An Application of Combinatorial Methods for Explainability in Artificial Intelligence and Machine Learning

D. Richard Kuhn
Computer Security Division
Information Technology Laboratory

Raghu N. Kacker
Computational and Applied Math Division
Information Technology Laboratory

May 22, 2019

National Institute of Standards and Technology
U.S. Department of Commerce

Abstract

This short paper introduces an approach to producing explanations or justifications of decisions made in some artificial intelligence and machine learning (AI/ML) systems, using methods derived from those for fault location in combinatorial testing. We show that validation and explainability issues are closely related to the problem of fault location in combinatorial testing, and that certain methods and tools developed for fault location can also be applied to this problem. This approach is particularly useful in classification problems, where the goal is to determine an object's membership in a set based on its characteristics. We use a conceptually simple scheme to make it easy to justify classification decisions: identifying combinations of features that are present in members of the identified class but absent or rare in non-members. The method has been implemented in a prototype tool called ComXAI, and examples of its application are given. Examples from a range of application domains are included to show the utility of these methods.

Keywords

artificial intelligence (AI); assurance of autonomous systems; combinatorial testing; covering array; explainable AI; machine learning.

Disclaimer

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.

Additional Information

For additional information on NIST's Cybersecurity programs, projects and publications, visit the Computer Security Resource Center, https://csrc.nist.gov. Information on other efforts at NIST and in the Information Technology Laboratory (ITL) is available at https://www.nist.gov and https://www.nist.gov/itl.

Public Comment Period: *May 22, 2019 through July 3, 2019*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: xai@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA).

1 Background

Artificial intelligence and machine learning (AI/ML) systems have exceeded human performance in nearly every application where they have been tried. AI is also starting to be incorporated into commercial applications such as loan qualification, diagnostics, and steering and braking functions in passenger vehicles. This trend is accelerating, and AI will be increasingly used in safety-critical systems. AI systems are good, but sometimes make mistakes, and human users will not trust their decisions without explainable justification. There is a tradeoff between AI accuracy and explainability: the most accurate methods, such as convolutional neural nets (CNNs), provide no explanations; understandable methods, such as rule-based, tend to be less accurate [1][2][3][4].

Traditional expert systems based on rules such as *"if symptom A is present with symptom B or C, then diagnosis is X"* provide understandable explanations, but their accuracy is typically lower than obtained through methods such as neural networks. Conversely, neural nets provide a conclusion with no explanation of internal calculations that led to a particular result. Such black-box predictions are generally inadequate, because explanations must be understandable by non-specialists, who know their own subject area but may not be familiar with AI/ML algorithms. Subject matter experts are more likely to think in terms of the original expert systems, looking for the presence or absence of various values or properties that indicate a particular result. Consequently, an extensive body of research has been developed with the goal of explaining black-box AI/ML predictions [2][3].

In many ways, the classification problem in machine learning is closely related to the problem of fault location in combinatorial testing for software. The objective in both cases is to identify combinations of properties or values, out of a very large number that trigger a failure in the system under test (in combinatorial testing) or produce a conclusion (in machine learning). Methods and tools developed for fault location in combinatorial testing can be adapted to ML problems, to identify those rare combinations of variable values that produce conclusions in these systems.

2 Fault location

The fault location problem in combinatorial testing is extremely difficult to solve, but simple to state: given a set of tests for which the system under test fails, which combinations of the values of only a few factors triggered the failure? The reason this is a challenging problem is that more than one factor may be involved in triggering a failure of the system. For example, if four factors or values must be present to induce a failure, and we have 15 input variables, every test includes $\binom{15}{4} = 1365$ 4-way combinations. If there are multiple failing tests, then the number of failure-triggering combinations which need to be checked can number in the thousands. The question for testers is which of these thousands of combinations of the values of only a few factors cause the failure.

The conventional paradigm for solving this problem is to identify combinations that occur in failing tests but not in passing tests. If a combination of factor values occurs in a passing test, then clearly it did not trigger the failure. Thus, combinations that occur only in failing tests are those that are considered in narrowing down the set of suspect combinations. This is illustrated in Fig. 1. Some combinations occur in both passing tests (P/pattern) and failing tests (F/gray), but those in failing tests only are the suspect combinations.

