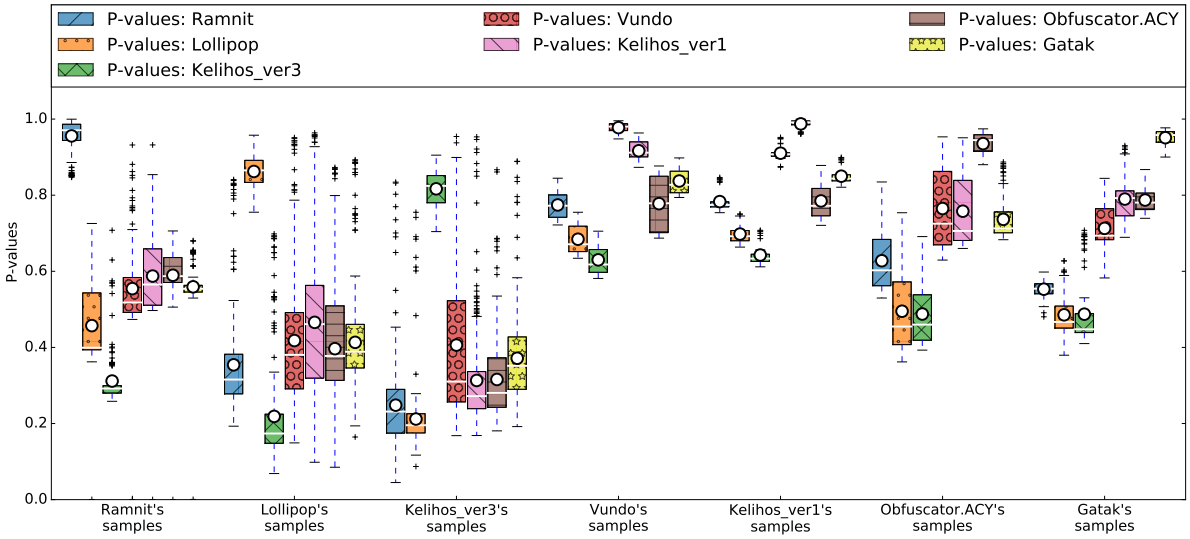


Figure 4: Multiclass Classification Case Study: Alpha assessment for the Microsoft classification challenge showing the quality of the decision taken by the algorithm. Although, perfect results are observed on the confusion matrix , the quality of those results vary a lot across different families.

It does not matter if the probabilities are well calibrated or not, the limitation is inherent to the metric. As discussed, the work by Rieck et al. [19] faces similar challenges when choosing the probability threshold. Moreover, the p-value metric provided by our framework, can be calculated from algorithms that do not provide probabilities, e.g., custom algorithms like [22], thus extending the range of algorithms that can benefit from a statistical evaluation.

Performance. Calculation of p-values is a computationally intensive process—for each sample z in a class $c \in C$, the calculation of a p-value requires computation of a non-conformity measure for every element in the dataset. This can be further exacerbated by non-conformity measures that rely on distances that are complex to compute. The computational complexity in relation to the number of the times that the non-conformity measure needs to be computed is $O(C \cdot N^2)$, where N represents the total number of samples and C represent the number of classes. Calculations can be sped up by computing a whole set of non-conformity scores in one single algorithm run. For example, SVM used in Drebin [2] can directly supply the total non-conformity scores for the calculation of one p-value in only one run of the algorithm, thus reducing the complexity to $O(C \cdot N)$. Further optimizations can be made for algorithms that treat each class separately; in such a scenario we can run the algorithm just for the class under analysis.

6 Related Work

Solutions to detect concept drift, specific to security domains, have been proposed [12, 15, 23], in contrast our framework provides a generic solution which is algorithm agnostic. On the other hand, solutions [6, 7] developed by the ML community have constrains that are not suitable for security applications (e.g., retrospective detection of concept drift when the classification decision has already been made).

