Opening Black Boxes by XAI:
Explainable Root Cause Analysis for Modern Cloud Applications

by

Lilit Poghosyan

BA, YEREVAN STATE UNIVERSITY, 2013

A thesis submitted in partial satisfaction of

the requirements for the degree of

Master of Engineering

in

Industrial Engineering & Systems Management

in the

COLLEGE OF SCIENCE AND ENGINEERING

of the

AMERICAN UNIVERSITY OF ARMENIA

Supervisor: ____ **Ashot Harutyunyan**_____

Signature: _____ Date:_____

Committee Member: _____

Signature: _____ Date:_____

Committee Member: _____

Signature: _____ Date:_____

Committee Member: _____

Signature: _____ Date:_____

Content Copyright License (to be included with Technical Report)

March 2, 2004

LICENSE
Terms and Conditions for Copying, Distributing, and Modifying

Items other than copying, distributing, and modifying the Content with which this license was distributed (such as using, etc.) are outside the scope of this license.

1. You may copy and distribute exact replicas of the OpenContent (OC) as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the OC a copy of this License along with the OC. You may at your option charge a fee for the media and/or handling involved in creating a unique copy of the OC for use offline, you may at your option offer instructional support for the OC in exchange for a fee, or you may at your option offer warranty in exchange for a fee. You may not charge a fee for the OC itself. You may not charge a fee for the sole service of providing access to and/or use of the OC via a network (e.g. the Internet), whether it be via the world wide web, FTP, or any other method.

2. You may modify your copy or copies of the OpenContent or any portion of it, thus forming works based on the Content, and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified content to carry prominent notices stating that you changed it, the exact nature and content of the changes, and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the OC or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License, unless otherwise permitted under applicable Fair Use law.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the OC, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the OC, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Exceptions are made to this requirement to release modified works free of charge under this license only in compliance with Fair Use law where applicable.

3. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, distribute or modify the OC. These actions are prohibited by law if you do not accept this License. Therefore, by distributing or translating the OC, or by deriving works herefrom, you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or translating the OC.

NO WARRANTY
4. BECAUSE THE OPENCONTENT (OC) IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE OC, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE OC "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK OF USE OF THE OC IS WITH YOU. SHOULD THE OC PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE YOU ASSUME THE COST OF ALL NECESSARY REPAIR OR CORRECTION.

5. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MIRROR AND/OR REDISTRIBUTE THE OC AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE OC, EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

(This license is known as "OpenContent License (OPL)" and can be found at http://opencontent.org/opl.shtml)

Acknowledgements

Foremost, I would like to express my profound appreciation to my supervisor Dr. Harutyunyan for his continuous guidance and cooperation in completing this research. Moreover, my immense gratitude to AUA CSE People for the endless support, consistent encouragement, and willingness to assist in any way they could.

Table of Contents

List of Tables

List of Figures

Abstract

Cloud computing solves many critical problems. Together with the cost savings, it provides improved security, elasticity, mobility, flexibility, etc. However, there are potential risks that cloud consumers and end-users should be aware of. One of them is limited control over the resources. End users can manage the applications, data, and services owned on the cloud, but key administrative tasks and infrastructure management should be out of control. Another critical disadvantage affecting end-users is possible cloud downtimes with the inaccessibility of business data and services. Alternatively, cloud providers are responsible for cloud resources' availability, security, and productivity. They continuously monitor, manage and optimize cloud infrastructures and applications to avoid performance (health) degradations. Monitoring systems collect and store time series data, log data, and application traces for further analysis. These are the pillars of observability that reveal the hidden IT processes for detecting bottlenecks and performance degradations, including the root cause analysis of IT issues. The latest will help accelerate or/and automate the resolution process before those issues affect end-users and businesses. However, due to the complexity of modern cloud infrastructures and applications, analyzing the acquired information will be only feasible via AI-empowered data analytics known as AIOps. System administrators can no longer understand and timely manage complex environments without innovative ML-based solutions. Moreover, those solutions must be interpretable and explainable, otherwise, the system experts will not follow the recommendations without seeing and understanding the decision-making process. That is why explainable AI (XAI) is gaining more popularity. This paper shows how the classical explainable methods should be applied to application traces, revealing microservices potentially responsible for performance degradation.

Key Words: cloud computing, modern applications, distributed tracing, explainable AI, feature importance, boosting, importance analysis, neural networks, SHAP

Chapter 1

Introduction

Cloud computing is the reality of modern IT infrastructures and applications. It relies on networked servers that authorize the end-users to operate a specific program as a service on any device. Cloud computing provides numerous benefits like scalability, resiliency, security, elasticity, etc. Gartner [1] anticipates that cloud service spending will increase to around $500 billion this year. The global pandemic lockdowns served as additional amplifiers to the cloud shift, impacting the entire IT market.

However, cloud computing has several complications which, if not handled appropriately, will alleviate its benefits. The first and foremost is the cloud cost unpredictability. Optimization of resource utilization like CPU, memory, network, disk, etc., should be prioritized. One of the main problems that make it unrealistic for system administrators to perform reasonable optimizations is the cloud complexity with its heterogeneous and distributed structure. Moreover, cloud system reliability requires continuous monitoring of infrastructure and application components before the potential malfunctioning will affect end-users with performance degradations.

Modern monitoring systems provide sufficient observability of IT processes. They collect and visualize time series data, logs, and application traces to understand cloud systems' behavior. However, understanding operational flows and their interactions are impossible without AI/ML intelligent solutions due to the probabilistic nature of the processes. Gartner [2] emphasizes the importance of AI for IT operations, known as AIOps, for several critical missions like optimizations, predictions of potential IT issues, and root cause analysis (RCA) of performance degradations based on collected information/data.

In many industrial applications, ML/AI algorithms provide better **predictions** and **data understanding** than human experts. Unfortunately, those two goals are simultaneously infeasible, as more extensive predictable power results in poorer interpretability/explainability. AI solutions can be differentiated into white-box and black-

box approaches. White-box models provide relevant explainability but suffer from weak predictive capabilities. On the contrary, black-box models like deep learning are compelling but lacking in explainability.

Those solutions are hard to explain and can hardly be understood even by domain experts. However, for many applications, explainability can be a critical factor as end-users will not follow the solutions without sufficient understanding. Regardless of AI power, many users think it is risky to trust and follow the insights blindly. Instead, they need to understand the foundation of those insights.

1.1. Problem Statement

Explainable AI (XAI) [3,4] is becoming popular and more required. It provides sufficiently robust and interpretable models, helping other more traditional machine learning algorithms benefit from increased explainability. XAI enables trustworthy models where users comprehend the decision-making process and thus deal more efficiently with AI solutions. XAI builds trust between a user and AI, increases satisfaction with solutions, and leads to more actionable and robust models for predictions and root cause analysis.

To consume the benefits of cloud computing architectures, applications must be native-cloud. Such modern applications focus on modularity and independence of their services (microservices). Each microservice implements a business capability, runs its process and communicates via application programming interfaces (APIs) [5]. Observability of modern applications can be achieved via collecting time series, logs, and traces. However, the observability is not the same as the monitoring [6]. Its goal is to understand the IT processes, detect and resolve issues to keep systems efficient and reliable [7]. In this work, we focus on application traces/spans for improving the observability of the classical monitoring tools via XAI capabilities. ***One of the main goals is the detection of potentially responsible microservices that cause service performance degradation.*** When possible, we will also indicate those specific tags and their values that may better explain the problems and support the resolution process.

The methods we developed and discussed here can also be helpful for analytical engines that work with time series and logs. However, in those cases, the main problem remains data labeling. In the case of application traces, the labels can be naturally derived from traces based on the errors or durations.

1.2. Distributed Tracing for Modern Applications

Distributed tracing is a well-known technology for monitoring modern applications primarily based on microservices architecture. It tracks application requests as they flow from one microservice to another [7,8]. It is a technology designed specifically for a distributed system. It's a diagnostic technique that reveals how a set of services coordinates and handles individual user requests [8]. A single trace (see Fig. 1) typically shows the activity for a particular transaction or request within the application being monitored, from the browser or mobile device down to the database and back [8].

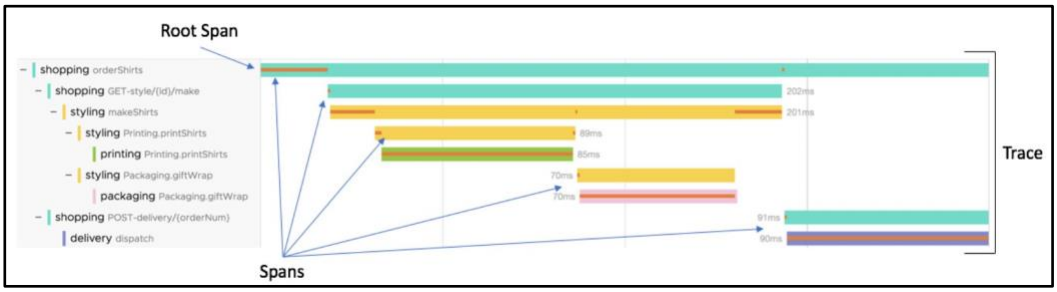


Figure 8. An example of a trace consisting of several spans.

In distributed tracing, a single trace contains a series of tagged time intervals called spans (see Fig. 1). A span can be considered a single unit of work. Spans have a start and end time and optionally may include other metadata like logs or tags that can help classify "what happened." Spans have relationships (including parent-child relationship), which

show the specific path a transaction takes through the numerous services or components that make up the application [8].

A group of traces (see Fig. 2) can show which microservice affects the most on performance. A monitoring system can track application performance via RED metrics (number of requests per minute, number of errors – failed requests per minute, p95 quantile of the durations in a minute) (see Fig. 2). Each span has a specific "error" tag indicating the status of the process. It has the value "true" in case of any problem. If the span is normal, the corresponding tag is missing. In our analysis, a trace is assigned to the erroneous class (label = 1) if one of the spans in the trace is erroneous. A trace to the normal class (label = 0) if all spans are normal. This entire information is visualized as a tracing browser shown in Fig. 2.

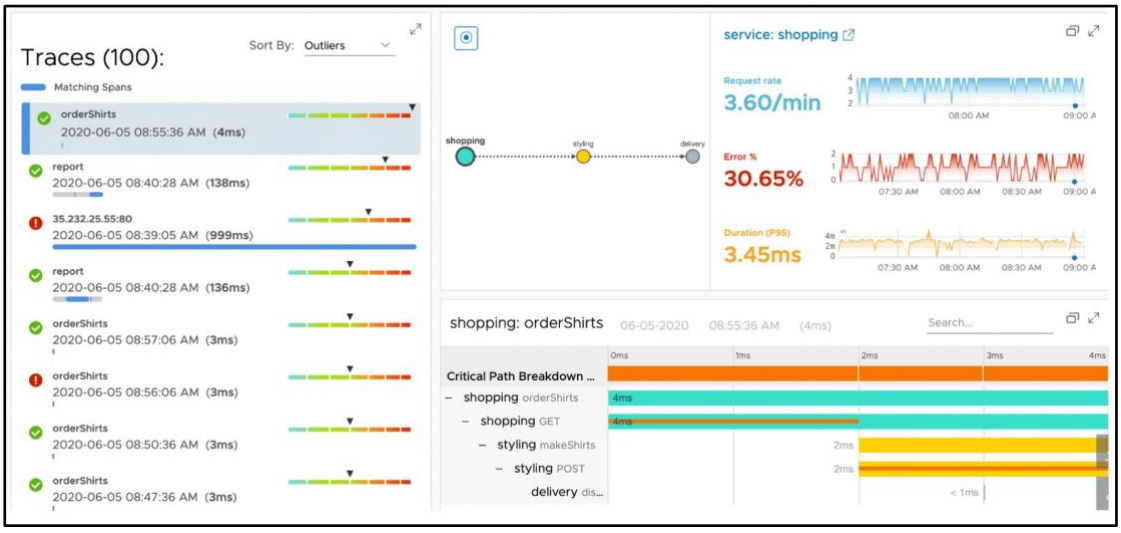


Figure 9. Collection of traces for an application troubleshooting.

Fig. 3 shows an application map that reveals the relationships of the microservices [9]. The colors indicate the statuses of the corresponding microservices. The colors optionally can be assigned by the values of RED metrics. The red-colored microservices have poor performances and our main goal is to understand/explain the problem. We can

select a specific poor microservice, collect tracing traffic passing through it and analyze via XAI methods for some acceptable interpretations of the performance degradations.