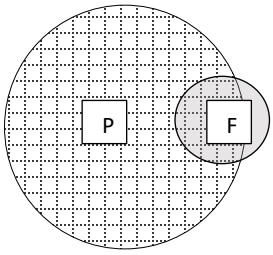

Figure 1: Passing and failing combinations

After identifying combinations that appear in only failing tests, a variety of heuristics can be used in identifying the likely cause of a failure. For example, combinations that occur more frequently in failing tests may be considered first. Additional tests containing suspect combinations can then be executed (run) to rule out irrelevant combinations and incrementally reduce the set of potential causes. In all cases, though, there are combinations of the values of only a few factors that are unique to the failing tests.

Now consider the classification problem for AI/ML. Given an individual item in a particular class, what is it about this item that distinguishes it from non-class members? For example, a cat may have features furry, brown, whiskers, claws, and so on, while living things not in the cat class do not have these features. In other words, certain combinations of features justify classifying the animal as a cat, and not some other animal. The individual animal shares features with cat class but does not share features with non-cat classes. As shown in Fig. 2, there is a parallel with the fault location problem in combinatorial testing. Some combinations of features occur in both non-class (N) and class (C) items, but others are unique to the cat class, explaining why a particular animal can be categorized as a cat.

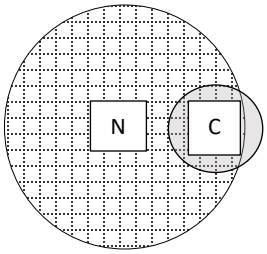

Figure 2: Class and non-class combinations.

A combinatorial testing tool, referred to as ComXAI, that applies this strategy has been developed and can be used to identify combinations that explain a classification result, treating the classifier as a black box system and not requiring any modification of the AI/ML algorithms.

The ComXAI tool is designed to accept as input a single observation of a particular classification, or a small number of observations. A much larger set of observations outside of the classification is also entered. Observations contain fields of categorical data, or partitions of numeric/continuous value data. For example, Fig. 3 shows a portion of the data used in Example 1, with 16 fields of categorical data (see Table 1), an observation classified as a reptile, and 96 observations classified as non-reptile. Also shown are the number of combinations for each *t*-way set: $\binom{16}{2} = 120$ 2-way combinations, 560 3-way combinations, etc. Traits or properties of the class observation are shown as well.

To determine combinations of traits or properties that are relevant in making a classification, ComXAI computes the proportion of *t*-way combinations of traits that occur in non-class observations. In example 1, single-value observations are shown in Fig. 3 and 2-way combinations are shown in Fig. 4. As seen in Fig. 4, 2.1 % of the non-reptiles have properties toothed=0 and number of legs = 4. Thus this 2-way combination of traits does not uniquely identify a reptile and therefore does not explain why the "reptile" conclusion was reached.

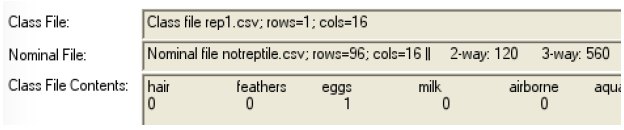

Figure 3: ComXAI example.

3 Examples

Example 1, animal classification: Consider the problem of explaining why an animal with the traits shown in Table 1 is classified as a reptile, where 0 = trait absent, 1 = trait present [5][6].

Table 1: Animal traits for identification.

hair=0,	feathers=0	egg laying=1	milk=0
airborne=0	aquatic=0	predator=0	toothed=0
backbone=1	breathes=1	venomous=0	fins=0
nlegs=4	tail=1	domestic=0	cat size=1