Thomas et al. [23] present *Monarch* a real-time system that crawls URLs as they are submitted to web services and determines whether the URLs direct to spam. The system uses machine-learning to classify URLs as malicious or benign. The authors suggest training the model continuously to keep classification error low as the nature of malicious URLs keeps evolving. Kantchelian et al. [12] propose fusing human operators with the underlying machine-learning based security system to address concept drift in adversarial scenarios. Maggi et al. [15] present a machine-learning based system to classify malicious web applications. They use techniques specific to web application to detect concept drift and thus retrain their model to reduce false positives. Miconti et al. [16] show how models decay over time and propose ways to resist longer. Our model unifies these techniques as it generalizes to both the area of application and machine-learning algorithm used. The presented model can not only accurately predict when to retrain a model but also provides a quality estimate of the decisions made. These results can reduce human intervention and make it more meaningful thus decreasing the cost of operation. Transcend can be plugged on top of any such approach to provide a clear separation between non-drifting and drifting objects.

Deo et al. [5] propose using Venn-Abers predictors for assessing the quality of binary classification tasks and identifying concept drift. The Venn-Abers predictors offer automatically well calibrated and probabilistic guidance to detect change in distribution of underlying samples. Although useful, the approach has limitations and cannot draw concrete conclusions on sample clusters which are outliers. Also, Venn-Abers outputs multiple probabilities of which one is perfectly calibrated but it is not possible to know which. Our approach provides a simple mechanism to compare predictions through p-values and does not suffer from the discussed shortcomings. CE also works on multi-class prediction tasks, while this is not currently supported by Venn-Abers predictors.

Other works try to detect change point detection when the underlying distribution of data samples changes significantly, e.g., in case of evolving malware which is observed as a disruption in ex-changeability [25]. Martingales have often been used to detect drift of multidimensional data sequences using ex-changeability [8, 9]. Prior works [6, 7] use conformal prediction to detect deviation of the data sequence from independent and identically distributed (iid) assumption which could be caused by concept drift. The drift is measured by creating a martingale function. If the data is not iid, then the conformal predictor outputs an invalid result. Some p-values assigned to the true hypotheses about data labels are too small (or have another deviation from uniformity), and this leads to high values of the martingale. However, this martingale approach does not use p-values assigned to *wrong* hypotheses, which is another cause of wrong classification, e.g., malicious samples being classified as benign. We consider this information to be important because in the case of malware evolution, malicious samples are often specially designed to be indistinguishable from benign samples, therefore they tend to get high p-values assigned to wrong hypotheses. Additionally, the martingale approach uses true labels to study the drift of data without making any predictions, in contrast our approach does not have access to true labels and analyses the predictions made by a given model.

Comparison with Conformal Predictor. Although

conformal evaluator is built on top of conformal predictor (CP), it does not share the same weaknesses as that of other solutions based on it [6, 7]. Fern and Dietterich[10] also show that CP is not suited for anomaly detection as it outputs a set of labels and hence needs to be modified to predict quality of predictions. We further highlight the differences between CP and CE that makes CE better-suited to the concept drift detection task.

Conformal Predictor [24] (CP) is a machine learning classification algorithm. It relies on a non-conformity measure (NCM) to compute p-values in a way similar to CE. For each classification task, CP builds on such p-values to introduce credibility—the class, in a classification problem, with the highest p-value and confidence—defined as one minus the class with the second highest p-value (these metrics are different from CE metrics, see § 2.4). The CP algorithm then outputs either a single class prediction with the identified credibility and confidence, or, given a fixed confidence level $1-\varepsilon$ (where ε represents the significance level), a prediction set that includes classes that are above it. This set is proven to cover the true class with probability not lower than $1-\varepsilon$.

CE dissects CP metrics and to extract its p-values calculation. The p-values are used together with the output labels provided by the algorithm under evaluation, to build CE metrics. CP ignores these labels as it tries to predict them. Conversely, CE uses this information to provide quality metrics to assess the quality of the encapsulated algorithm. This change is of paramount importance to derive the thresholds (computed by Transcend) used to accept or reject a prediction.

The *posterior* use of the labels is a key feature that enables CE to detect concept drift. On the contrary, CP is designed as a predictive tool making only use of prior information. Since labels are important pieces of information, CE uses them to build its metrics and assessments (see, § 2.4 and § 3). The labels used by CE are the ones of the training samples and not the labels of the testing samples that are unavailable at the time of classification.

