S. Krajči (ed.): ITAT 2018 Proceedings, pp. 52–58
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Martin Kopp, Lukáš Bajer, Marek Jílek, and Martin Holeňa

Comparing rule mining approaches for classification with reasoning

Martin Kopp[1,2], Lukáš Bajer[2], Marek Jílek[1], and Martin Holeňa[1,3]

[1] Faculty of Information Technology, Czech Technical University in Prague
Thákurova 9, 160 00 Prague
[2] Cisco Systems, Cognitive Research Team in Prague
[3] Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Prague

Abstract: Classification serves an important role in domains such as network security or health care. Although these domains require understanding of the classifier's decision, there are only a few classification methods trying to justify or explain their results.

Classification rules and decision trees are generally considered comprehensible. Therefore, this study compares the classification performance and comprehensibility of a random forest classifier with classification rules extracted by Frequent Item Set Mining, Logical Item Set Mining and by the Explainer algorithm, which was previously proposed by the authors.

Keywords: Classification; Comprehensibility; Random Forest; Rule Mining

1 Introduction

Classification is one of the main directions of machine learning research. In the past decades, researchers developed classification models which, when learnt properly, can beat humans in tasks we believed they would never succeed, like handwritten text recognition [**?**, **?**], or even tasks that were specifically designed to be unsolvable for computers, like CAPTCHAs [**?**] and other human interaction proofs [**?**].

Current state-of-the-art classifiers excel in predictive performance, but they lack other quality characteristics for human decision process: deep neural networks, for instance, do not provide explanation of their decisions. Yet, the reasoning justifying decisions is critical in many application domains such as medicine or network security. In medicine, physicians would rather believe a less precise model, if the classification is justified by explanation they can understand. In the network security domain, analyst are often overwhelmed by alarms. But investigation without providing context or explanation, why the alarm was raised, takes a substantial amount of time. Current ransomware affairs [**?**] have shown that time is critical when infection occurs.

This work is focused on comparison of context provided by a random forest classifier and three different rule mining approaches. Random forests were chosen as a representative of the state-of-the-art classifiers which is also able to provide a reasonable justification of its decisions. The precision of classifiers based on sets of rules highly depends on the process of mining rules from data. We tested the quality of rules mined by Frequent Item Set Mining [**?**], Logical Item Set Mining [**?**] and extracted by the Explainer algorithm [**?**]. The comparison focuses on the precision and recall over time and also the quality of presented context. The experiments were done on a dataset from the network security domain.

The rest of the work is organized as follows. The next section covers the related work in the field of comprehensible classification. Section **??** briefly introduces all used rule mining approaches that are experimentally evaluated in Section **??**, followed by the conclusion and future plans.

2 Related work

The pressing issue of comprehensible predictive models has been intensively studied in the field of anomaly detection. The first work considering anomaly explanation was [**?**]. It defined an explanation as "provision of a description or an explanation of why an identified anomalous sample is exceptional". The proposed method first detected all distance-based anomalies in the whole attribute space. Then, it identified the smallest subspace in which the anomaly could be still detected and used that subspace as an explanation.

In the approach presented in [**?**], artificial samples are generated in the vicinity of each sample x. Then a classifier is trained to separate the artificial samples from real samples. If x was an anomaly, then artificial samples should be separated easily, which would result in the classifier having low error and vice-versa.

The method proposed in [**?**] derives the anomaly score from the frequency of histogram bins, from which the method also extracts context and explanation of the anomaly.

Authors of [**?**] used the probabilistic RBF kernels to extract and compare feature impact (positive and negative) for each class in multi-class classification problems.

Most of the recent prior art stops the explanation after identifying the set of features by which the sample under investigation can be differentiated from the rest. The Explainer [**?**] describes the sample by a set of association rules.

A comparison by means of comprehensibility of the well established classifiers such as decision trees, nearest neighbours or Bayesian networks was done in [**?**].

The authors of [**?**] compared multiple rule mining approaches, their general similarities and differences and also their computational efficacy. This paper is focused on the comprehensibility and complexity of a context provided to the domain experts by different rule mining approaches.