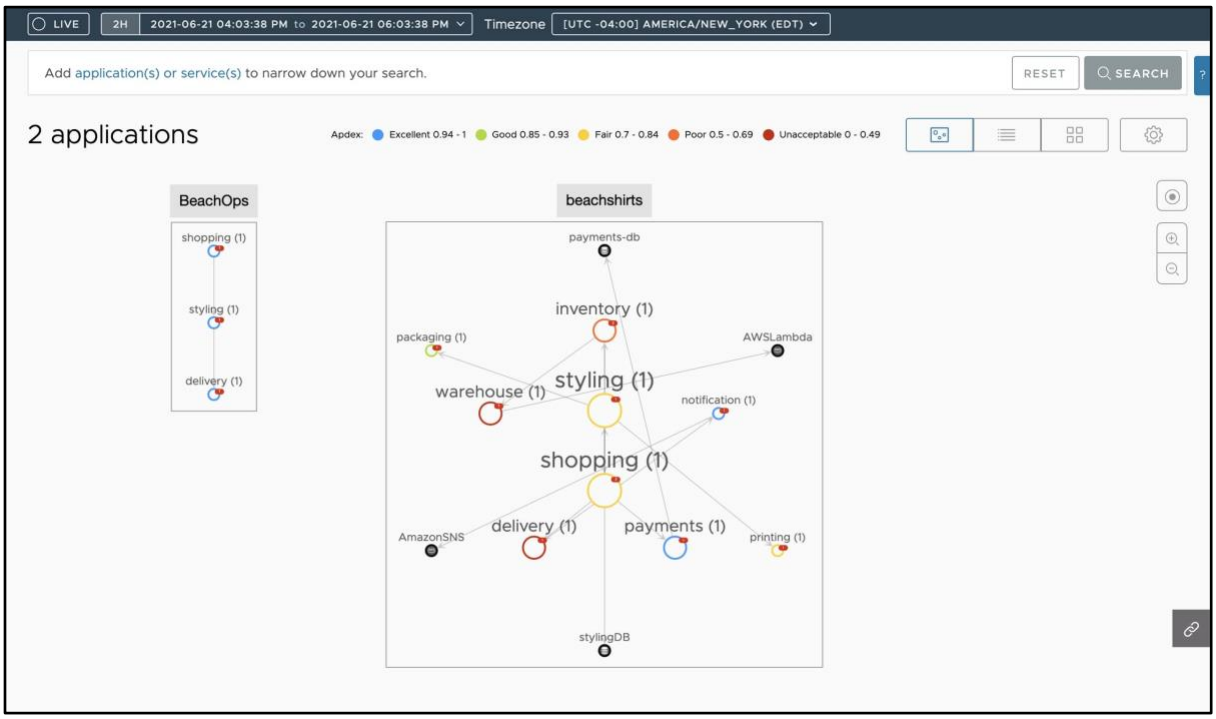


Figure 10. Application map.

1.3. Literature review

The problem of application troubleshooting based on traces has a long history of research. One such approach was described in [10] for intrusion detection that used "sendmail" system call-traces or "tcpdump" data. The problem is not the same as it is related to monolithic applications (not native-cloud), but the idea of application of XAI methods to troubleshooting has some resemblance. The article analyzed application of RIPPER for the explainability. According to Fürnkranz and Kliegr [11], RIPPER is still state-of-the-art in inductive rule learning. It has some interesting properties like supporting missing values, numerical and categorical variables, multiple classes, efficiently handling large noisy datasets, scaling nearly linearly with the number of instances in a dataset, etc... RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [12,13] was the first rule learning system that effectively countered the overfitting problem. Rule induction/learning systems [14-18] have the highest level of explainability. However, they have a low predictive power compared to other black-box models like boosting or neural networks. We can't rely on the explanations that result in low accuracy, precision or recall. Similar problems related to intrusion detection were discussed in papers [19-20].

XAI has many diverse applications in healthcare, judicial system, banking, financial services and insurance, etc. [21-25]. Those areas especially require trust and confidence to follow AI recommendations. XAI literature contains many approaches which can be divided into optional groups – local, global, model-agnostic, model built-in, etc. For example, tree-based methods like boosting, bagging and random forests have built-in importance capabilities. Model agnostic approach is "permutation feature importance". More current data agnostic approaches are LIME and SHAP.

LIME (local interpretable model-agnostic explanations) assumes that every complex model is linear on a local scale. Hence, it is possible to train a flexible global model with acceptable accuracy and fit a simple model around a single observation that will mimic how the global model behaves at that locality. SHAP (SHapley Additive exPlanations) is based on Shapley values emerging from the game theory [26-37].

Patented analytics utilizing rule induction methods and performing application troubleshooting is described in [52-54]. Those mentioned show application of RIPPER to the root cause analysis problem for revealing responsible microservices or their tags and tag-values that can explain health degradations indicated by the RED metrics. RIPPER outputs some (if, then) rules that contain span-names, tags and their threshold values. Labeling of the corresponding trace-traffic can be performed based on erroneous traces or their durations. RCA analysis can be applied simultaneously to both scenarios and the related recommendations can explain the problem from different angles. Those recommendations in terms of the span-names, span-tags, and their values can help system administrators understand the essence of problems and accelerate the remediation process. Unfortunately, as previously stated, RIPPER has a low predictive power like other rule induction methods or decision trees. So, its explainability capabilities can be helpful only for specific applications where RIPPER shows acceptable classification scores. Otherwise, we cannot rely on the rules and their recommendations.

We reproduce some of the results of [52-54], explain the cases when RIPPER is not applicable due to small classification scores and provide comparisons with more robust methods like boosting. In cases when RIPPER provides acceptable classification scores, we show how boosting will give almost the same recommendations. In all such cases, the results demonstrated by RIPPER can be viewed as a validation method for the boosting. We used the implementation of RIPPER in Weka.

Chapter 2

Research Methodology

2.1. Theoretical background

Application maps derived via distributed tracing reveal the relationship between microservices, databases, and other components. Moreover, based on the values of RED metrics, they indicate the statuses of the components. One of the common characteristics is Apdex metric (Application performance index) that allows us to monitor the health of a component in terms of the response time of service and the predefined response time threshold [38] (see Fig. 3). “Poor” or “Unacceptable” Apdex values of a specific component will trigger the root cause analysis engine to start investigations for the origins of problems. This engine is the critical component of the AIOps platform whit the main goal of detecting and explaining performance issues. RCA engine will collect the tracing traffic passing through the malfunctioning microservice, perform feature engineering, data labeling and apply XAI algorithms for interpretations. Traces have internal span-tags indicating their statuses – “normal” or “erroneous”. It means we don’t need to train models to predict future statuses as they are known. We need only explanations why the traces have “erroneous” statuses, which features (spans) are responsible for it, and specifically which span-tags and their values are causing traces to malfunction.

There are plenty of ML algorithms with a high level of explainability - logistic regression with regularization, decision trees, tree-based methods (bagging, boosting, rule-induction systems). Unfortunately, ML models' explainability and predictive power have opposite directions (see Fig. 4). All models with highly explainable decision paths like decision trees, rule-induction systems, and linear approaches have low predictive powers. It implies we cannot rely on the provided explanations as they lead to low-power predictions. Merely, those explanations are misleading, especially for complex, non-linear datasets (environments). Hence, we can only rely on the methods that are sufficiently

powerful and flexible with the highest possible predictive powers. Further, we can apply modern XAI methods like SHAP and LIME to reveal the decision-making process.

Our experiments started with the ensemble methods [39-41]. We apply interchangeably some boosting approaches like XGBClassifier [42], xgboost [42] and HistGradientBoostingClassifier [43]. They exhibit impressive predictive power with the accuracy, precision and recall for all classes close to 100%.

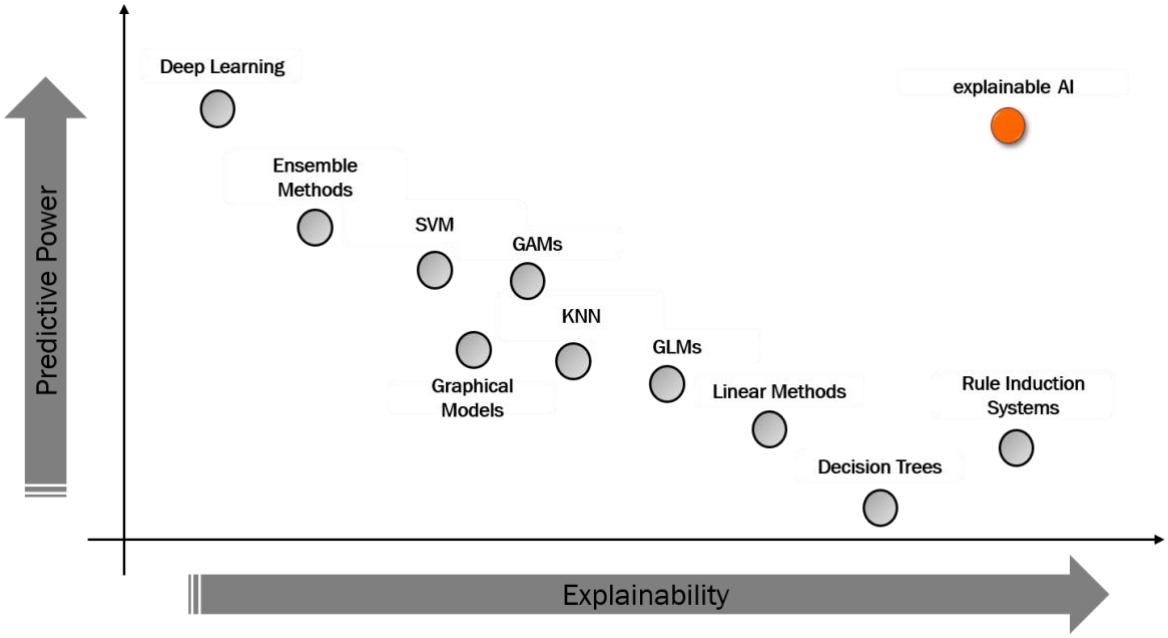


Figure 11. Explainability vs Predictive Power.

In general, tree-based methods have built-in feature-importance capabilities based on the impurity measures. Before each split in a tree node, all features must be verified with all possible split-values, and the most significant information gain (based on Entropy or Gini index) will specify the best choice. However, information gains corresponding to all features can be stored and averaged across all trees. The biggest average information gain will correspond to the most important feature. Then, all features can be ranked by their importance scores. So called as mean decrease in impurity (MDI). However, this method

can give high importance to features that may not predict unseen data when the model is overfitting [44]. Permutation-based feature importance avoids this issue [44]. It randomly shuffles the values of features and verifies the corresponding decrease in the model accuracy. The sharpest decrease corresponds to the most important feature. Permutation-based importance analysis is a model agnostic approach. It can be applied to any model. Worth noting that it is a rather time and resource-consuming approach. These methods provide global explainability. They are responsible for the entire dataset. It is impossible to select an observation, predict the label and try to explain it with those approaches.

Local explainability can be achieved via LIME or SHAP. LIME assumes that any global flexible method (model agnostic approach) can be locally approximated via explainable linear method like ridge or lasso. It can be applied to any observation and explain how its label was derived. SHAP estimates the contribution of individual features and their groups to the prediction accuracy. It should be infeasible to calculate the exact contributions of all possible groups (known as Shapley values), so SHAP is a collection of methods that estimates those contributions. SHAP can be applied both to the problems of local and global explainability.

Throughout the work, we perform comparisons with RIPPER. When the results of RIPPER are acceptable, we consider them as validation for the tree-boosting results. Otherwise, we discuss how tree-boosting is outperforming RIPPER in specific situations. There are few comparison choices as we need ML algorithms that can handle missing values.

In the next section, we will describe the datasets for the classification problems. Where, general problem related to our datasets was mentioned. Almost 90% of values are missing, and this is not a technical problem but closely related to the nature of feature engineering. There are two possibilities to work with such datasets. Apply methods that can handle the missing values like RIPPER, decision trees, boosting, and XAI methods that will also be able to ignore those values. Unfortunately, the list of methods with efficient implementations working with the missing values is concise. The next possibility is to fill

those values somehow. We need to be very careful as 90% will make models more time-consuming and negatively impact the predictions.

Another important XAI tool is data visualization. Sometimes relevant visualizations can reveal the nature of problems without applying classification/regression models. We show the potential of the application of t-SNE manifold learning method to our multidimensional data [45,46]. It projects the observations from a high-dimensional space into a low-dimensional space by preserving the "distances" between them. We show the results of projections into 2- and 3-dimensional spaces and the labels in different colors to reveal some hidden patterns in the datasets. It can probably help filter out some noise and increase the predictive power of approaches with higher explainability than NN and ensemble classifiers.

2.2. Data Creation

Let us consider a customer dataset containing 5899 traces. It is trace traffic passing through a specific microservice during several hours or minutes. The number of spans in each trace varies from 1 to 1000. It is a typical situation for modern applications with many services composed of different microservices. Those applications are very dynamic. Fig. 5 shows the plot of the numbers of spans across 5899 traces. We see an interesting periodic structure.

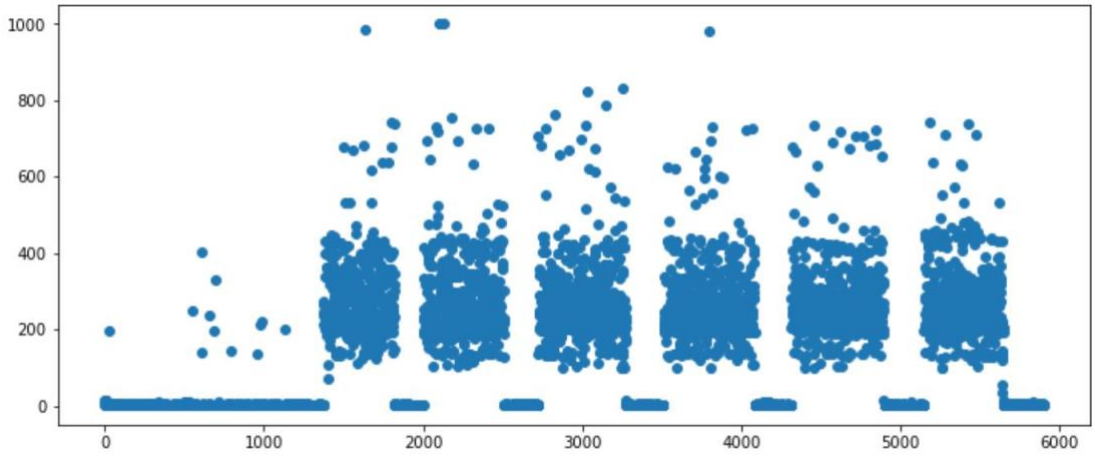


Figure 12. The number of spans in different traces.