7 Conclusions

We presented Transcend—a fully tunable tool for statistically assessing the performance of a classifier and filtering out unreliable classification decisions. At the heart of Transcend, CE's statistical confidence provides evidence for better understanding model generalization and class separation; for instance, CE has been successfully adopted to selectively invoke computationally expensive learning-based algorithms when predictions choose classes with low confidence [4], trading off performance for accuracy. Our work details the CE metrics used in [4] and extend it to facilitate the identification of concept drift, thus bridging a fundamental research gap when dealing with evolving malicious software.

We present two case studies as representative use cases of Transcend. Our approach provides sound results for both binary and multi-class classification scenarios on different datasets and algorithms using proper training, calibration and validation, and testing datasets. The diversity of case studies presents compelling evidence in favor of our framework being generalizable.

[10] A. Fern and T. Dietterich. "Toward Explainable Uncertainty". https://intelligence.org/files/csrbai/fern-slides-1.pdf

Availability

We encourage the adoption of Transcend in machine learning-based security research and deployments; further information is available at:

https://s2lab.isg.rhul.ac.uk/projects/ce

Acknowledgments

This research has been partially supported by the UK EPSRC grants EP/K033344/1, EP/L022710/1 and EP/K006266/1. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. We are equally thankful to the anonymous reviewers' comments and Roberto Perdisci, our shepherd, for their invaluable comments and suggestions to improve the paper. Also, we thanks Technology Integrated Health Management (TIHM) project awarded to the School of Mathematics and Information Security at Royal Holloway as part of an initiative by NHS England supported by Innovate UK. We also thank the authors of [2], for their public dataset used in our evaluation, and Mansour Ahmadi for providing us the algorithm used in [1].

References

[1] AHMADI, M., ULYANOV, D., SEMENOV, S., TROFIMOV, M., AND GIACINTO, G. Novel feature extraction, selection and fusion for effective malware family classification. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy* (New York, NY, USA, 2016), CODASPY '16, ACM, pp. 183–194.

[2] ARP, D., SPREITZENBARTH, M., HUBNER, M., GASCON, H., AND RIECK, K. DREBIN: effective and explainable detection of android malware in your pocket. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014* (2014).

[3] BREIMAN, L. Random Forests. *Machine Learning 45*, 1 (2001), 5–32.

[4] DASH, S. K., SUAREZ-TANGIL, G., KHAN, S. J., TAM, K., AHMADI, M., KINDER, J., AND CAVALLARO, L. Droidscribe:

Classifying android malware based on runtime behavior. In *2016 IEEE Security and Privacy Workshops, SP Workshops 2016, San Jose, CA, USA, May 22-26, 2016* (2016), pp. 252–261.

[5] DEO, A., DASH, S. K., SUAREZ-TANGIL, G., VOVK, V., AND CAVALLARO, L. Prescience: Probabilistic guidance on the retraining conundrum for malware detection. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security* (New York, NY, USA, 2016), AISec '16, ACM, pp. 71–82.

[6] FEDOROVA, V., GAMMERMAN, A. J., NOURETDINOV, I., AND VOVK, V. Plug-in martingales for testing exchangeability online. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012* (2012).

[7] HO, S. A martingale framework for concept change detection in time-varying data streams. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005* (2005), pp. 321–327.

[8] HO, S., AND WECHSLER, H. Query by transduction. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 9 (2008), 1557–1571.

[9] HO, S., AND WECHSLER, H. A martingale framework for detecting changes in data streams by testing exchangeability. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 12 (2010), 2113–2127.

[10] HUBERT, M., AND VANDERVIEREN, E. An adjusted boxplot for skewed distributions. *Computational Statistics and Data Analysis 52*, 12 (2008), 5186 – 5201.

[11] KAGGLE INC. Microsoft Malware Classification Challenge (BIG 2015). `https://www.kaggle.com/c/malware-classification`, 2015.

[12] KANTCHELIAN, A., AFROZ, S., HUANG, L., ISLAM, A. C., MILLER, B., TSCHANTZ, M. C., GREENSTADT, R., JOSEPH, A. D., AND TYGAR, J. D. Approaches to adversarial drift. In *AISec'13, Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, Co-located with CCS 2013, Berlin, Germany, November 4, 2013* (2013), pp. 99–110.

[13] LI, P., LIU, L., GAO, D., AND REITER, M. K. On challenges in evaluating malware clustering. In *Recent Advances in Intrusion Detection, 13th International Symposium, RAID 2010, Ottawa, Ontario, Canada, September 15-17, 2010. Proceedings* (2010), pp. 238–255.