The most advanced approach we are aware of is [**?**]. It introduced a framework for measuring not only a comprehensibility (if a prediction is presented in a way that it is easy to understand) but also a justifiability (if a model is in line with the domain knowledge). We would like to use this framework in our future work.

3 Rule mining approaches

Association rules are defined as an implication of the form $\mathcal{A} \Rightarrow \mathcal{C}$, where $\mathcal{A},\mathcal{C} \subseteq \mathbf{I}$ and $\mathbf{I} = \{i_1, i_2, ..., i_d\}$ is a set of binary attributes called items. Association rules are typically mined from a database $\mathbf{X} = \{x_1, x_2, ..., x_n\}$. Each line $x \in \mathbf{X}$ represents a set of items $x \subseteq \mathbf{I}$ and is called a transaction. In this paper, we work with a specific type of association rules, often called classification rules, that are also defined as $\mathcal{A} \Rightarrow \mathcal{C}$. Here, $\mathcal{A} \subseteq \mathbf{I}$ and $\mathcal{C} \in \mathbf{C} = \{c_1, c_2, ..., c_k\}$ is a single item representing a particular class.

The rest of this section surveys rule mining approaches used in this paper.

3.1 Frequent item set mining

The Frequent Item Set Mining (FISM) is a general framework for discovering groups of items (item sets) that are often seen together. The first algorithm mining association rules called GUHA was published by Petr Hájek in [**?**]. In fact, it was a mathematical method for automatic search of hypotheses valid in given data, based on generalized quantifiers of Boolean predicate logic, and the quantifier corresponding to association rules was called *founded almost implication*. The very same approach was rediscovered for data mining purposes as the Apriori algorithm by Agrawal almost thirty years later [**?**].

Since then, a variety of improvements for the original Apriori algorithm was proposed [**?**, **?**], and also several new algorithms for frequent items mining were invented, e.g., FP-Growth [**?**], LCM [**?**] or Eclat [**?**]. This paper uses the FP-Growth algorithm as the most time and memory efficient representative of the FISM algorithms.

FP-Growth algorithm starts by building a specific prefix tree called **frequent pattern tree (FP-tree)**. First, the frequencies of all items $i \in \mathbf{I}$ are calculated. Then, all items i with frequency lower than the user specified threshold $\theta_{minFreq}$ are filtered out from all transactions $x \in \mathbf{X}$. Items remaining in the filtered transactions are sorted in descending order according to their frequency. The prefix tree is built by inserting the filtered and sorted transactions.

Once the FP tree is built, it is recursively traversed in a bottom-up manner, mining frequent item sets laying on the path from leaf to root. Based on the FP-tree construction process, each transaction is mapped to a path in the FP-tree. The FP-tree structure also guarantees that all frequent item sets present in the database $\mathbf{X}$ can be found on the path from some leaf to the root. Moreover, one path in the FP-tree may represent frequent item sets in multiple transactions.[1] This property also ensures the memory efficiency of FP-Growth.

3.2 Logical item set mining

The Logical Item Set Mining (LISM) [**?**] is an alternative approach to mining association rules from data. The key difference from FISM is that LISM captures logical relations not only between frequent items, but it also extracts strong relations between rarely occurring items. By leveraging indirect relationships between items, it can also discover relations between item sets that are not present in a dataset. The algorithm has counting, consistency, denoising and discovery phases.

During the **counting phase**, co-occurrence counts, marginal counts and total counts are calculated.

Co-occurrence count $\psi(i_a, i_b)$ for every pair of items $(i_a, i_b) \in \mathbf{I} \times \mathbf{I}$, where $i_a \neq i_b$, is defined as the number of transactions in which both items co-occurred:

$$\psi(i_a, i_b) = \sum_{j=1}^{n} \delta(i_a \in x_j)\delta(i_b \in x_j), \tag{1}$$

where $\delta(condition)$ is an indicator function which is 1 if the *condition* is true and 0 otherwise. The results are stored in the symmetrical matrix $\Phi = [\phi(i_a, i_b)]$, which is usually very sparse.