Fig. 6 shows the 11th trace containing only 2 spans.
Their names are

Zipkin.ad-selector-canary.get-user-info-local

and

Zipkin.ad-selector-canary.getadstruct

Each trace has an identification number “traceId”, a start time “start_ms” and an end time “end_ms”. We are not using them. Trace duration is given at the end of a trace by “total_duration_ms”. It is an important trace characteristic. In general, normal processes have some identical durations. Extremely short or long durations may identify a service malfunctioning.

```
{'traceId': 'ab84a01c-04ef-48f4-8f14-ef0eddf5ade7',
 'spans': [{'name': 'Zipkin.ad-selector-canary.get-user-info-local',
   'host': 'zipkin',
   'startMs': 1596127195099,
   'durationMs': 2,
   'spanId': '653b9c49-5d50-4859-a255-a2fbbe1bd8a4',
   'traceId': 'ab84a01c-04ef-48f4-8f14-ef0eddf5ade7',
   'annotations': [{'parent': '9e991f5e-98f9-4bff-9e08-0d780859ebb6'},
    {'followsFrom': '9e991f5e-98f9-4bff-9e08-0d780859ebb6'},
    {'service': 'ad-selector-canary'},
    {'application': 'Zipkin'},
    {'cluster': 'none'},
    {'component': 'baseplate'},
    {'shard': 'none'},
    {'spanId': '653b9c49-5d50-4859-a255-a2fbbe1bd8a4'},
    {'traceId': 'ab84a01c-04ef-48f4-8f14-ef0eddf5ade7'},
    {'parent': '9e991f5e-98f9-4bff-9e08-0d780859ebb6'},
    {'error': 'true'},
    {'ipv4': '10.96.136.183'},
    {'_operationName': 'get-user-info-local'},
    {'_matched': 'true'}]},
  {'name': 'Zipkin.ad-selector-canary.getadsstruct',
   'host': 'zipkin',
   'startMs': 1596127195099,
   'durationMs': 3,
   'spanId': '9e991f5e-98f9-4bff-9e08-0d780859ebb6',
   'traceId': 'ab84a01c-04ef-48f4-8f14-ef0eddf5ade7',
   'annotations': [{'parent': '94e2411f-69d0-4a4a-ae71-9b432558feb3'},
    {'followsFrom': '94e2411f-69d0-4a4a-ae71-9b432558feb3'},
    {'service': 'ad-selector-canary'},
    {'application': 'Zipkin'},
    {'cluster': 'none'},
    {'span.kind': 'server'},
    {'component': 'baseplate'},
    {'shard': 'none'},
    {'spanId': '9e991f5e-98f9-4bff-9e08-0d780859ebb6'},
    {'traceId': 'ab84a01c-04ef-48f4-8f14-ef0eddf5ade7'},
    {'parent': '94e2411f-69d0-4a4a-ae71-9b432558feb3'},
    {'error': 'true'},
    {'ipv4': '10.96.136.183'},
    {'_operationName': 'getadsstruct'},
    {'_matched': 'true'}]}],
 'start_ms': 1596127195099,
 'end_ms': 1596127195102,
 'total_duration_ms': 3}
```


Figure 13. An example of a trace.

It means traces' duration can be used for dataset labeling. It will lead to a trace latency-based root cause analysis, which can be a problem for future research. In the current work, we consider data labeling based on erroneous traces. Both spans in Fig. 6 have tags “error” with the values “true”. Those values indicate erroneous spans. We assume that a trace is erroneous if it contains at least one erroneous span. In other words, a service has a problem if one of its microservices has a problem. Then, we label the corresponding traces as erroneous (label = 1). Otherwise, if the tag “error” is missing, a trace is normal with the (label = 0). We see that the spans in Fig. 6 have other tags containing information that can be useful for problem identification/explanation. Our target is an indication of the span names, their tags, and tag-values responsible for erroneous traces/spans. The dataset contains 3145 normal traces and 2754 erroneous ones. Main cause is the feature engineering for the described problem and the construction of the corresponding data frames (tabular data) based on 5899 traces with the approximate structures shown in Fig. 6. Let us discuss different scenarios.

The first scenario uses only the names of spans. We search the entire dataset for all available spans and construct the features by the set of span-names. We found 451 different span-names. The corresponding Dataframe will contain 451 columns for the names and one for the output. The number of rows (corresponding to various traces) is 5899. For each trace (row), we investigate its span-names and put value=1 for the related columns and value=missing for the others. This will be our first Dataframe (Dataframe1). We verified that 90% of values are missing. In this scenario, the RCA will indicate only the responsible spans for the erroneous traces. In some instances, we will try to fill the missing values by 0 or -1. It is possible that in a trace the same span-name appears several times with different tag-values. The number of repetitions can indicate problem in service. In the second scenario, we calculate the number of repetitions and put value = “number of repetitions” instead of value=1 as in the previous scenario. This will be our second Dataframe (Dataframe2). Hence, Dataframe1 and Datframe2 have the same column-names and missing values, the other values are 1 (Datframe1) or the number of repetitions of available

spans (Dataframe2). Figs. 7,8 show the projection of Dataframe1 into the 2,3-dimensional spaces with t-SNE via Jaccard distance.

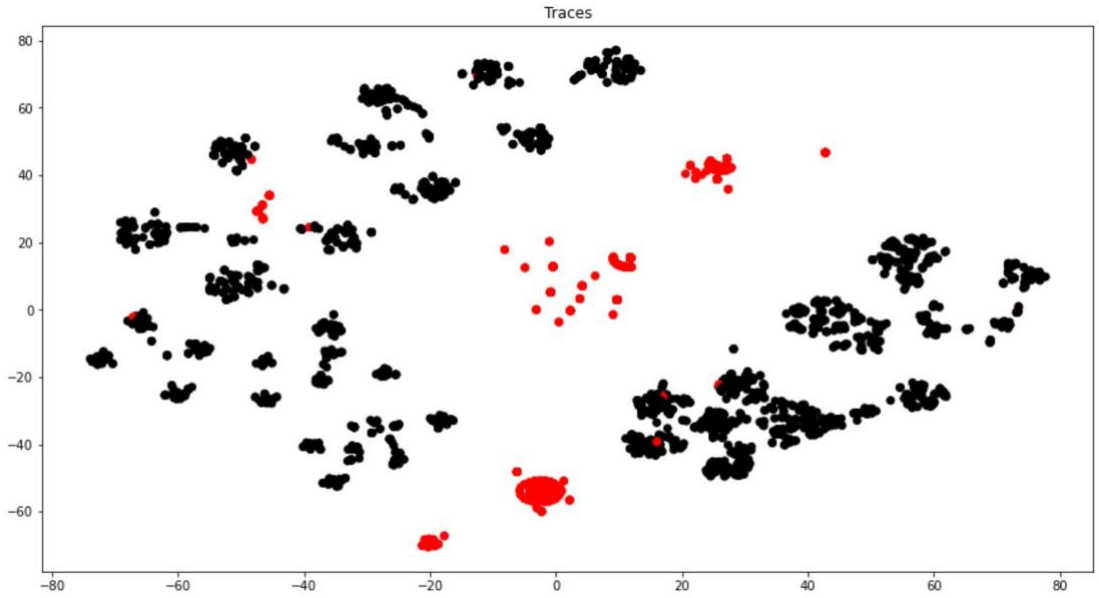


Figure 14. The projection of Dataframe1 to the 2-dimensional space via t-SNE manifold learning with Jaccard distance. Red points correspond to erroneous traces.

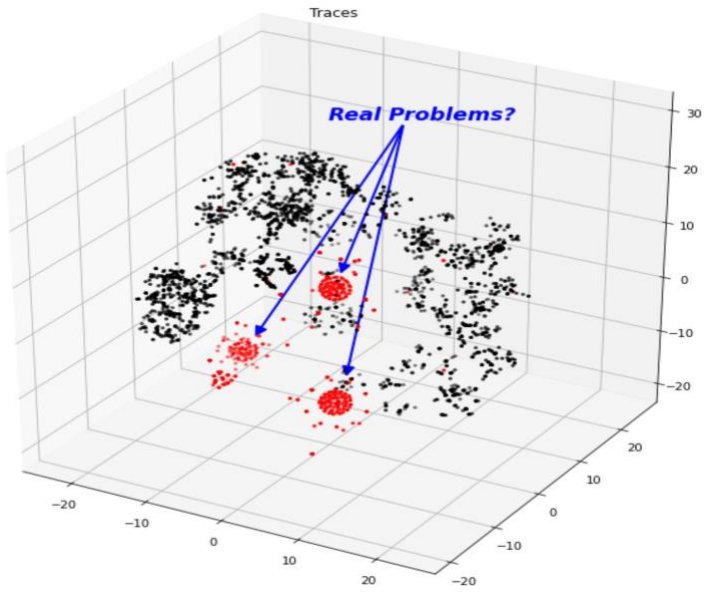


Figure 8. The projection of Dataframe1 to the 3-dimensional space via t-SNE manifold learning with Jaccard distance. Red points correspond to erroneous traces.

t-SNE (t-distributed Stochastic Neighbor Embedding) [47-51] converts the relationships of data points into probabilities. Gaussian joint probabilities represent the similarities in the original space, and Student's t-distributions represent the affinities in the embedded space. t-SNE is commonly using Euclidian distance, but for discrete vectors like in our case, the correct distance is Jaccard one.

The third scenario takes into consideration the tags of spans. The names of variables have the form "span-name + span-tag-name". We found 4916 such combinations. We search for all available spans and their tags for each trace and put the corresponding values in appropriate cells. Other cells of the same row will contain value=missing. Then, we combine this Dataframe with Dataframe1. This is our third Dataframe (Dataframe 3). It has 4899 rows and 4916+451+1 = 5368 columns. We verified that 90% of Dataframe3 contains missing values. We performed data preprocessing by removing some redundant variables and ended up with 4755 variables for Dataframe3.

In the fourth scenario, we merge the previous Dataframe composed of "span-name + span-tag-name" values and merge with Dataframe2 containing the number of repetitions of spans.

Fig. 9 shows some of the span-names. Fig. 10 shows some of the span-names plus span-tags. We see "Zipkin.activity.count_activity_multi" span. It has "annotations" with some tags like "_matched". Putting together, we constructed a new variable where we put the corresponding values of "_matched".

```
array(['Zipkin.activity.count_activity_multi',
       'Zipkin.activity.record_activity',
       'Zipkin.activity.redis.pipeline_cache', ...,
       'Zipkin.user-db-api-prod.getuserinfostruct',
       'Zipkin.user-db-api-prod.user-db-with-retry.getuserinfostruct',
       'Zipkin.user-db-api-prod.user-db.getuserinfostruct'], dtype='<U87')
```

Figure 9. Some of the span-names.

```
array(['Zipkin.activity.count_activity_multi_annotations__matched',
       'Zipkin.activity.count_activity_multi_annotations__operationName',
       'Zipkin.activity.count_activity_multi_annotations_application',
       ...,
       'Zipkin.user-db-api-prod.user-db.getuserinfostruct_annotations_span.kind',
       'Zipkin.user-db-api-prod.user-db.getuserinfostruct_durationMs',
       'Zipkin.user-db-api-prod.user-db.getuserinfostruct_host'],
      dtype='<U114')
```

Figure 10. Some of the span-names plus tag-names.

Chapter 3

Importance Analysis

3.1. Dataframe1

XGBoost is one of the robust tree-boosting implementations. In our cause, it is twice important as the XGBoost implementation handles the missing values. We experimented with XGBClassifier from xgboost library and HistGradientBoostingClassifier from scikit-learn library. The latest transforms numeric variables into categorical variables with a predefined number of bins and accelerates the computations. We also experimented with RIPPER from Weka library.

Let us start with Dataframe 1, which uses only the span-names of our dataset. This is the smallest and simplest Dataframe. Fig. 11 shows the rules of RIPPER and the corresponding accuracy. It shows empty list of rules for the positive class and assigns all observations to the negative class. As a result, the misclassification rate is 46.68% (the entire positive class is misclassified).

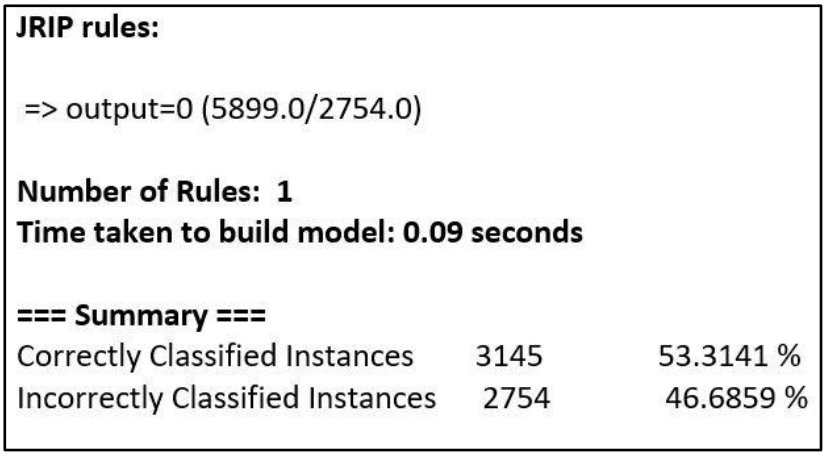


Figure 11. RIPPER applied to Dataframe1.