[14] LINDORFER, M., NEUGSCHWANDTNER, M., AND PLATZER, C. MARVIN: efficient and comprehensive mobile app classification through static and dynamic analysis. In *39th IEEE Annual Computer Software and Applications Conference, COMPSAC 2015, Taichung, Taiwan, July 1-5, 2015. Volume 2* (2015), pp. 422–433.

[15] MAGGI, F., ROBERTSON, W. K., KRÜGEL, C., AND VIGNA, G. Protecting a moving target: Addressing web application concept drift. In *Recent Advances in Intrusion Detection, 12th International Symposium, RAID 2009, Saint-Malo, France, September 23-25, 2009. Proceedings* (2009), pp. 21–40.

[16] MARICONTI, E., ONWUZURIKE, L., ANDRIOTIS, P., DE CRISTOFARO, E., ROSS, G., AND STRINGHINI, G. Mamadroid: Detecting android malware by building markov chains of behavioral models. *arXiv preprint arXiv:1612.04433* (2016).

[17] PLATT, J., ET AL. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers 10*, 3 (1999), 61–74.

[18] RIDGEWAY, G. The state of boosting. *Computing Science and Statistics* (1999), 172–181.

[19] RIECK, K., HOLZ, T., WILLEMS, C., DÜSSEL, P., AND LASKOV, P. Learning and classification of malware behavior. In *Detection of Intrusions and Malware, and Vulnerability Assessment, 5th International Conference, DIMVA 2008, Paris, France, July 10-11, 2008. Proceedings* (2008), pp. 108–125.

[20] SHAFER, G., AND VOVK, V. A tutorial on conformal prediction. *The Journal of Machine Learning Research 9* (2008), 371–421.

[21] TANG, Y. extreme gradient boosting. `https://github.com/dmlc/xgboost`.

[22] TEGELER, F., FU, X., VIGNA, G., AND KRUEGEL, C. Botfinder: Finding bots in network traffic without deep packet inspection. In *In Proc. Co-NEXT 12* (2012), pp. 349–360.

[23] THOMAS, K., GRIER, C., MA, J., PAXSON, V., AND SONG, D. Design and evaluation of a real-time URL spam filtering service. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA* (2011), pp. 447–462.

[24] V. VOVK, A. G., AND SHAFER, G. *Algorithmic learning in a random world*. Springer-Verlag New York, Inc., 2005.

[25] WECHSLER, H. Cyberspace security using adversarial learning and conformal prediction. *Intelligent Information Management 7*, 04 (2015), 195.

Appendix A.

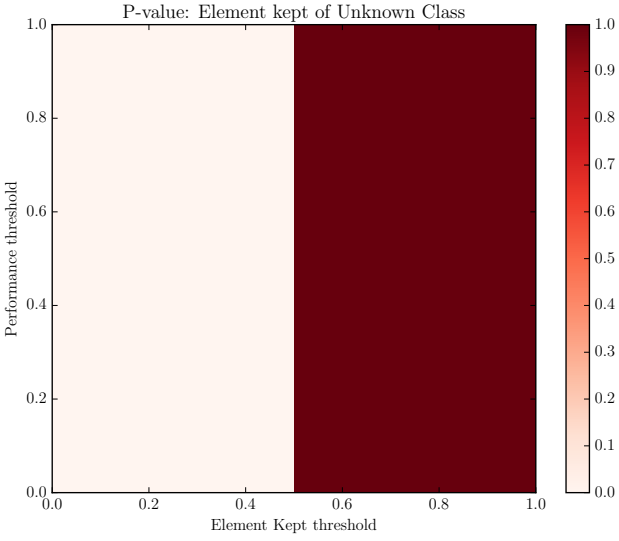
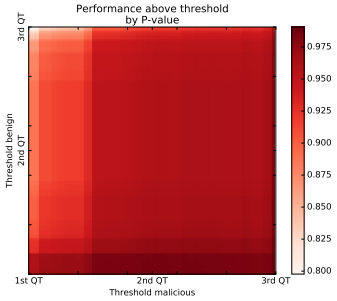
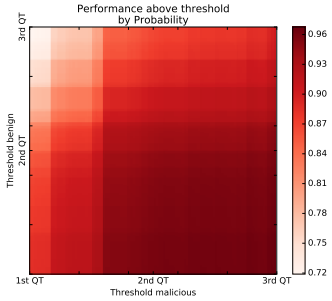