Marginal count $\psi(i_a)$ is defined as the number of pairs in which the item $i_a \in \mathbf{I}$ occurred with some other item:

$$\psi(i_a) = \sum_{i_b \in \mathbf{I}, i_a \neq i_b} \psi(i_a, i_b). \tag{2}$$

Total count ψ_0 is defined as the total number of pairs in which some item co-occurred with some other item:

$$\psi_0 = \frac{1}{2}\sum_{i_a \in \mathbf{I}} \psi(i_a) = \frac{1}{2}\sum_{i_a \in \mathbf{I}}\sum_{i_b \in \mathbf{I}} \psi(i_a, i_b) \tag{3}$$

These three results are then used as estimates of the co-occurrence and marginal probabilities

$$P(i_a, i_b) = \frac{\psi(i_a, i_b)}{\psi_0}, \quad P(i_a) = \frac{\psi(i_a)}{\psi_0}. \tag{4}$$

The **consistency phase** reduces the effect of noise and amplifies the importance of rare items that are consistently

[1] Items in item sets are ordered in the descending order, frequent items are arranged closer to the top of the FP-tree and more likely to be shared.

seen together. A variety of distance measures can be employed, e.g., cosine, Jaccard or point-wise mutual information. The cosine similarity defined as

$$\phi(i_a, i_b) = \frac{P(i_a, i_b)}{\sqrt{P(i_a)P(i_b)}} \in [0,1] \tag{5}$$

was used in our experiments.

The iterative **denoising phase** uses the co-occurrence consistencies obtained in the previous iteration to remove noisy co-occurrence counts in Φ. Then, marginal and total counts are recomputed solely from the updated Φ, there is no need to touch data again. As a last step of every iteration a consistency counts are updated.

In the **discovery phase**, a graph is created from the denoised co-occurrence consistency matrix Φ. Given Φ, a logical item set is defined as a set of items where each item has a high co-occurrence consistency with all other items in the set. Such sets are found by application of an algorithm for finding maximal cliques, e.g., [**?**], on the co-occurrence consistency graph.

3.3 Explainer

The Explainer [**?**] is a tree based algorithm designed to explain why a sample $x^a \in \mathbf{X}$ is an anomaly with respect to the rest of the data in $\mathbf{X}$. The output can be a set of the most important features or a set of association rules. Properties of extracted rules were studied in [**?**].

To explain an anomaly x^a, the Explainer trains a set of specific decision trees to separate x^a from rest of the data in $\mathbf{X}$. Rules are extracted from each of those trees and assembled into a set of association rules. The key steps are summarized in Algorithm **??**.

Algorithm 1 Summary of the Explainer algorithm for a single anomaly x^a.

Input:
data – input dataset; x^a – anomalous sample; *size* – training set size; n_T – the number of trees to be trained.
Output:
rules – rules explaining x^a

```
Forest ← ∅
for i ← 1 ... n_T do
    T ← createTrainingSet(data, size, x^a)
    t ← trainTree(T)
    Forest ← Forest ∪ t
end for
rules ← extractRules(Forest)
```

During the **training set selection**, a dataset $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ is split into two disjoint sets $\mathbf{X}^a$, containing anomalous samples, and $\mathbf{X}^n$, containing normal samples. Then, a training set $\mathbf{T}$ contains the anomaly x^a as one class and a subset of $\mathbf{X}^n$ as the other. The simplest strategy is to select k samples randomly from $\mathbf{X}^n$ with uniform probability. This approach is computationally effective and was proven to work well when compared to more sophisticated approaches [**?**].

Training a tree is very similar to standard random forests [**?**]. A candidate node is found and the optimal splitting function is applied on that node. This greedy procedure repeats until the specified stopping criteria are met.