Contrarily, XGBClassifier trained an ideal model. We performed the tuning of hyperparameters on a train set (75% of the original set) via 5-fold cross-validation by means of GridSearchCV in scikit-learn. We tuned n_estimators=60 (the number of trees) and learning_rate=1. We used global parameters tree_method = "gpu_hist" and enable_categorical = "True". Fig. 12 shows the corresponding learning curve (with fixed learning_rate=1), where the grey shade corresponds to the standard deviations of the CV-test-accuracies calculated on 5 folds.

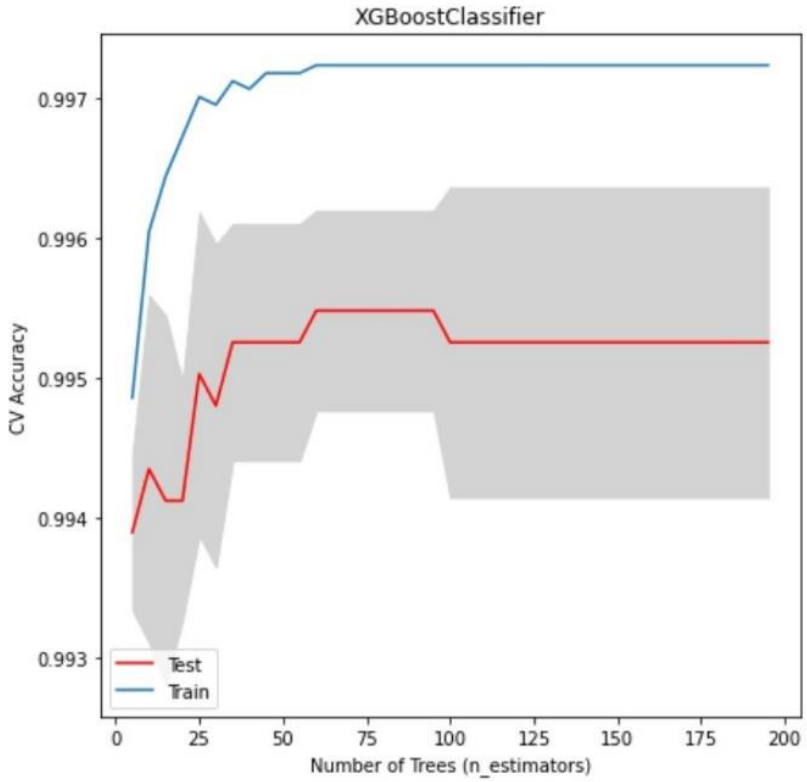


Figure 12. Learning curve for XGBClassifier on Dataframe1 with learning_rate=1.

Tables 1 and 2 summarize the results for train and test sets, respectively.

	precision	recall	F1-score
0	1	1	1
1	1	0.99	1

Table 2. Classification report for the XGBClassifier on the train set (75% of the dataset).

	precision	recall	F1-score
0	0.99	1	1
1	1	0.99	1

Table 2. Classification report for the XGBClassifier on the test set (25% of the dataset).

XGBClassifier shows a perfect optimal model with precision=1, recall=0.99 and F1=1 on a test set. Here is the difference of a higher predictive-power compared to rule-induction methods. Fig. 13 shows the list of important features based on the built-in MDI approach corresponding to the optimal model. We see three important microservices "Zipkin.ad-selector-canary.compare-edge-context", "Zipkin.ad-selector-canary.post-process-ads" and "Zipkin.ad-selector-canary.ad-selector-shard". The importance-scores of the other microservices are ignorable. It should be interesting to verify the importance of those features via permutation-based analysis.

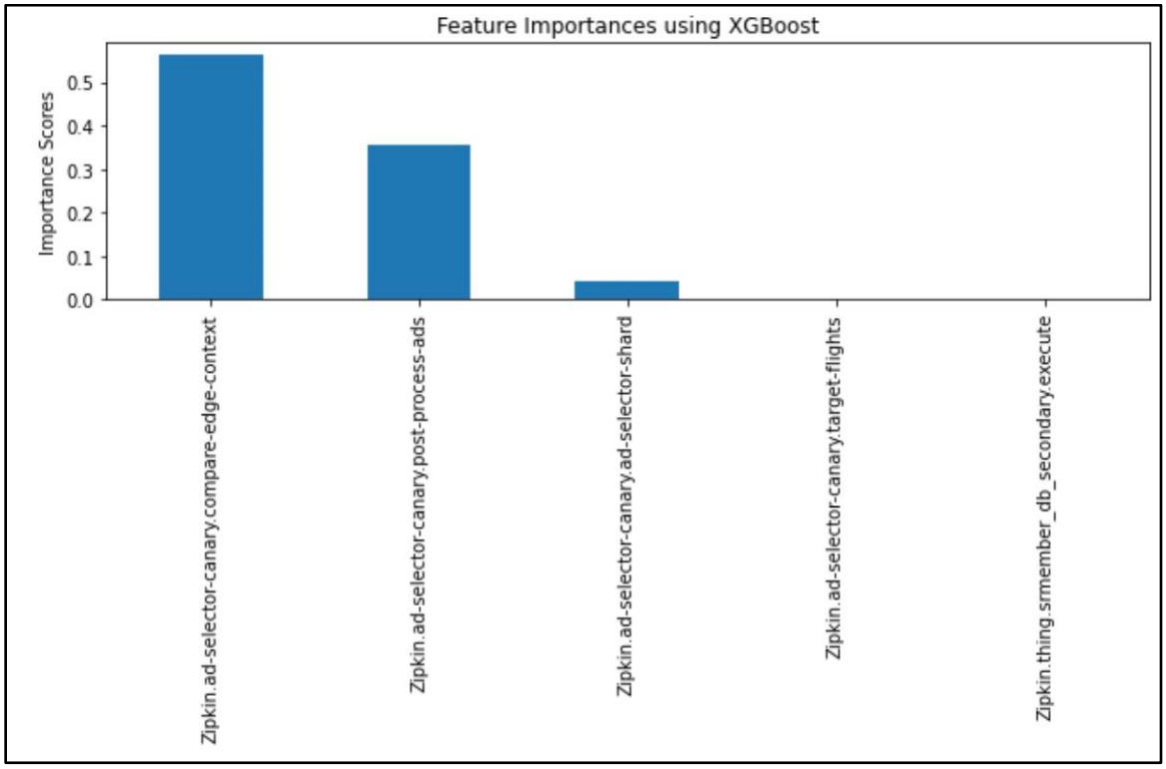


Figure 13. Important features corresponding to XGBoost.

Fig. 14 shows only one important feature "Zipkin.ad-selector-canary.compare-edge-context" derived via permutation-based analysis. Unfortunately, we cannot validate this recommendation as the real problem related to the trace-traffic is unknown (RIPPER also doesn't reveal any insights for the comparison). However, it is important that Figs. 13 and 14 return the same most important microservice. It is somehow increasing our confidence towards the recommendation.

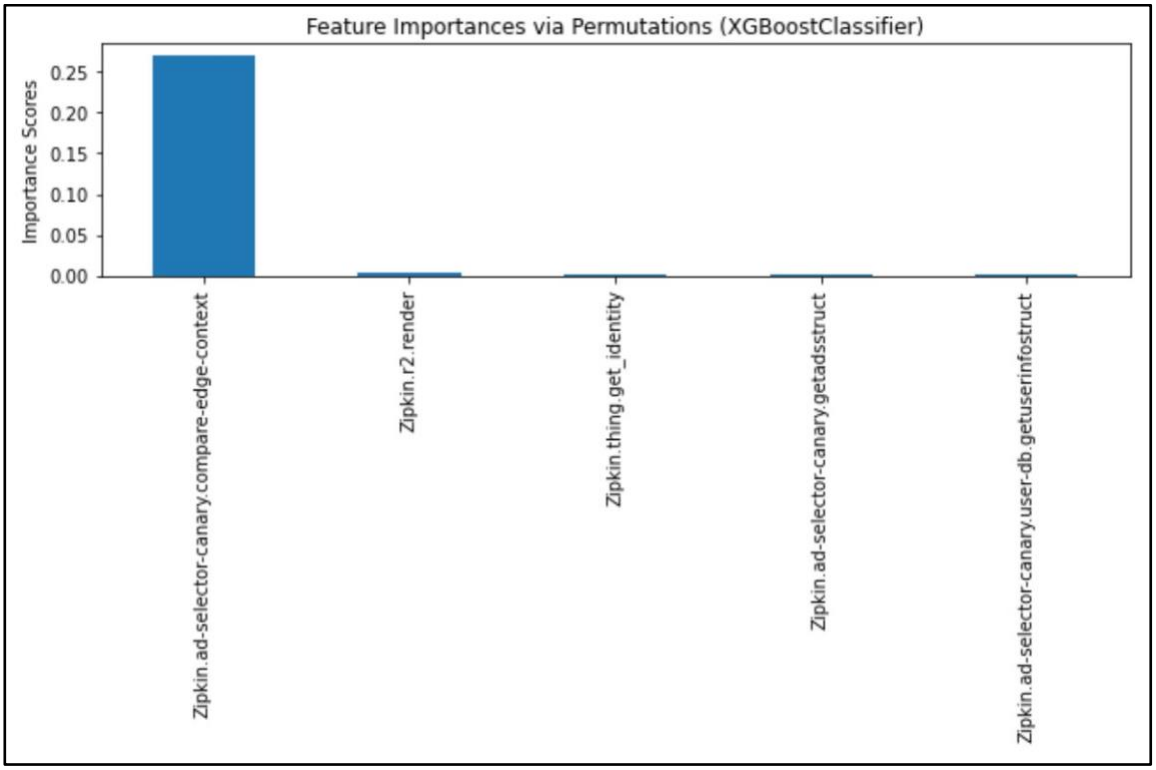


Figure 14. Important features corresponding to XGBoost with permutations.

Fig. 15 shows the list of important features calculated via SHAP based on the optimal HistGradientBoostingClassifier model. This model has the same characteristics as XGBClassifier. We used the locality of SHAP and applied it only to the positive class (label=1). Actually, it means global-local approach compared to the results of Figs. 13 and 14. Here also, the most important microservice is "Zipkin.ad-selector-canary.compare-edge-context". The second one is "Zipkin.ad-selector-canary.post-process-ads". Interestingly, the second one coincides with the second important feature shown in Fig. 13 that was ignored by the permutation-based approach.

However, just recalling that comparison of the lists derived by permutation-based approach and SHAP is not fare. SHAP shows the results only for the positive class while permutation-based approach returns a global explainability.

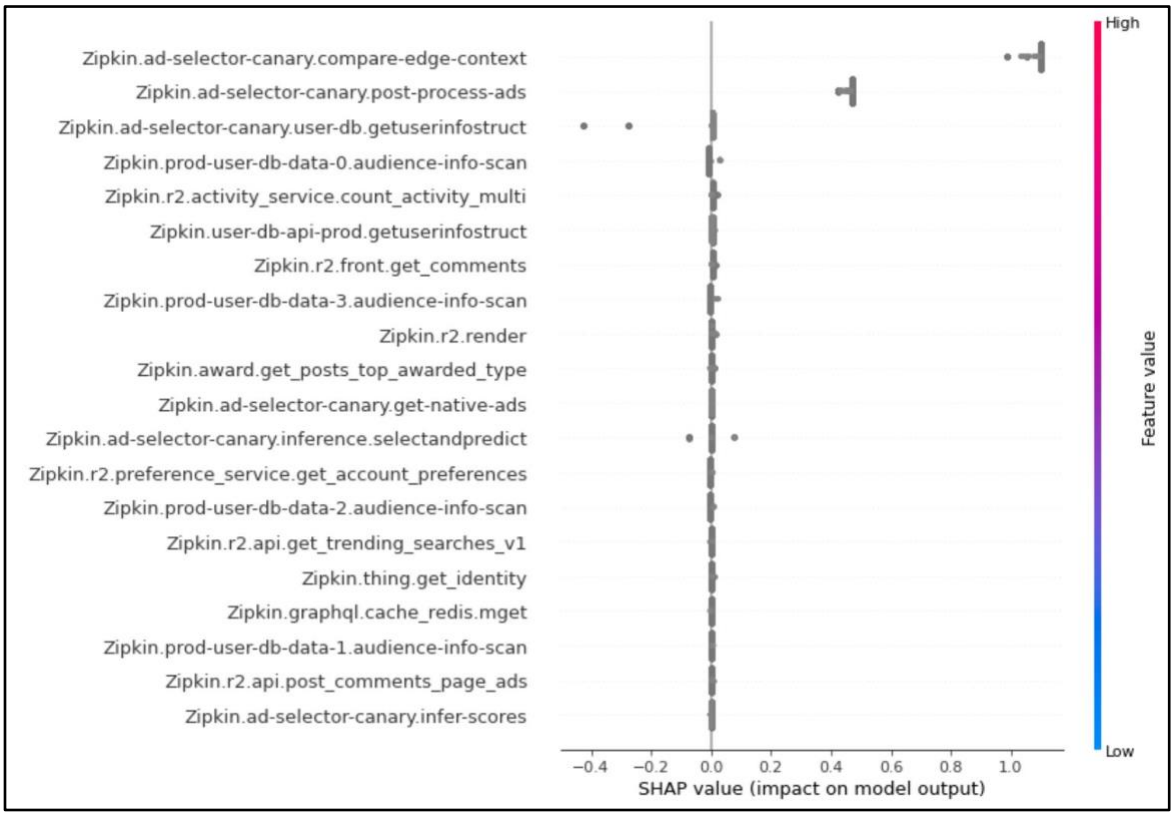
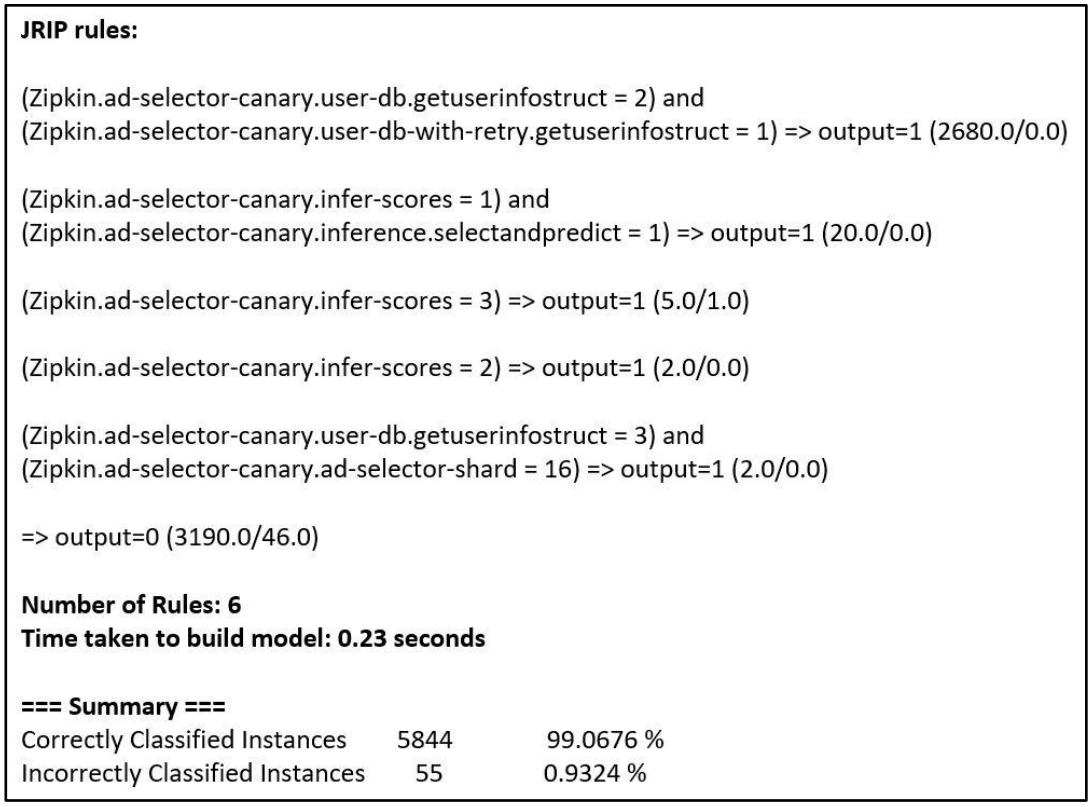