Fig. 4 shows the proportion of other animals, i.e., non-reptiles, in the database that possess the specified traits. Clearly no single feature is sufficient to identify the animal as a reptile, because it shares any of these single traits with more than 40 % of the other animals (e.g., 55.2 % are hairless, 79.2 % are featherless, etc.).

```
---------------
0053 occurrences = 0.552 of cases, hair = 0
0076 occurrences = 0.792 of cases, feathers = 0
0055 occurrences = 0.573 of cases, eggs = 1
0055 occurrences = 0.573 of cases, milk = 0
0072 occurrences = 0.750 of cases, airborne = 0
0061 occurrences = 0.635 of cases, aquatic = 0
0044 occurrences = 0.458 of cases, predator = 0
0039 occurrences = 0.406 of cases, toothed = 0
0078 occurrences = 0.813 of cases, backbone = 1
0076 occurrences = 0.792 of cases, breathes = 1
0090 occurrences = 0.938 of cases, venomous = 0
0079 occurrences = 0.823 of cases, fins = 0
0036 occurrences = 0.375 of cases, nlegs = 4
0070 occurrences = 0.729 of cases, tail = 1
0083 occurrences = 0.865 of cases, domestic = 0
0043 occurrences = 0.448 of cases, catsize = 1
```

Figure 4: Animal traits.

Similarly, no pair of features is sufficient to identify the animal as a reptile, as shown in Fig. 5. For example, 2.1 % of the non-reptiles are both toothless and have four legs, 5.2 % are hairless and have four legs, and so on.

However, consider the 3-way combinations of traits shown in Fig. 6. Non-reptiles in the database do not have these 3-way combinations.

```
0002 occurrences = 0.021 of cases, toothed,nlegs = 0,4
0005 occurrences = 0.052 of cases, hair,nlegs = 0,4
0005 occurrences = 0.052 of cases, milk,nlegs = 0,4
0006 occurrences = 0.063 of cases, eggs,nlegs = 1,4
0008 occurrences = 0.083 of cases, toothed,catsize = 0,1
0011 occurrences = 0.115 of cases, milk,catsize = 0,1
0012 occurrences = 0.125 of cases, eggs,catsize = 1,1
0013 occurrences = 0.135 of cases, hair,catsize = 0,1
0015 occurrences = 0.156 of cases, predator,catsize = 0,1
0015 occurrences = 0.156 of cases, predator,nlegs = 0,4
0017 occurrences = 0.177 of cases, airborne,toothed = 0,0
0019 occurrences = 0.198 of cases, feathers,toothed = 0,0
0020 occurrences = 0.208 of cases, predator,toothed = 0,0
0021 occurrences = 0.219 of cases, hair,predator = 0,0
0021 occurrences = 0.219 of cases, toothed,backbone = 0,1
0022 occurrences = 0.229 of cases, hair,aquatic = 0,0
```

Figure 5: 2-way combinations of traits.

```
00000 occurrences = 0.000 of cases, aquatic,toothed,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, eggs,aquatic,nlegs = 1,0,4
00000 occurrences = 0.000 of cases, hair,aquatic,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, hair,nlegs,catsize = 0,4,1
00000 occurrences = 0.000 of cases, milk,aquatic,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, milk,nlegs,catsize = 0,4,1
00000 occurrences = 0.000 of cases, predator,toothed,nlegs = 0,0,4
00001 occurrences = 0.010 of cases, eggs,nlegs,catsize = 1,4,1
00001 occurrences = 0.010 of cases, eggs,predator,nlegs = 1,0,4
00001 occurrences = 0.010 of cases, feathers,toothed,backbone = 0,0,1
```

Figure 6: 3-way combinations of traits.

Thus, each of these 3-way combinations will distinguish this animal from other animal types in this database. Only reptiles have these combinations of features

not aquatic AND not toothed AND four legs
OR egg-laying AND not aquatic AND four legs
OR not hairy AND four legs AND cat size
OR not milk-producing AND not aquatic AND four legs
OR not milk-producing AND four legs AND cat size
OR not predator AND not toothed AND four legs

In other words, the presence of any of these combinations of traits justifies the conclusion that the animal is a reptile. A single trait would not justify this conclusion, because the individual traits are present in large percentages of other animals (Fig. 4). Similarly, no pair of features (Fig. 5) is unique to reptiles, because small percentages of the other animals have these pairs as well. However, by considering 3-way combinations of traits, we can produce the easily understood rule below:

IF not aquatic AND not toothed AND four legs
OR egg-laying AND not aquatic AND four legs
OR not hairy AND four legs AND cat size
OR not milk-producing AND not aquatic AND four legs
OR not milk-producing AND four legs AND cat size
OR not predator AND not toothed AND four legs
THEN classification = reptile

Example 2, sensor data analysis: Another example [7][8] is a data set captured from sensors deployed for determining if rooms are empty or human-occupied. Variables are temperature, humidity, light, CO2, and humidity ratio. The objective is to develop rules to determine if a room is occupied, based on sensor readings. A variety of machine learning algorithms provide successful predictions, but an explanation is needed. No single variable value is sufficient, and as shown in Fig. 7, 2-way combinations include some strong indications of occupancy, but do not uniquely identify an occupied room.