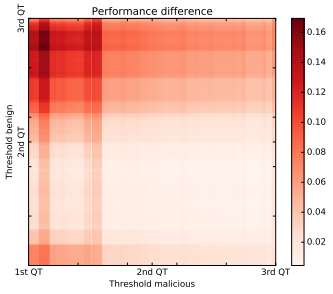
Figure 5: Multiclass Classification Case Study: Element kept during the test of the new class. The test elements belong to a new class so every samples kept will be missclassified. The net separation between good and bed performance comes from the perfect classification of training samples used to derived the thresholds.



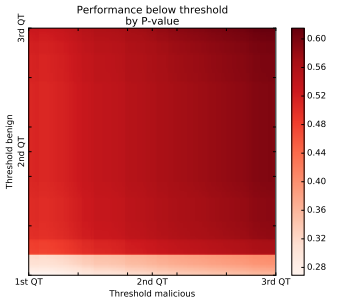
(a) Performance of p-value driven threshold for element above the threshold.



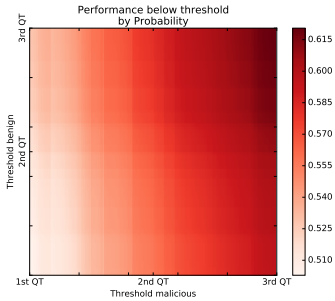
(b) Performance of probability driven threshold for element above the threshold.



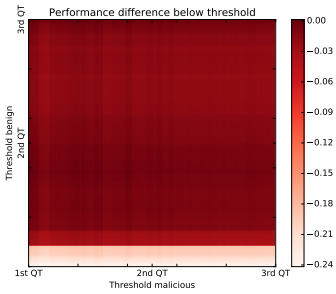
(c) Performance difference between p-value and probability for element above the threshold.



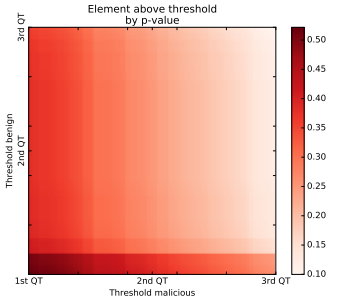
(d) Performance of p-value driven threshold for element below the threshold.



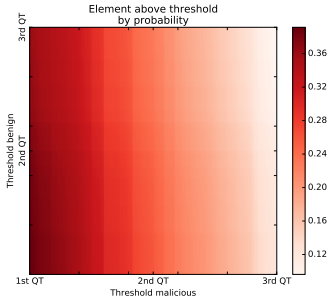
(e) Performance of probability driven threshold for element below the threshold.



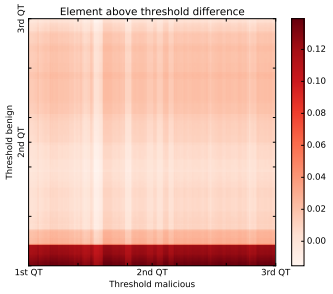
(f) Performance difference between p-value and probability for element below the threshold.



(g) Number of element above the p-value threshold.



(h) Number of element above the probability threshold.



(i) Difference between the number of element above the threshold between p-value and probability.

Figure 6: Binary Classification Case Study [2]: complete comparison between p-value and probability metrics. Across all the threshold range we can see that the p-value based thresholding is providing better performance than the probability one, discarding the samples that would have been incorrectly classified if kept.