Figure 15. Important features corresponding to HistGradientBoostingClassifier with SHAP. The analysis is performed only for the positive class.

3.2. Dataframe2

Dataframe2 differs from Dataframe1 by counting the repetitions of spans in a trace. Duplications of the same spans in a trace may indicate some malfunctioning and it should be interesting the comparison of the results to the ones of Dataframe1. The corresponding insights are expected to be different. RIPPER is much more successful for Dataframe 2. This time the accuracy is 99.07% and it shows 6 rules (see Fig.16). Five rules correspond to the positive class. The largest coverage has the first rule. According to it, the most important spans are "Zipkin.ad-selector-canary.user-db.getuserinfostruct" and "Zipkin.ad-selector-canary.user-db-with-retry.getuserinfostruct" both related to DB. Let us see the

insights by tree-boosting. The optimal model has the same scores on train and test sets as Tables 1 and 2 show.



JRIP rules:

(Zipkin.ad-selector-canary.user-db.getuserinfostruct = 2) and
(Zipkin.ad-selector-canary.user-db-with-retry.getuserinfostruct = 1) => output=1 (2680.0/0.0)

(Zipkin.ad-selector-canary.infer-scores = 1) and
(Zipkin.ad-selector-canary.inference.selectandpredict = 1) => output=1 (20.0/0.0)

(Zipkin.ad-selector-canary.infer-scores = 3) => output=1 (5.0/1.0)

(Zipkin.ad-selector-canary.infer-scores = 2) => output=1 (2.0/0.0)

(Zipkin.ad-selector-canary.user-db.getuserinfostruct = 3) and
(Zipkin.ad-selector-canary.ad-selector-shard = 16) => output=1 (2.0/0.0)

=> output=0 (3190.0/46.0)

Number of Rules: 6
Time taken to build model: 0.23 seconds

=== Summary ===

Correctly Classified Instances	5844	99.0676 %
Incorrectly Classified Instances	55	0.9324 %

Figure 16. RIPPER applied to Dataframe2.

Fig.17 shows a single important feature "Zipkin.ad-selector-canary.user-db.getuserinfostruct" coinciding with the one revealed by RIPPER.

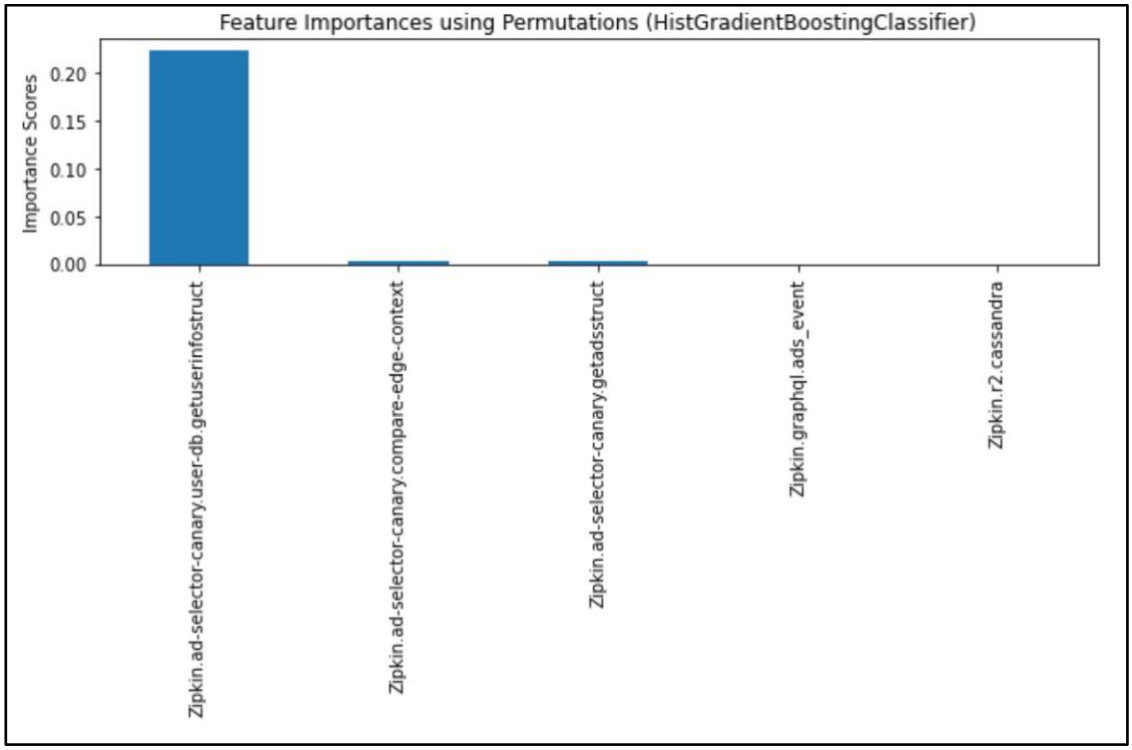


Figure 17. Permutation-based analysis for Dataframe2 via tree-boosting approach.

Fig. 18 shows the list of important features by SHAP. The most important is the same "Zipkin.ad-selector-canary.user-db.getuserinfostruct" span. The second important microservice is "Zipkin.ad-selector-canary.compare-edge-context" which appeared in the analysis of Dataframe1. The third one is "Zipkin.ad-selector-canary.post-process-ads" which appeared in Fig. 13. SHAP likewise to RIPPER reveals the impact of the feature-

values on the decision-making process (see colors in Fig. 18, where the grey colors correspond to the missing values). Recall that SHAP is applied only to the positive class.

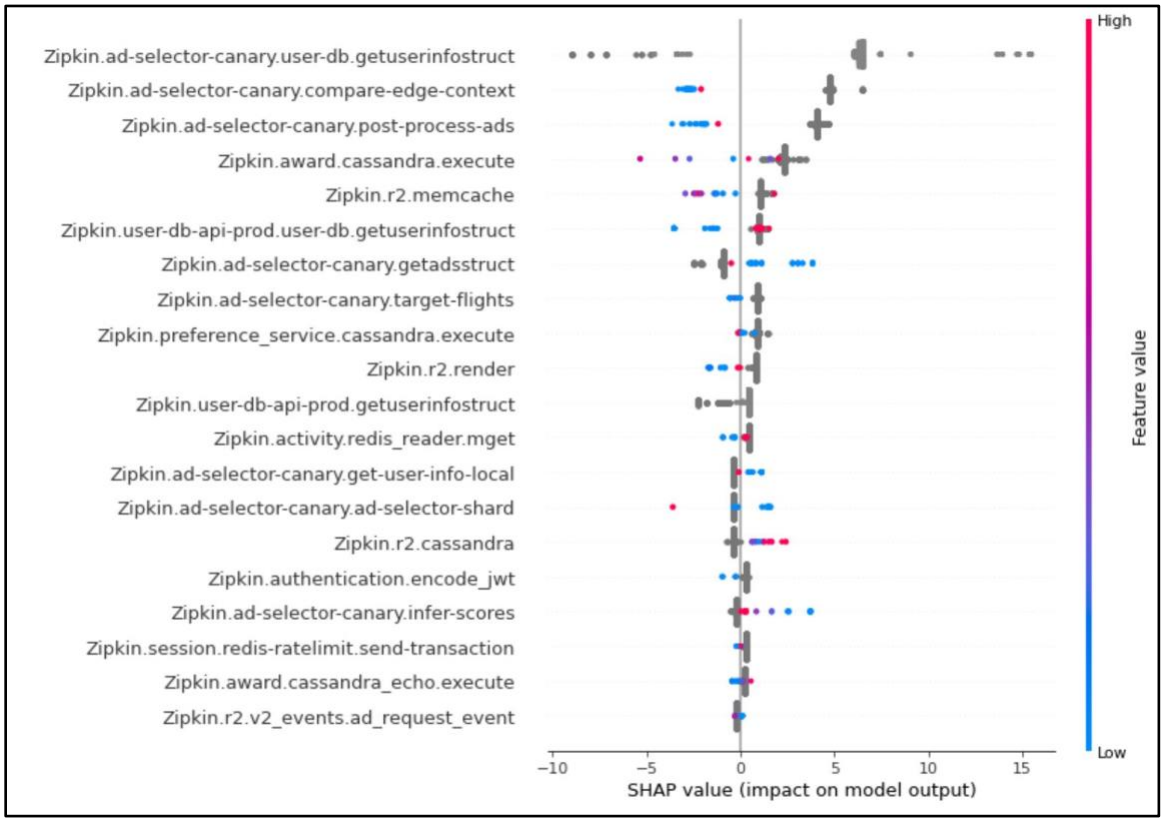


Figure 18. Important features corresponding to HistGradientBoostingClassifier with SHAP. The analysis is performed only for the positive class.

3.3. Dataframe3

Dataframe3 and Dataframe4 contain more detailed information than Dataframe1 and Dataframe2. Together with the span-names, they contain also span-tags and their values. Fig. 19 shows the results after application of RIPPER to Dataframe3. It shows 99.4% accuracy and 4 rules where 3 rules lead to the positive class. According to rules, the durations of microservices are important for explanations. This can be obvious, so we

remove all tags containing the string “duraionMs” and reapply RIPPER (see Fig. 20). This time, it failed with the unacceptable 53% accuracy.

JRIP rules:

(Zipkin.ad-selector-canary.user-db-with-retry.getuserinfostruct_durationMs >= 42) => output=1 (2685.0/1.0)

(Zipkin.ad-selector-canary.infer-scores_durationMs >= 43) => output=1 (18.0/0.0)

(Zipkin.ad-selector-canary.run-filters_durationMs >= 72) => output=1 (20.0/0.0)

=> output=0 (3176.0/32.0)

Number of Rules: 4
Time taken to build model: 5.23 seconds

=== Summary ===

Correctly Classified Instances	5865	99.4236 %
Incorrectly Classified Instances	34	0.5764 %

Figure 19. RIPPER applied to Dataframe3.

JRIP rules:

(Zipkin.user-db-api-prod.getuserinfostruct_annotations_ipv4 <= 9) and
(Zipkin.user-db-api-prod.getuserinfostruct_annotations_ipv4 >= 8) => output=1 (169.0/77.0)

(Zipkin.user-db-api-prod.getuserinfostruct_annotations_ipv4 <= 4) and
(Zipkin.user-db-api-prod.getuserinfostruct_annotations_ipv4 >= 3) => output=1 (35.0/13.0)

=> output=0 (5695.0/2640.0)

Number of Rules: 3
Time taken to build model: 4.55 seconds

=== Summary ===

Correctly Classified Instances	3132	53.0937 %
Incorrectly Classified Instances	2767	46.9063 %

Figure 20. RIPPER applied to Dataframe3 after removing all tags with the string “durationMs”.

Let us see the results of HistGradientBoostingClassifier. Fig. 21 indicates "Zipkin.ad-selector-canary.compare-edge-context_annotations__matched" span-tag. It corresponds to the span "Zipkin.ad-selector-canary.compare-edge-context" which has the collection of tags named as "annotations" with the tag "_matched". The second and third features are related to durations.