The node that contain x^a is always the one being split in the Explainer algorithm. The standard procedure to find the splitting function h is maximizing the information gain over the space of all possible splitting functions $\mathcal{H}$. But as there is only a single point x^a in the anomaly class, the information gain is equivalent to minimizing the size of the node containing x^a:

$$\arg\min_{h \in \mathcal{H}} |\mathbf{S}^a(h)|, \tag{6}$$

where $\mathbf{S}^a \subset \mathbf{T}$ denotes the subset of the training set containing the anomaly x^a after the split. The training is stopped when all leaves are pure (contain samples from a single class).

Once a tree is trained, it is used for **rule extraction**. Let $h_{f_1,\theta_1}, \ldots, h_{f_d,\theta_d}$ be the set of splitting functions, with features $f_1, \ldots, f_d$ and threshold $\theta_1, \ldots, \theta_d$, used in inner nodes on a path from the root to the leaf with x^a. Then x^a is explained as a conjunction of atomic conditions:

$$(x_{f_1} > \theta_1) \wedge (x_{f_2} > \theta_2) \wedge \ldots \wedge (x_{f_d} > \theta_d), \tag{7}$$

which is the output of the algorithm. This conjunction can be read as "the sample is anomalous because it is greater than threshold θ_1 in feature f_1 and greater than θ_2 in feature f_2 and ... than majority of samples". Each tree provides one such conjunction, that are then aggregated.

4 Experiments

This section experimentally compares the classification performance of the three described rule mining approaches with the random forest classifier. The section starts with a description of the dataset used in our research and with the setting of the performed experiments. The comparison is then based on the precision and recall measures, and the final part concludes with a thorough discussion of differences of explanations provided by each approach.

4.1 Dataset description

The dataset in this research consists of 8 consecutive weeks of network traffic collected by the Cognitive Threat Analytics (CTA) intrusion detection system. It contains about 9 million samples, where each sample is a collection of all network events observed on a particular network host within the 24-hours window. In the words used in the market basket analysis, each sample x is a transaction containing a subset of all events/items $x \subseteq \mathbf{I}$.

CTA distinguishes about 300 events falling into four categories:

- **signature based (SB)** – produced by matching behavioural signatures handmade by a domain expert.
- **classifier based (CB)** – created by supervised classifiers trained on historical data.
- **anomaly based (AB)** – created by the anomaly detection engine which consists of 70 individual detectors (statistical, volumetric, proximity, targeted, domain specific).
- **contextual events (CE)** – describe various network behaviours to provide an additional context, e.g., file download, direct access on raw IP, software update.

These events together serve as high level behavioural indicators that can be used to create a behavioural profile of a malware family. This behavioural profile can be used to identify malware infections in their early stages and stop them before they do any harm.

The database was labelled by the CTA engine in a way that transactions of a network hosts infected by either banking trojan, click fraud, information stealer or malware distribution were marked as positives/malicious (4801 and 6463 transactions in training and testing sets respectively) and the other transactions were labelled negative/benign (3.75 mil. and 5.23 mil. transactions respectively).

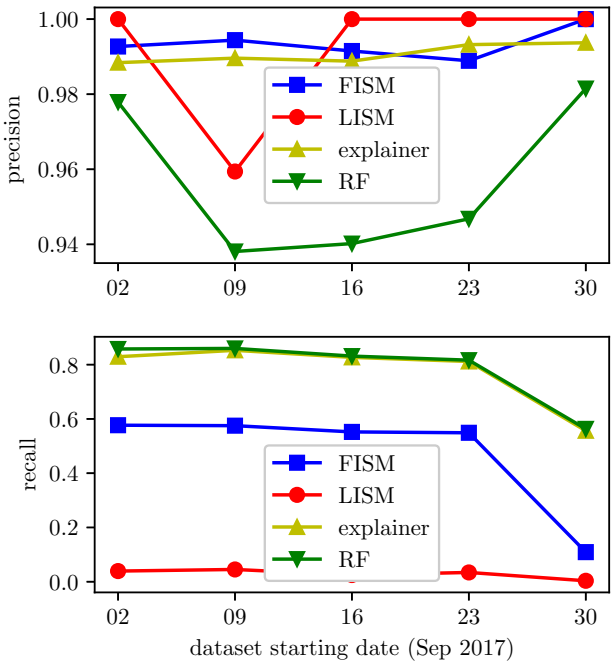

Figure 1: Precision and recall of the four considered classifiers measured in five consecutive weeks in September 2017.