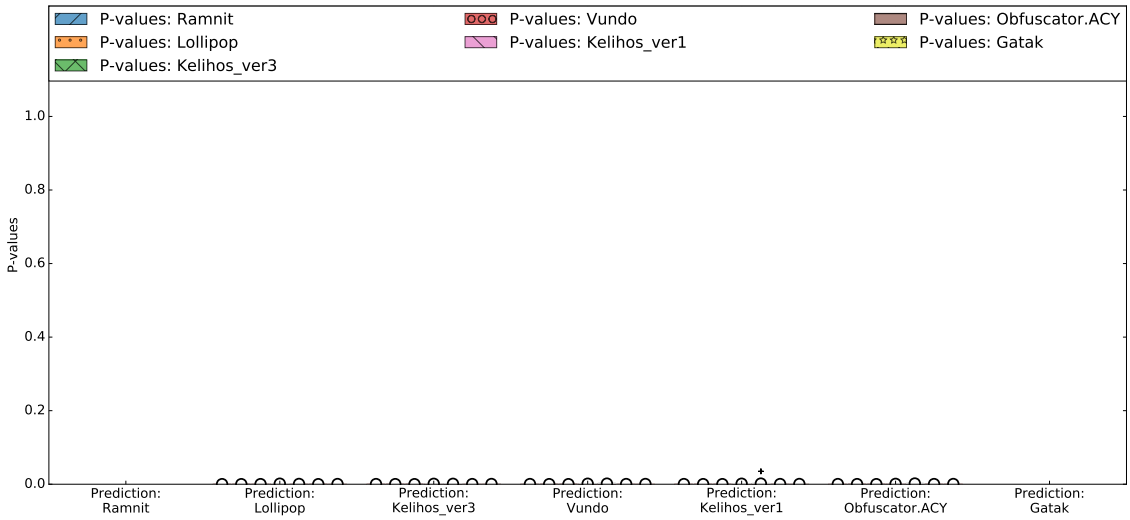


Figure 7: Multiclass Classification Case Study [1]: P-value distribution for samples of Tracur family omitted from the training dataset; as expected, the values are all close to zero.

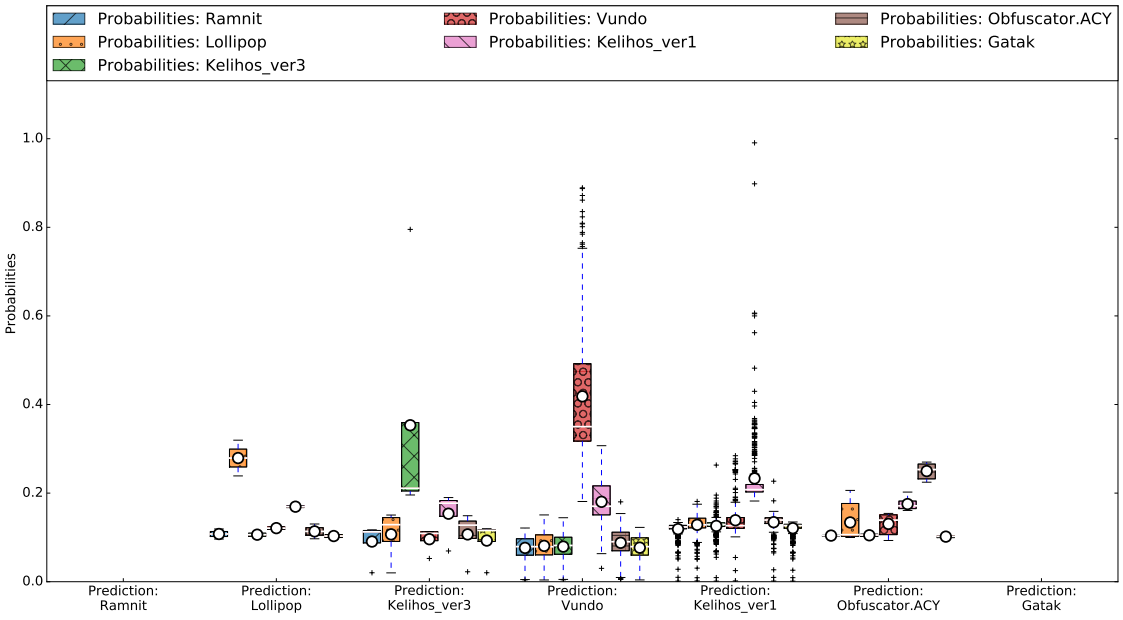


Figure 8: Multiclass Classification Case Study [1]: probability distribution for samples of Tracur family omitted from the training dataset. Probabilities are higher then zero and not equally distributed across all the families, making the classification difficult. It is worth noting some probabilities are skewed towards large values (i.e., greater than 0.5) further hindering a correct classification result.