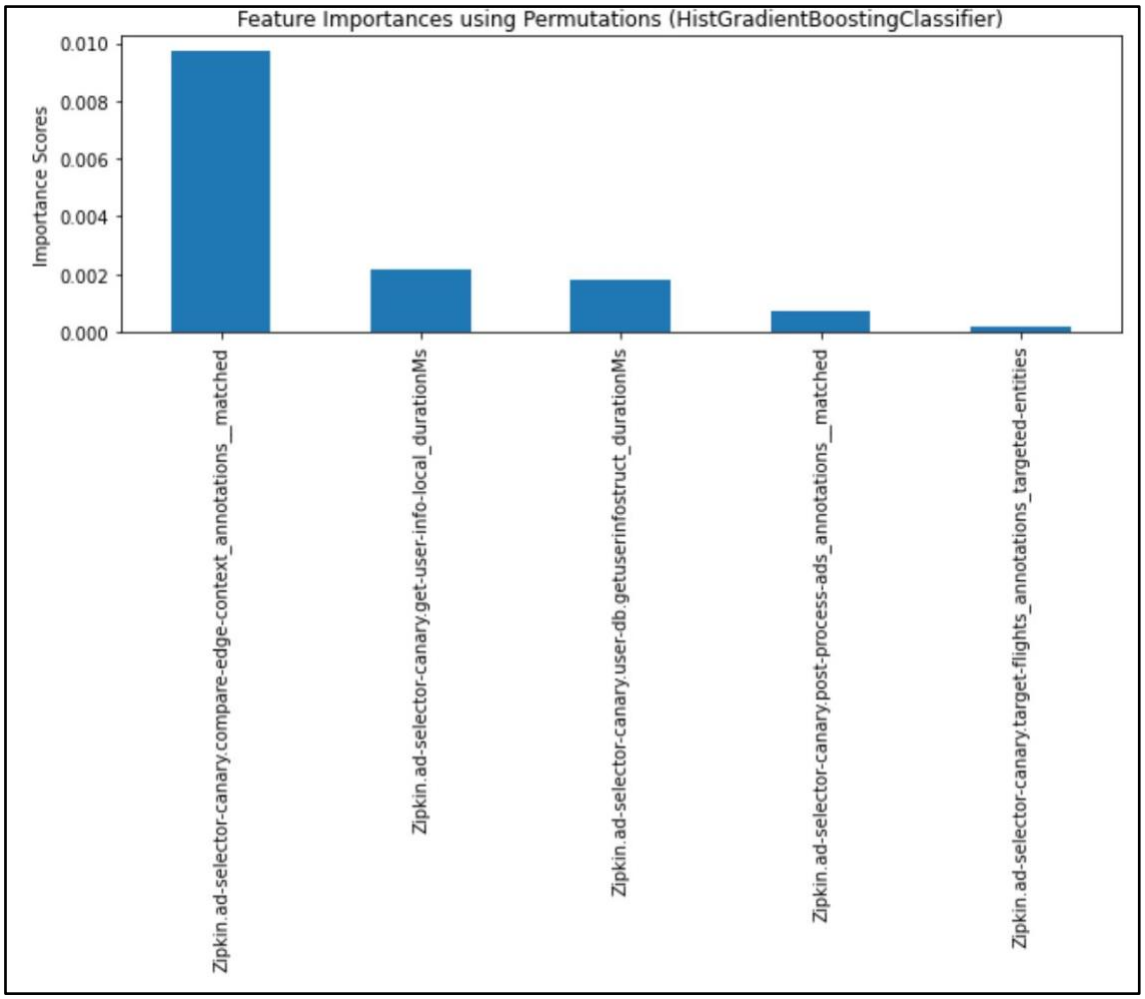


Figure 21. Permutation-based importance analysis for Dataframe3 for the optimal model for HistGradientBoostingClassifier.

Fig. 22 reveals the results of SHAP. The first important feature is the same as in Fig. 21. However, SHAP provides with more detailed information. The second, third and fourth ones also can be found in Fig.21.

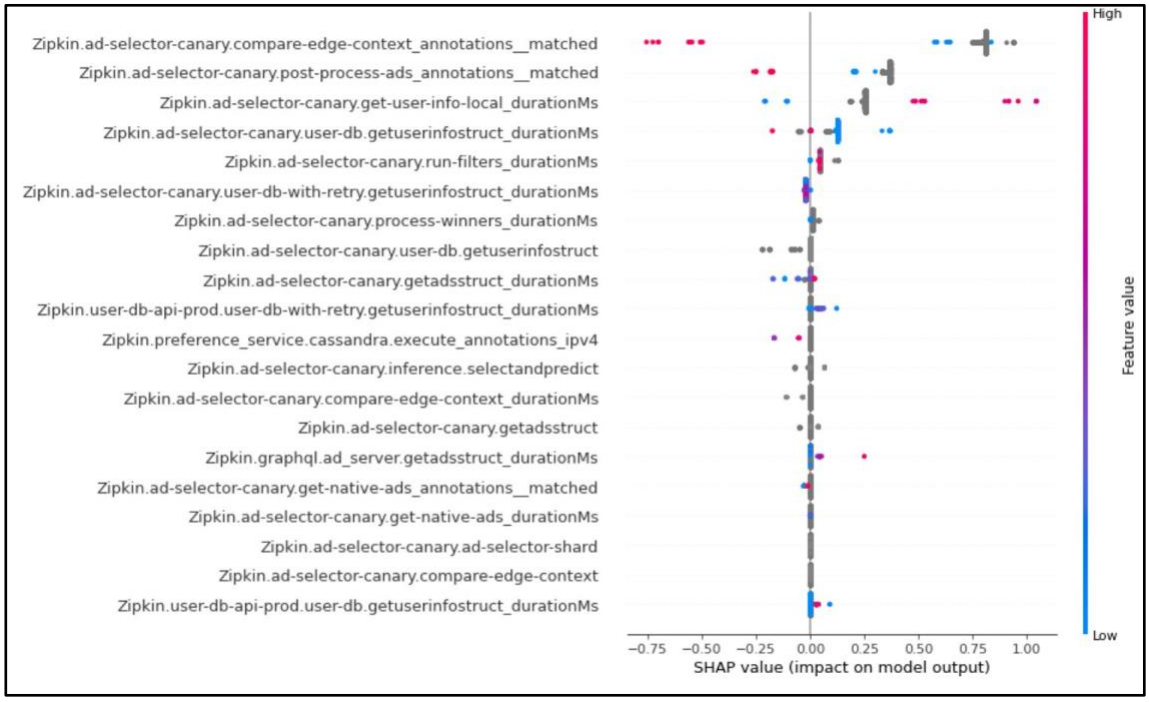


Figure 22. Importance analysis by SHAP for Dataframe3 applied to the optimal model for HistGradientBoostingClassifier. The analysis is performed only for the positive class.

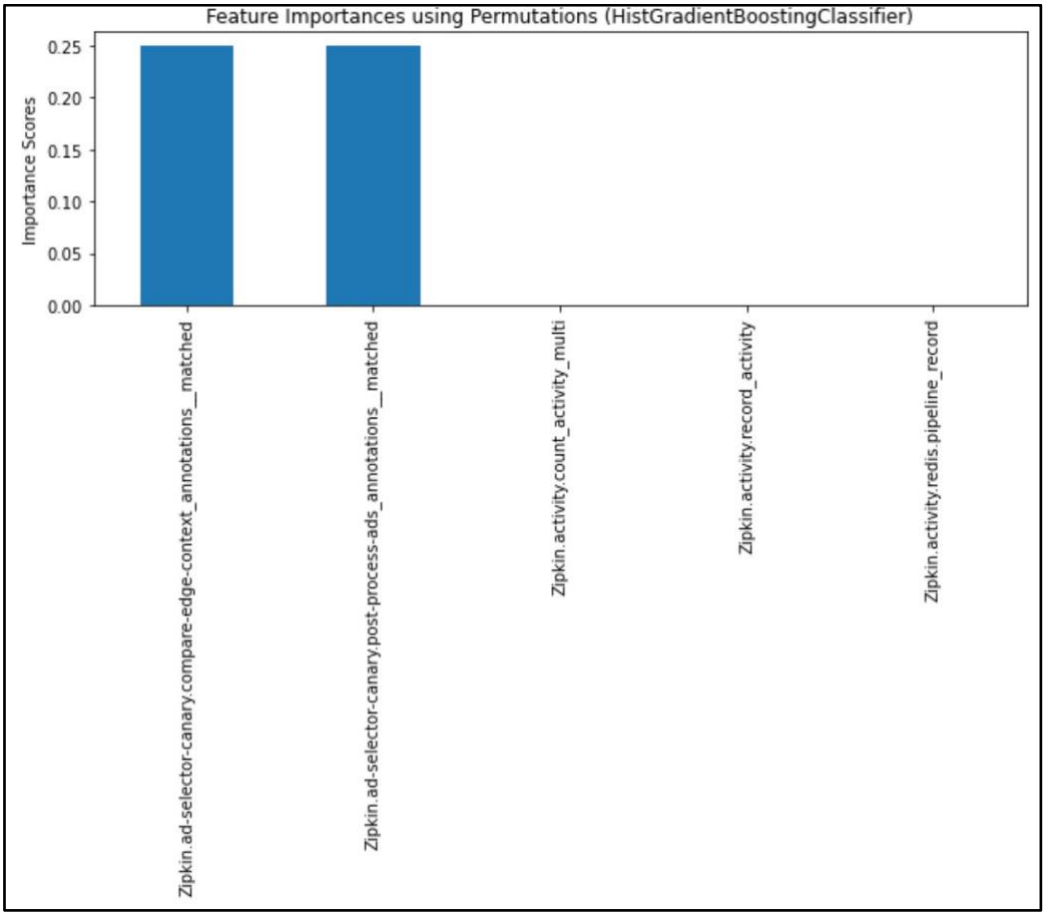


Figure 23. Permutation-based importance analysis for Dataframe3 for the optimal model for HistGradientBoostingClassifier after removing the tags containing the string "durationMs".

As we mentioned above, it is possible that durations are not so helpful. Degraded microservices have shorter or longer durations. We need more detailed information regarding the underlying processes. We remove all durations and reapply the analysis. Fig. 23 shows the result of permutation-based analysis after removing the durations. The most important one is "Zipkin.ad-selector-canary.compare-edge-context_annotations__matched" which appeared above.

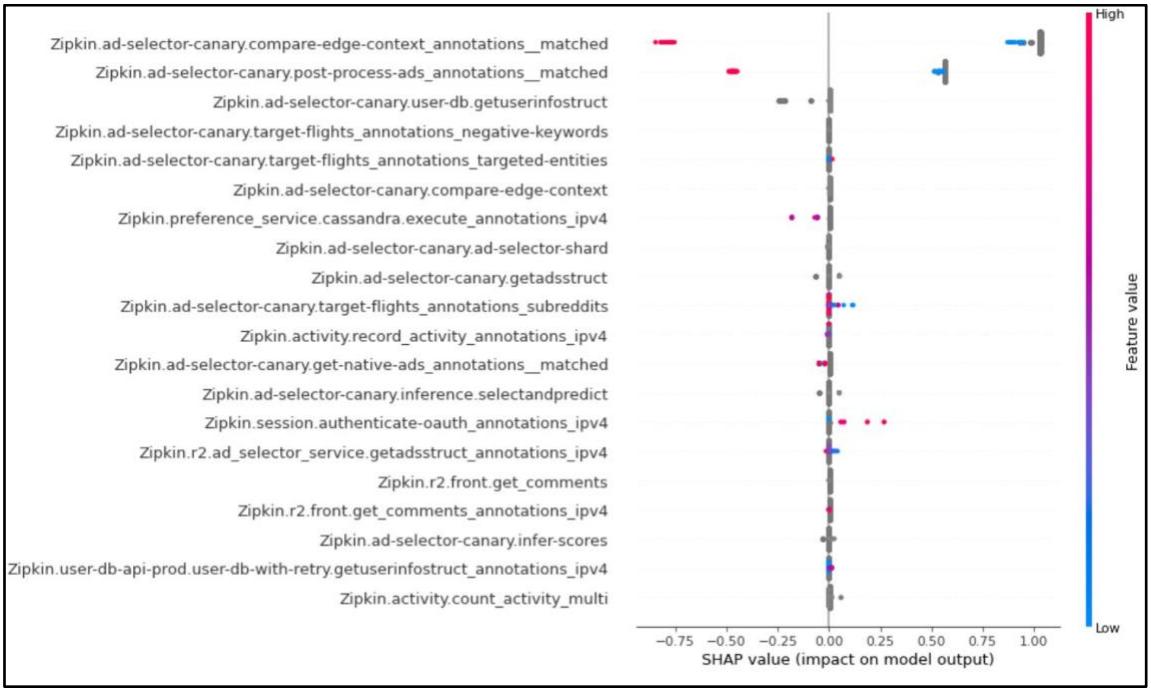


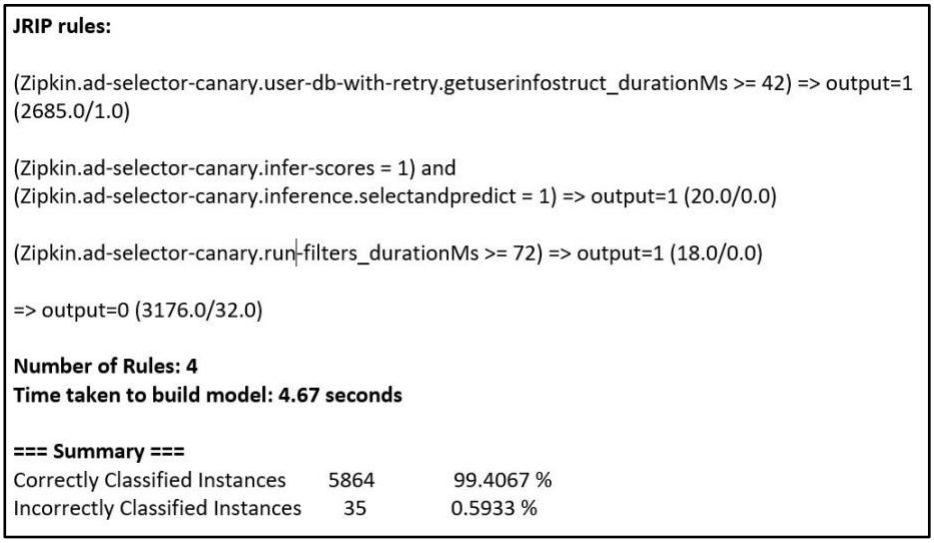
Figure 24. SHAP analysis for Dataframe3 for the optimal model for HistGradientBoostingClassifier after removing all features that contain "durationMs".