4.2 Experimental setup

All following experiments used 3 weeks of data, approximately 3.75 million of transaction, for training/rule mining and 5 weeks, 5.23 million of transactions, for testing.

Parameters of the random forest were set as follows: number of trees = 19; maximal depth of a tree = 25; number of features per split = 100.

The Explainer was set to train 10 trees per positive sample while selecting a random training set of size 1000 samples.

The FP-growth algorithm was set to produce only rules with support higher than 3 transactions.

The parameters of LISM was: similarity measure = cosine (**??**); $\theta_{cooc} = 1 \cdot 10^{-7}$; $\theta_{cons} = 2$. Cliques discovered using this setting were split into 3–5 items long rules. Details about the optimization of parameters can be found in [**?**].

As a last step, all rules were filtered to have precision on the training set at least 80%. The filtered rules were then used in the following experiments.

4.3 Classification efficacy

The described rule filtering, based on the minimal precision 80% on the training set, resulted in very few false positive predictions: all classifiers reached more than 90%-precision on all the testing sets, as depicted in the upper graph in Figure **??**. While the rules mined by the Explainer and FISM algorithm provide stable results with precision reaching roughly 99% during all five testing weeks, the rules generated by the LISM and random forest exhibit greater variance in time.

The ability to discover the malicious content in the testing data, as measured by the classifiers' recall, diversifies considerably more. Both Explainer's and random forest's sets of rules were able to identify more than 80% of the malicious samples in the testing sets (except the last week). The graph in Figure **??**, on the other hand, clearly shows that the highest precision rate of the LISM-generated rules is accompanied with the lowest degree of recall.

The LISM is able to generate complex and very descriptive rules that, however, can be hardly located in the data. That results in only 34 generated rules reaching the training precision threshold 80%, which is probably the cause behind the lowest recall. The 100%-precision in 4 out of 5 testing weeks is surprisingly good result, though.

The performance of the FISM and Explainer differs mainly in their recall, where the better results of the Explainer is probably caused by the algorithm's focus to the shortest and strongest rules; only 14 rules reached the 80% precision threshold. The FISM, on the other hand, is able to generate longer rules, which are harder to match.

Random forest classifier with its up to 25-levels deep decision trees is able to identify the highest number of the testing malicious transactions. On the contrary, the high complexity of the trees, at the same time, causes the lowest observed precision out of four compared classifiers.

4.4 Context comparison

This section presents the comparison of context provided by the random forest classifier, Explainer, Logical Item Set Mining and Frequent Item Set Mining. Examples of mined rules are presented and discussed in detail.

Random Forest

In general, random forests can provide two types of context; feature importance and classification rules.

The feature importance provides less information than rules. It represents aggregated global knowledge for all classes in the training set. It can be extracted per class if a random forest is trained for each class separately. Our database represents a dichotomy problem with both classes being in fact mixtures of multiple sub-classes that we are unable to separate.

Classification rules provide more specific/local information. Unfortunately, trees in a random forest are typically trained deep to be as precise as possible. In other words, the path from a root to a leaf will be long and therefore, the extracted rule will be also long. The other drawback of training random forests directly on transactions is so called *negative evidence*. In the network security domain, the negative evidence is when a network host is being marked as infected because of some behaviour it doesn't have, e.g. "a host is infected, because it didn't downloaded an image".

The real example of a rule with multiple negative evidences extracted directly from the random forest used in the experiments is: "If you are not infected by a click fraud and you are not infected by an information stealer and you are not infected by the Sality trojan and you are not infected by the malware called Gamarue and you don't have encrypted connection then you are infected." As you can see from this simple yet real example, rules extracted from a random forest may be more puzzling than explaining.