That is, there are two combinations, with particular ranges of humidity/light, and light/CO2, that occur in only 0.2 % of empty rooms, so these value combinations suggest that the room is occupied. Considering 3-way combinations, shown in Fig. 8, we have a combination that is not found in any empty room, i.e., it occurs only in occupied rooms.

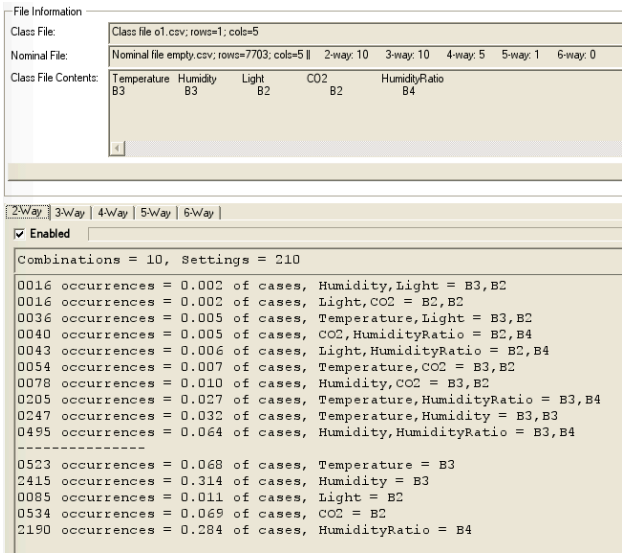

Figure 7: Sensor data, room occupancy.

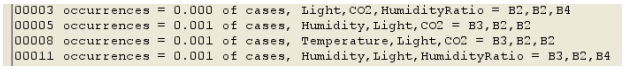

Figure 8: Sensor data, 3-way combinations.

An explanation for the conclusion of *occupied* for this data set could thus point out to the user that this 3-way combination of light, CO2, and humidity ratio is unique to occupied rooms, and that there are two 2-way combinations that are hardly ever found (0.2 %) in empty rooms.

Example 3: Lymphography. A lymphography data set that has been used in several machine learning experiments provides categorical data with 18 attributes on lymphoma [9][10]. Four possible class values to be predicted from the attributes are normal find, metastases, malign lymph, fibrosis.

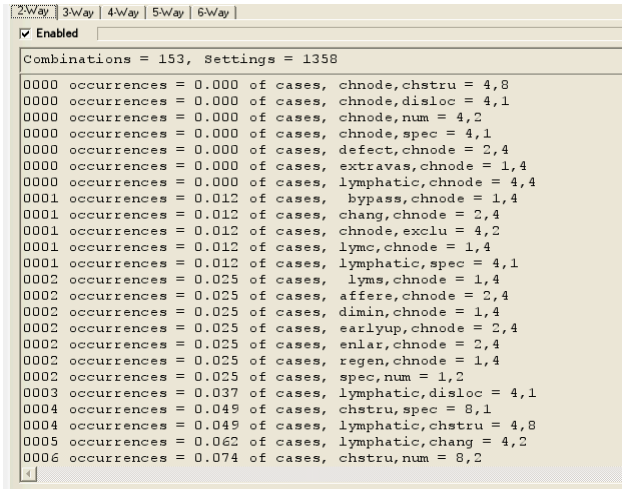

Figure 9: Lymphography data set 2-way combinations.

Analysis shown in Fig. 9 identifies seven 2-way combinations, out of 1358, that are unique to the example case. These combinations are characteristic of lymphoma that arises in a lymph node instead of metastatic that spread to the node from somewhere else.