Fig. 24 shows the result of application SHAP after removing all features that contain the string "durationMs". The two top features coincide with the two top ones in Fig. 23. SHAP explains that the lowest values (blue ones) are the most important for those two features. They contain three different values: {nan, False, True}. We applied OrdinalEncoder() and transformed them into: {nan, 0, 1}. It means that the value = False relates to a problem. If this value is also not so much indicative, we can remove it and reapply the procedure. Ideally, this feature engineering part should be done before the analysis by domain experts who have better understanding which spans and tags can help to explain trace problems.

3.4. Dataframe4

We repeat the same analysis as for Dataframe3. Figs. 25 and 26 shows the results corresponding to RIPPER. The results of Fig. 26 are derived after removing the fields

containing the string “durationMs”. RIPPER shows excellent results in both cases. Both figures show “strong” rules in terms of the recall and precision.



JRIP rules:

(Zipkin.ad-selector-canary.user-db-with-retry.getuserinfostruct_durationMs >= 42) => output=1 (2685.0/1.0)

(Zipkin.ad-selector-canary.infer-scores = 1) and
(Zipkin.ad-selector-canary.inference.selectandpredict = 1) => output=1 (20.0/0.0)

(Zipkin.ad-selector-canary.run-filters_durationMs >= 72) => output=1 (18.0/0.0)

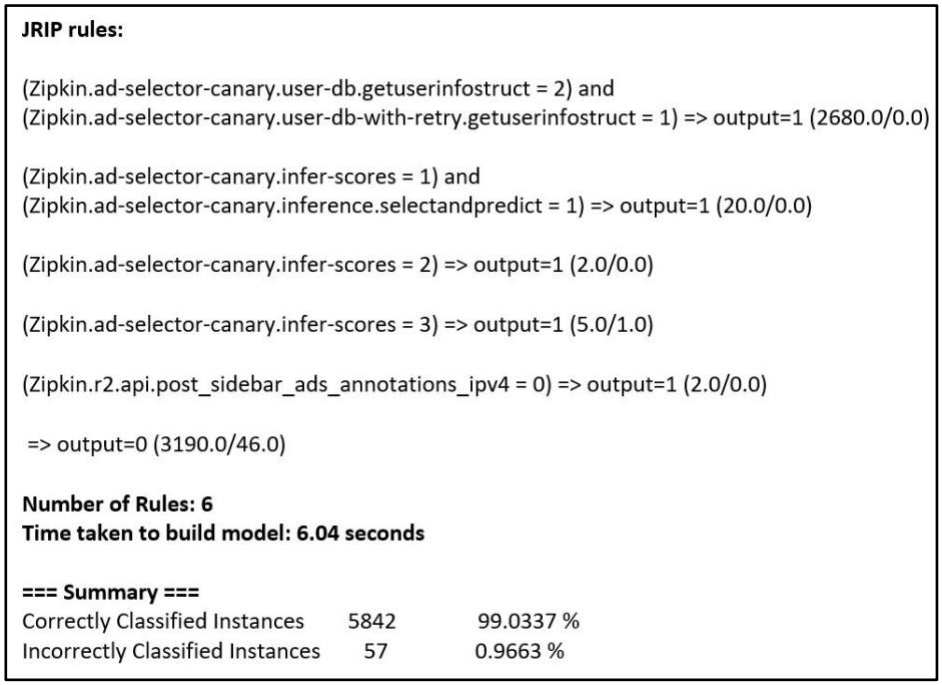
=> output=0 (3176.0/32.0)

Number of Rules: 4
Time taken to build model: 4.67 seconds

=== Summary ===

Correctly Classified Instances	5864	99.4067 %
Incorrectly Classified Instances	35	0.5933 %

Figure 25. RIPPER applied to Dataframe4.



JRIP rules:

(Zipkin.ad-selector-canary.user-db.getuserinfostruct = 2) and
(Zipkin.ad-selector-canary.user-db-with-retry.getuserinfostruct = 1) => output=1 (2680.0/0.0)

(Zipkin.ad-selector-canary.infer-scores = 1) and
(Zipkin.ad-selector-canary.inference.selectandpredict = 1) => output=1 (20.0/0.0)

(Zipkin.ad-selector-canary.infer-scores = 2) => output=1 (2.0/0.0)

(Zipkin.ad-selector-canary.infer-scores = 3) => output=1 (5.0/1.0)

(Zipkin.r2.api.post_sidebar_ads_annotations_ipv4 = 0) => output=1 (2.0/0.0)

=> output=0 (3190.0/46.0)

Number of Rules: 6
Time taken to build model: 6.04 seconds

=== Summary ===

Correctly Classified Instances	5842	99.0337 %
Incorrectly Classified Instances	57	0.9663 %

Figure 26. RIPPER applied to Dataframe4 after removing the fields containing the string “durationMs”.

Fig. 27 and 28 show the results obtained by the permutation-based analysis and SHAP respectively. They all indicate almost similar processes.

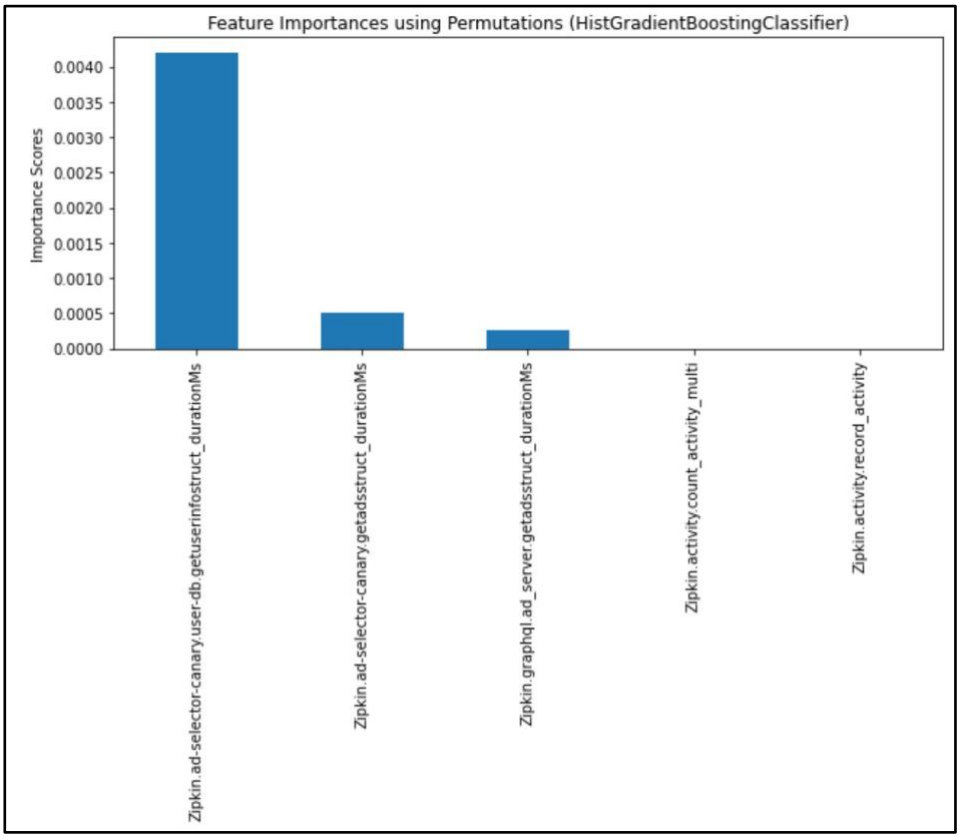


Figure 27. Permutation-based analysis for Dataframe4 based on tree-boosting.

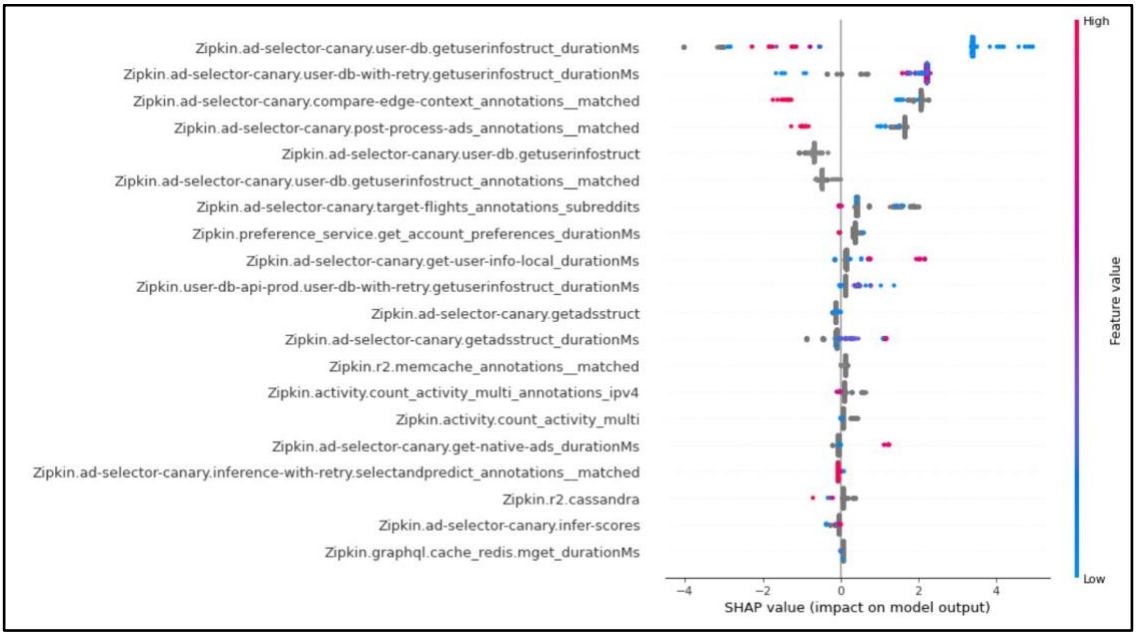


Figure 28. SHAP for Dataframe4 and tree-boosting.

We remove durations and reapply the analyses. The two top recommendations of Fig. 30 are the same as in Fig. 29.

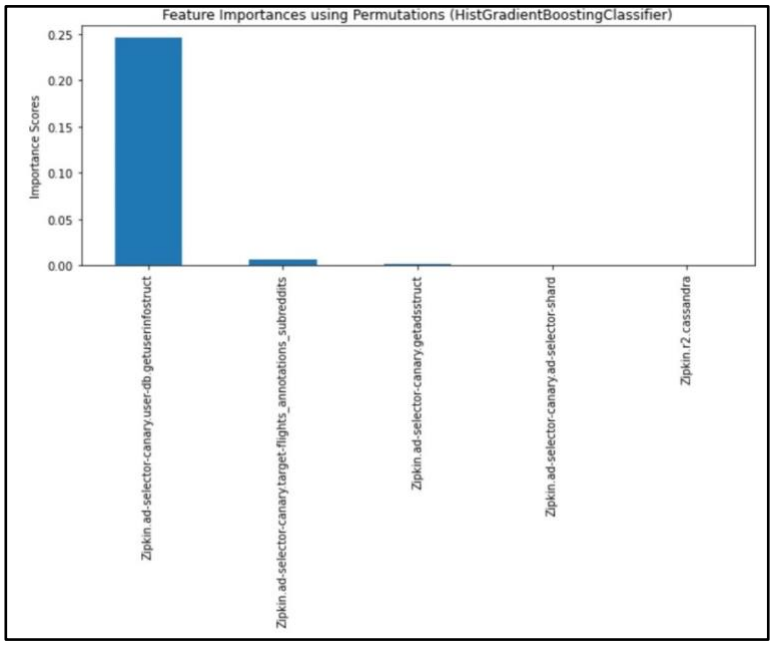


Figure 29. Permutation-based analysis for Dataframe4 based on tree-boosting after removing the durations.