Explainer

The Explainer can be easily set to provide rules with only *positive evidences*. The trouble is, that it was designed to extract the smallest set of the shortest possible rules. The extracted rules contained the real causes of incidents but not any additional context which would simplify the investigation. From the rules created by the Explainer, 14 had a precision higher than 80%. They typically contained only one event produced by a supervised classifier. The second event was presents in only 3 cases and there was no rule longer than that. Selected examples of longer rules follow:

1. AB:ShadowUser CB:ClickFraud
2. CB:SuspAdvertising CB:MaliciousAdvertising

The first example contains an anomaly based event AB:ShadowUser and a classifier based event CB:ClickFraud. AB:ShadowUser identifies network hosts that are visiting a high number of network domains which nobody else visit. The CB:ClickFraud event is created by the random forest classifier trained to discover malware from the click fraud family. The click fraud malware family is known for visiting a lot of weird pages without user's knowledge and is often used for clicking on web advertisements to generate money.

The second rule contains two classifier based events indicating a host visited a lot of suspicious advertisement pages(CB:SuspAdvertising), some of them probably malicious (CB:MaliciousAdvertising). Here, the malware started by showing additional unwanted advertisements, banners and pop-ups and ended by ex-filtrating sensitive data.

LISM

Rules extracted by the Logical Item Set Mining create a very logical and justifiable connections between items. All items in a rule are always very strong indicators of a particular threat. Unfortunately, they would appear all at once very rarely, but if they do, it is for sure a serious infection. LISM created 34 rules with precision over 80%. The length of the rules is ranging from 3 to 5. Examples follow:

1. SB:Blocked CB:MalBinary SB:Sality
2. CB:ClickFraud CB:Malwartising CB:MalwareDistr

The first example shows a download of malicious binary (CB:MalBinary) from a black listed domain (SB:Blocked). Furthermore, it has a well known signature of a malware called Sality (SB:Sality). Each of these events is strong enough evidence to trigger an immediate re-mediation alert on its own.

The second rule shows a nice example of a malicious escalation. At first, a host was infected by a click fraud malware (CB:ClickFraud), which was visiting a shady advertising sites (CB:MalAdvertising). Then, the host started to download and distribute additional malicious modules (CB:MalwareDistr).

FISM

The Frequent Item Set Mining provided the best explanation/context from all compared method. Rules contain all important indicators while providing a reasonable additional context. FISM created 454 rules, with length ranging from 3 to 5, that satisfied 80% threshold on precision, examples follow:

1. CE:jQuery AB:SuspDomain CB:ClickFraud
2. AB:PathCount AB:ShadowUser CB:ClickFraud CE:WPManagment

The first example shows a host that downloaded a modified javascript library jQuery(CE:jQuery) from a suspicious network domain (AB:SuspDomain). This modified library was the source of infection, which was later detected by the random forest classifier trained for detection of a click fraud malware family (CB:ClickFraud).

The second rule shows a sophisticated example of click fraud capabilities (CB:ClickFraud). In that case, the malware had a password cracker module installed and was used to crack into administration sections of web pages created using Wordpress(CE:WPManagment). Again, we can see a host visiting large amount of low probability network domains (AB:ShadowUser), WordPress blogs in that case, where on most of these pages was visited the outlying number of paths (AB:PathCount), specifically only one.[2]

5 Conclusion

This paper compared the classification performance and comprehensibility of a random forest classifier with classification rules extracted by the Frequent Item Set Mining, Logical Item Set Mining and by the Explainer algorithm, which was previously proposed by the authors. All the algorithms showed surprisingly similar precision with rules extracted by LISM performing slightly better, with exception on one week. From the recall point of view the best performing algorithm was the random forest followed by rules extracted by the Explainer.

The comparison of provided explanations revealed that rules extracted from random forests trained on transaction data can be more puzzling than comprehensible. The relationships between items mined by LISM were not only comprehensible but also justifiable (in line with the domain knowledge). Unfortunately, they are very rarely seen within the real data. Rules extracted by the Explainer tend to reveal the root cause of an incident but provide only a little to non additional context. Rules mined by FISM appeared as an optimal mix of root cause events together with a reasonable amount of additional context.