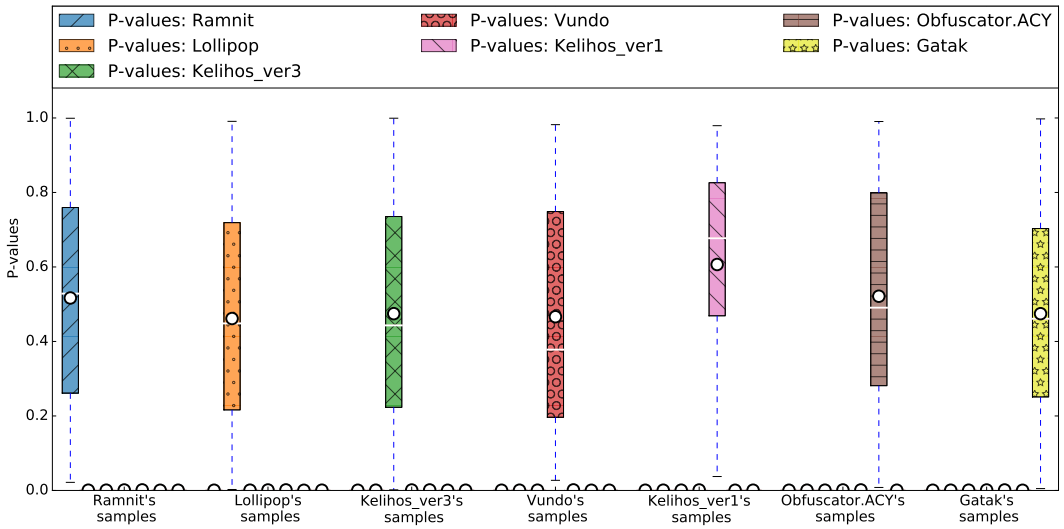


Figure 9: Multiclass Classification Case Study [1]: a new family is discovered by relying on the p-value distribution for known malware families. The figure shows the amount of conformity each sample has with its own family; for each sample, there is only one family with high p-value.

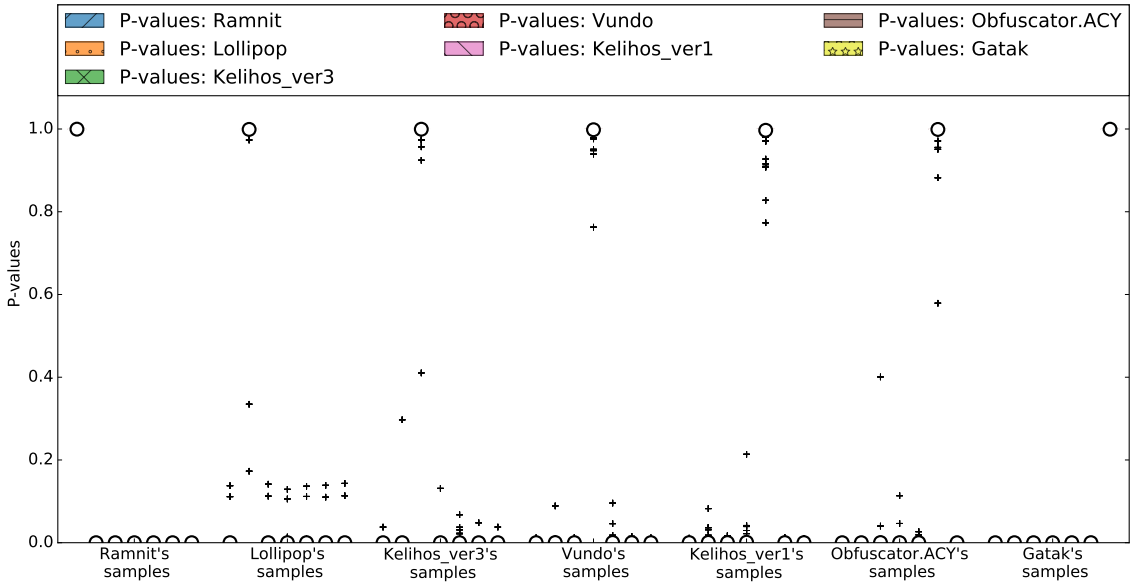


Figure 10: Multiclass Classification Case Study [1]: probability distribution for samples of families included in the training dataset. High probabilities support the algorithm classification choice.