4 Discussion

The method and tool described in this paper have been designed to provide intuitive explanations by identifying *t*-way combinations that are present in a given member of a class, and not present or extremely rare in non-members of the class. We believe this is a natural form of explanation because it relies on observable features but quantifies the degree to which feature combinations occur in the class and non-class sets. Using methods developed for fault location makes it possible to apply the approach across a very large number of *t*-way combinations, providing strong justifications for AI/ML conclusions.

It should also be noted that identifying *t*-way combinations of features that distinguish a class member is essentially the same as specifying predicates in a rule-based expert system. Referring back to Example 1, the six 3-way combinations could be mapped directly to a rule such as "if *not aquatic && not toothed && four legs || egg-laying && not aquatic && four legs* …. then genus = Testudo". It is often suggested that rule-based expert systems are the most interpretable, so this correspondence between *t*-way combinations and rule-based predicates also suggests that the ComXAI explanations can be understood well by users. At this preliminary stage, the approach has not been validated with human users, but this will be done in future research.

The correspondence between *t*-way predicates and rules discussed above also suggests the possibility of using the ComXAI approach to implement a machine learning algorithm for building a classification model, rather than only for explaining conclusions of other ML algorithms. While this may be practical, and could be considered in the future, there may be some limitations for this type of application. Since thousands of combinations can be included for even small problems, rule sets may become very large. Overfitting, in which a learning model incorporates noise variation from the training data, might be a problem with using combinations in this way. Work would be needed to determine if overfitting would occur and if it could be avoided to produce useful models, in addition to determining the accuracy that could be achieved.

5 Conclusions and Research Directions

The methods described in this note have been successfully implemented for fault location in combinatorial testing, and show potential for explanations in AI/ML. For both problems, the objective is to identify a combination of the values of only a few factors that lead to a specified result, either triggering an identified error, or uniquely identifying an individual observation as part of a class. As shown in this paper, after identifying combinations that are unique to a class, it is trivial to map these combinations into the form of if-then rules that are considered the most easily understandable AI/ML scheme for human users. This method can be applied to any black-box machine learning algorithm.

In future work, the approach described here may be implemented in tools with improved user interfaces or integrated into existing AI/ML platforms. This method may also have utility for validation of AI models. Combinations of few factor values that characterize a class should conform with user expectations. If not, there may be a deficiency in the derived model, or some unanticipated relationship may be affecting results, in either case leading to better understanding of the model.

References

[1] Gunning D ([2018]) *Explainable Artificial Intelligence (XAI)*. Available at https://www.darpa.mil/program/explainable-artificial-intelligence

[2] Pedreschi D, Giannotti F, Guidotti R, Monreale A, Pappalardo L, Ruggieri S, Turini F (2018) Open the black box data-driven explanation of black box decision systems. *arXiv preprint*. https://arxiv.org/abs/1806.09936v1

[3] Adadi A, Berrada M (2018) Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6:52138-60. https://doi.org/10.1109/ACCESS.2018.2870052

[4] Došilović FK, Brčić M, Hlupić N (2018) Explainable artificial intelligence: A survey. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, (IEEE, Opatija, Croatia), pp 210-215. https://doi.org/10.23919/MIPRO.2018.8400040

[5] Lampert CH, Pucher D, Dostal J (2019) *animals with attributes 2: A free dataset for Attribute Based Classification and Zero-Shot Learning*. Available at https://cvml.ist.ac.at/AwA2/

[6] Lampert CH, Nickisch H, Harmeling S (2009) Learning to detect unseen object classes by between-class attribute transfer. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, (IEEE, Miami, FL), pp 951-958. https://doi.org/10.1109/CVPR.2009.5206594

[7] Candanedo L (2016) *Occupancy Detection Data Set*. Available at

https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

[8] Candanedo LM, Feldheim V (2016) Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buildings* 112:28-39. https://doi.org/10.1016/j.enbuild.2015.11.071

[9] University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia (1988) *Lymphography Domain*. Available at https://archive.ics.uci.edu/ml/datasets/Lymphography

[10] Michalski RS, Mozetic I, Hong J, Lavrac N (1986) The Multi-Purpose Incremental Learning System AQ15 and its Testing Applications to Three Medical Domains. *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence* (AAAI, Philadelphia, PA), pp 1041-1045.

[11] University of California Irvine (2019) *Machine Learning Repository*. Available at https://archive.ics.uci.edu/ml/index.php