As the future work, we would like to extend our research into the other application domains and also to implement the work of Martens et al. [**?**] and compare their numerical representations of comprehensibility and justifiability with our domain knowledge.

Acknowledgement

The research reported in this paper has been supported by the Czech Science Foundation (GAČR) grant 17-01251 and by Czech Technical University student grant SGS17/209/OHK3/3T/18.

References

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, volume 1215, pages 487–499, 1994.

[3] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Detecting outlying properties of exceptional objects. *ACM Transactions on Database Systems (TODS)*, 34(1):7, 2009.

[4] Zachary K Baker and Viktor K Prasanna. Efficient hardware data mining with the apriori algorithm on fpgas. In *Field-Programmable Custom Computing Machines, 2005. FCCM 2005. 13th Annual IEEE Symposium on*, pages 3–12. IEEE, 2005.

[5] Christian Borgelt. Efficient implementations of apriori and eclat. In *FIMI03: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2003.

[6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] Randy Carraghan and Panos M Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9:375–382, 1990.

[8] Kumar Chellapilla, Kevin Larson, Patrice Simard, and Mary Czerwinski. Designing human friendly human interaction proofs (hips). In *Proceedings of the SIGCHI conference on HumanFactors in Computing Systems*, pages 711–720. ACM, 2005.

[9] Salvador Espana-Boquera, Maria Jose Castro-Bleda, Jorge Gorbe-Moya, and Francisco Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):767–779, 2011.

[10] Alex Alves Freitas. Comprehensible classification models: a position paper. *SIGKDD Explorations*, 15:1–10, 2013.

[11] Petr Hájek, Ivan Havel, and Metoděj Chytil. The GUHA method of automatic hypotheses determination. *Computing*, 1(4):293–308, 1966.

[12] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOND Record*, volume 29, pages 1–12. ACM, 2000.

[13] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining—a general survey and comparison. *ACM SIGKDD Explorations newsletter*, 2(1):58–64, 2000.

[14] Marek Jilek. Machine learning based context extraction for network security incidents. Master's thesis, Faculty of Information Technology, Czech Technical University in Prague, 2018.

[15] Edwin Knorr and Raymond T Ng. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1998.

[16] Martin Kopp and Martin Holeňa. Evaluation of association rules extracted during anomaly explanation. In *ITAT 2015: Information Technologies - Applications and Theory*, 2015.

[17] Martin Kopp, Matěj Nikl, and Martin Holeňa. Breaking captchas with convolutional neural networks. In *ITAT 2017: Information Technologies - Applications and Theory*, 2017.

[18] Shailesh Kumar, V Chandrashekar, and CV Jawahar. Logical itemset mining. In *Data Mining Workshops (ICDMW),*

[2] When a webpage is opened via a browser it downloads a lot of different directories and files such as cascade style sheets, html file, image folder, javascript libraries, etc.

2012 IEEE 12th International Conference on, pages 603–610. IEEE, 2012.

[19] David Martens, Jan Vanthienen, Wouter Verbeke, and Bart Baesens. Performance of classification models from a user perspective. *Decision Support Systems*, 51(4):782–793, 2011.

[20] Barbora Micenková, Raymond T Ng, Xuan-Hong Dang, and Ira Assent. Explaining outliers by subspace separability. In *IEEE 13th International Conference on Data Mining (ICDM 2013)*, 2013.

[21] Lukáš Neumann and Jiří Matas. Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3538–3545. IEEE, 2012.

[22] Tomáš Pevný and Martin Kopp. Explaining anomalies with sapling random forests. In *Information Technologies - Applications and Theory Workshops, Posters, and Tutorials (ITAT 2014)*, 2014.

[23] Marko Robnik-Šikonja, Igor Kononenko, and Erik Štrumbelj. Quality of classification explanations with prbf. *Neurocomputing*, 96:37–46, 2012.

[24] Margaret Rouse. Ransomware, 2018. https://searchsecurity.techtarget.com/definition/ransomware [Cited 15 May 2018].

[25] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/-maximal itemsets. In *FIMI*, volume 126, 2004.

[26] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 12(3):372–390, 2000.