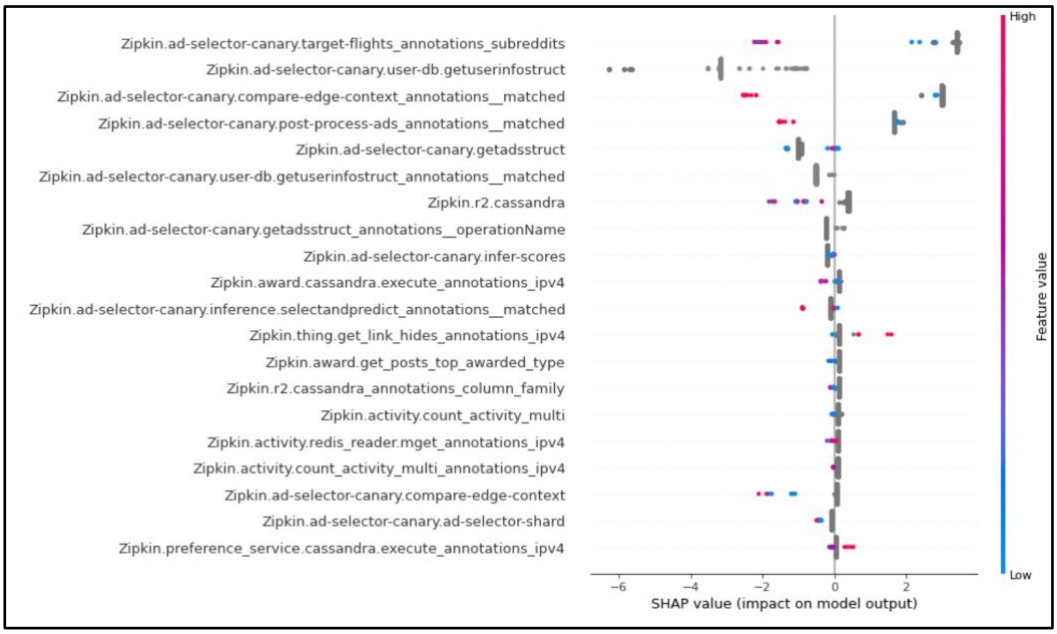


Figure 30. SHAP for Dataframe4 and tree-boosting after removing the durations.

Conclusion and Future Work

Application troubleshooting in case of performance degradations, otherwise known as root cause analysis (RCA), requires the highest degree of explainability from ML algorithms. There are several reasons for this requirement. In many industrial applications, data (process) understanding is much more important than the prediction of performance issues as the latest are already known in many situations. Better data understanding can reveal the underlying processes in complex modern applications and help system administrators accelerate the remediation process before they affect end-users. Conversely, explanations can increase the confidence in the recommendations and predictions, making AI solutions more sympathetic.

The explainability and predictive power of ML algorithms have opposite directions. Many powerful methods like neural networks are completely intransparent (black box) where the path of decision making is invisible. They make accurate predictions but fail in explanations. It means that they are totally useless for the root cause analysis. On the other side, highly explainable solutions derived from decision trees or rule induction methods suffer from inaccuracy. They are helpful only for specific applications where both solutions' explainability and predictive power coexist. Due to this the modern methods of XAI are becoming more popular as they allow us to acquire power and explanations simultaneously.

In this work, we experiment with RIPPER and tree-boosting based on distributed tracing for troubleshooting purposes. RIPPER is still state-of-the art in rule induction. Throughout the paper, we see how fast it is and what kind of rules it can provide. The rules contain the names of features and some threshold values. Violation of thresholds will fire the rules. They provide the complete explainability of how performance (health) degradations can arise. However, in some situations, they fail to return acceptable classification scores; even if they show some rules, we cannot confidently use them. It should be strange that trusted rules lead to wrong assignments. In those situations, we can rely on more powerful methods like tree-boosting.

In our application, the choice of possible ML approaches is severely limited. Our datasets contain a significant portion of missing values like 80%-90%. Imputation of those values is not feasible. Hence, we can only utilize algorithms that handle those missing values. Both RIPPER and tree-boosting have such capabilities. In the case of tree-boosting, we can apply built-in feature-importance analysis based on the mean decay in impurity (MDI) or apply model-agnostic permutation-based analysis.

Moreover, detailed information regarding the features and their values that lead to some label-assignments can be obtained via SHAP. Throughout the paper, we performed comparisons between different approaches. Our analysis showed that the recommendations derived from different sources can be used in conjunction for more transparency in many situations.

We also found the severe drawback of modern explainability ideas. They are extremely time-consuming and unrealistic to use in active dynamic environments where a timely reaction is the main requirement. Our main goal for the future is to develop implementations that will simplify the usage of the modern explainability approaches. Another goal is to develop/implement algorithms with sufficient predictive power capable of handling the missing values.

References

1. Gartner Research, Forecast: Public Cloud Services, Worldwide, 2020-2026, 1Q22 Update. Forecast: PCs, Worldwide, 2020-2026, 1Q22 Update (gartner.com).
2. Gartner Research, Market Guide for AIOps Platforms. Market Guide for AIOps Platforms (gartner.com)
3. Arrieta, A.B., Díaz-Rodríguez, N. et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion, vol. 58, pp. 82-115.
4. Adadi, A., Berrada, M., (2018). Peeking inside the black box: a survey on explainable artificial intelligence (XAI). IEEE Access, 6, pp. 52138-52160.
5. Defining the modern application. https://octo.vmware.com/defining-modern-application/
6. Dynatrace. What is observability? Not just logs, metrics and traces. https://www.dynatrace.com/news/blog/what-is-observability-2/
7. Real-Time Insights into Modern Applications. https://tanzu.vmware.com/content/vmware-tanzu-observability-solutions/real-time-insights-into-modern-applications
8. Distributed Tracing: A Complete Guide. https://lightstep.com/distributed-tracing
9. Distributed Tracing Overview. https://docs.wavefront.com/tracing_basics.html#distributed-tracing-videos
10. Lee, W., Stolfo, S.J. (1998). Data Mining Approaches for Intrusion Detection. In: Proceedings of the Seventh USENIX Security Symposium, 1998.
11. Fürnkranz, J., Kliegr, T. (2015). A Brief Overview of Rule Learning. Lecture Notes in Computer Science, v. 9202, pp. 54-69.
12. Cohen, W.W. (1995). Fast Effective Rule Induction. In: Proceedings of Twelfth International Conference on Machine Learning, pp. 115-123. Morgan Kaufmann.
13. Cohen, W.W. (1998). Rule Induction on Large Noisy Datasets, US Patent 5,719,692. Granted Feb 17.

14. Fürnkranz, J. (1997). Pruning Algorithms for Rule Learning. Machine Learing, v. 27, N2, pp. 139-171.
15. Furnkranz, J., Widmer, G. (1994). Incremental Reduced Error Pruning. In Machine Learning: Proceedings of the Eleventh Annual Conference, New Brunswick, New Jersey. Morgan Kaufmann.
16. J.R. Quinlan, J.R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco.
17. Hühn, J., Hüllermeier, E. (2009). An Algorithm for Unordered Fuzzy Rule Induction. Data Mining and Knowledge discovery, v. 19, N3, pp. 293-319.
18. Fürnkranz, J., Gamberger, D., Lavrac, N. (2012). Foundations of Rule Learning. Springer-Verlag, 2012.
19. Helmer, G., Wong, J.S.K., Honavar, V., Miller, L. (1998). Intelligent Agents for Intrusion Detection. In: Proceedings IEEE Information Technology Conference, Syracuse, NY, pp. 121-124.
20. Helmer, G., Wong, J.S.K., Honavar, V., Miller, L. (2002). Automated Discovery of Concise Predictive Rules for Intrusion Detection. The Journal of Systems and Software, v. 60, pp. 165-175.
21. Hägele, M., Seegerer, P., Lapuschkin, S., Bockmayr, M., Samek, W., Klauschen, F., Müller, K.-R., Binder, A. (2020). Resolving Challenges in Deep Learning-Based Analyses of Histopathological Images using Explanation Methods, Scientific Reports, Nature Research, vol. 10, Springer Nature, Article number: 6423, DOI: 10.1038/s41598-020-62724-2.
22. Horst, F., Lapuschkin, S., Samek, W., Müller, K.-R., Schöllhorn, W.I. (2019). Explaining the Unique Nature of Individual Gait Patterns with Deep Learning, Scientific Reports, Nature Research, vol. 9, London, UK, Springer Nature, Article number: 2391, DOI: 10.1038/s41598-019-38748-8.
23. Thomas, A.W., Heekeren, H.R., Müller, K.-R., Samek, W. (2019). Analyzing Neuroimaging Data Through Recurrent Deep Learning Models, Frontiers in Neuroscience, vol. 13, Article number: 1321, DOI: 10.3389/fnins.2019.01321.

24. Aeles, J., Horst, F., Lapuschkin, S., Lacourpaille, L., Hug, F. (2021). Revealing the unique features of each individual's muscle activation signatures, Journal of The Royal Society Interface, vol. 18, no. 174, The Royal Society, p. 20200770, ISSN: 1742-5662, DOI: https://doi.org/10.1098/rsif.2020.0770.
25. Sturm, I., Lapuschkin, S., Samek, W., Müller, K.-R. (2016). Interpretable Deep Neural Networks for Single-Trial EEG Classification, Journal of Neuroscience Methods, Elsevier Inc., vol. 274, pp. 141-145, DOI: 10.1016/j.jneumeth.2016.10.008.
26. Permutation feature importance. https://scikit-learn.org/stable/modules/permutation_importance.html.
27. Breiman, L., (2001). Random forests. Machine learning, 45(1), 5-32.
28. Främling, K., Westberg, M., Jullum, M., Madhikermi, M., Malhi, A., (2021). Comparison of Contextual Importance and Utility with LIME and Shapley Values. In International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems (pp. 39-54). Springer, Cham.
29. Dieber, J., Kirrane, S., (2020). Why model why? Assessing the strengths and limitations of LIME. ArXiv:2012.00093.
30. Saarela, M., Jauhiainen, S., (2021). Comparison of feature importance measures as explanations for classification models. SN Appl. Sci. 3, 272.
31. Elshawi, R., Sherif, Y., Al-Mallah, M., Sakr, S., (2021). Interpretability in healthcare: A comparative study of local machine learning interpretability techniques. Computational Intelligence, 37(4), 1633-1650.
32. Visani, G., Bagli, E., Chesani, F., Poluzzi, A., Capuzzo, D., (2022). Statistical stability indices for LIME: Obtaining reliable explanations for machine learning models. Journal of the Operational Research Society, 73(1), 91-101.
33. Ribeiro, M.T., Singh, S., Guestrin, C., (2016). Model-agnostic interpretability of machine learning. arXiv:1606.05386.
34. Ghalebikesabi, S., Ter-Minassian, L., DiazOrdaz, K., Holmes, C. C., (2021). On locality of local explanation models. Advances in Neural Information Processing Systems, 34.

35. SHAP (SHapley Additive exPlanations), https://christophm.github.io/interpretable-ml-book/shap.html
36. Bowen, D., Ungar, L., (2020). Generalized SHAP: Generating multiple types of explanations in machine learning. arXiv:2006.07155.
37. Lundberg, S., Lee, S.-I. (2017). A unified approach to interpreting model predictions. arXiv: 1705.07874.
38. Apdex function. https://docs.wavefront.com/hs_apdex_function.html
39. Schapire, R., (1999). A brief introduction to boosting. In IJCAI, 2, 1401-1406.
40. Baba, N., Makhtar, M., Fadzli, S., Awang, K., (2015). Current issues in ensemble methods and its applications. Journal of Theoretical and Applied Information Technology, 81(2), 266-276.
41. Rokach, L., (2005). Ensemble methods for classifiers. In Data mining and knowledge discovery handbook (957-980). Springer, Boston, MA.
42. XGBoost documentation. https://xgboost.readthedocs.io/en/stable/index.html
43. HistGradientBoostingClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html
44. Permutation feature Importance. https://scikit-learn.org/stable/modules/permutation_importance.html
45. Roweis, S., Hinton, G. (2002). Stochastic neighbor embedding, Neural Information Processing Systems.
46. van der Maaten, L.J.P., Hinton, G.E. (2008). Visualizing Data Using t-SNE , Journal of Machine Learning Research. Vol. 9, pp. 2579–2605.
47. van der Maaten, L.J.P., Hinton, G.E. (2008). Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9, 2579-2605.
48. van der Maaten, L.J.P. t-Distributed Stochastic Neighbor Embedding https://lvdmaaten.github.io/tsne/

49. van der Maaten, L.J.P. (2014). Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15, pp. 3221-3245. https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf
50. Belkina, A. C., Ciccolella, C. O., Anno, R., Halpert, R., Spidlen, J., Snyder-Cappione, J. E. (2019). Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. Nature Communications, 10(1), 1-12.
51. Kobak, D., Berens, P. (2019). The art of using t-SNE for single-cell transcriptomics. Nature Communications, 10(1), 1-14.
52. Nag, D.A., Grigoryan, N.M., Poghosyan, A.V., Harutyunyan, A.N. (2021). Methods and Systems that Identify Dimensions Related to Anomalies in System Components of Distributed Computer Systems Using Traces, Metrics, and Component-associated Attribute Values. Filed: Mar 27, 2020. Application No: US 16/833,102. Published: Mar 27, 2020. Publication No: US 2020/0264965A1. Granted: Sep 7, 2021. Patent No: US11113174.
53. Poghosyan, A., Harutyunyan, A., Grigoryan, N., Pang, C., Oganesyan, G., Baghdasaryan, D. (2021). Automated methods and systems that facilitate root cause analysis of distributed-application operational problems and failures. US patent application No.: 17/491,967, date filed: 2021-10-01.
54. Poghosyan, A., Harutyunyan, A., Grigoryan, N., Pang, C., Oganesyan, G., Baghdasaryan, D. (2021). Automated methods and systems that facilitate root-cause analysis of distributed-application operational problems and failures by generating noise-subtracted call-trace-classification rules. US patent application No.: 17/492,099, date filed: 2021-10-01.