Towards a practical process model for Anomaly Detection Systems

Nils Schwenzfeier
paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen
nils.schwenzfeier@paluno.uni-due.de

Volker Gruhn
paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen
volker.gruhn@paluno.uni-due.de

ABSTRACT

Process models are an important tool for software engineers to produce reliable software within schedule and budget. Especially technically challenging domains like machine learning need a supportive process model to guide the developers and stakeholders during the development process. One major problem type of machine learning is anomaly detection. Its goal is to identify anomalous data points (outlier) between the normal data instances. Anomaly detection has a wide scope of applications in industrial and scientific areas. Detecting intruders in computer networks, distinguishing between cancerous and healthy tissue in medical images, cleaning data from disturbing outliers for further evaluation and many more. The cross-industry standard process for data mining (CRISP-DM) has been developed to support developers with all kinds of data mining applications. It describes a generic model of six phases that covers the whole development cycle. The generality of the CRISP-DM model is as much a strength as it is a weakness, since the particularities of different problem types like anomaly detection can not be addressed without making the model overly complex. There is a need for a more practical, specialised process model for anomaly detection applications. We demonstrate this issue and outline an approach towards a practical process model tailored to the development of anomaly detection systems.

CCS CONCEPTS

• **Software and its engineering** → *Software development process management*; Software development methods;

KEYWORDS

Process model, Anomaly detection, Machine Learning, Data science, CRISP-DM

ACM Reference Format:
Nils Schwenzfeier and Volker Gruhn. 2018. Towards a practical process model for Anomaly Detection Systems. In *SE4COG'18: SE4COG'18:IEEE/ACM 1st International Workshop on Software Engineering for Cognitive Services , May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.1145/3195555.3195568

SE4COG'18, May 28–29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5740-1/18/05…$15.00
https://doi.org/10.1145/3195555.3195568

1 INTRODUCTION

An anomaly is a rare event, instance or data point that differs sufficiently from the rest of the data to stand out [1]. This instance has to be of special interest to the observer, which distinguishes an anomaly from noise in the data. Anomaly detection utilises various statistical techniques and machine learning algorithms to distinguish between anomalous and normal instances in data. Therefore, it has become one major problem type of machine learning beside classification, regression, recommender systems and clustering. The common imbalance between the anomalous and normal class raises many challenges in designing, implementing and evaluating anomaly detection systems. Hence, classification and anomaly detection are usually treated as different problems. In some cases, there are no labeled anomalous data instances at all. Clustering algorithms and extreme value analysis tools are used in such cases to isolate the normal class from outliers. Anomaly detection techniques are applied in all kinds of application domains. They are used to discover intruders in a computer network, detect credit card or insurance fraud, finding outliers in sensor data, predict faults in machines even before they happen (predictive maintenance) or distinguish between healthy and cancerous tissue in medical images [2].

Process models have a long history in the software engineering discipline [10]. They are used to facilitate and standardise the development process of software projects. A process model consists of several phases that describe the different development steps. Within a phase, several tasks and activities describe what needs to be done during the phase and provide guidelines how to perform these actions. Artefacts like a documentation, implemented code or generated test cases can be specified as in- and outputs of tasks. At the early stages of software development, many projects were characterised by missed schedules and budget overruns. Development processes help to control the development progress to meet project deadlines reliable and comply with budgets. Furthermore, they increase the overall product quality by enforcing regular quality checks during the process.

The machine learning community has developed various algorithms to achieve great results on many applications only humans were able to do for a long time. These algorithms are used by the knowledge discovery and data mining communities to gain knowledge from data and develop new applications for all kinds of domains. Automatic traffic sign recognition, speech recognition and synthesis, object detection and the classification between benign and malignant birthmarks are only a few examples of the wide range of applications. All these applications heavily depend on the quality of available training data. Nevertheless, a development process model is necessary to guide the actions of data scientists and software developers.

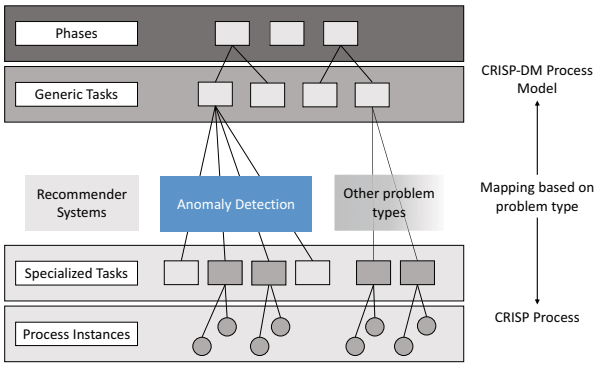

Figure 1: The hierarchical CRISP-DM process model. The mapping based on problem types introduces another layer between generic and specialised tasks. Figure is inspired by [4]

Section 2 presents the short comings of generic data mining process models. In particular, the benefits and flaws of the cross-industry process for data mining (CRISP-DM) process model are analysed. Section 3 introduces our approach to develop a more practical process model for the development of anomaly detection systems. Further developments of the CRISP-DM process model, as well as an overview of other process models of the knowledge discovery and data mining community are briefly presented in section 4. Section 5 outlines future work and concludes this paper.

2 THE DEVELOPMENT PROCESS OF MACHINE LEARNING APPLICATIONS

Today, the cross-industry standard process for data mining (CRISP-DM) is a widespread process model for all kinds of data mining and machine learning applications. Its goal is to provide a development guideline for all kinds of data driven applications [4]. This has been realised by a cyclic model containing the phases "Business Understanding", "Data Understanding", "Data Preparation", "Modelling", "Evaluation" and "Deployment". The model is refined in several hierarchy levels. Each phase is broken down into several tasks. A task is split into multiple activities that describe what needs to be done during the task. While this model divides the tasks in detail, it does not explain how to realise the activities. For instance, the phase "Modelling" contains a task "Assess Model", which contains the activity "Evaluate results with respect to evaluation criteria" [4]. It remains up to the developer which evaluation criteria make sense for a given application context. This approach is understandable, since the model's target is to be generally applicable. On the other hand, this limits the usability of the model for the developer. The wrong evaluation criteria can lead to a misleading interpretation of the machine learning model results. This can decide on the success of the whole project. To give the developers the best possible support, many companies refine common process models using their experience from successful projects. Unfortunately, this knowledge usually does not get to other developers. An open, detailed process model is needed to improve the development process for machine learning applications. The more detailed a model is, the greater the risk that the model becomes overly complex. A model whose use is complicated and not intuitive anymore will not be accepted by software developers. Therefore, it is necessary to limit the scope of application of such a process model in order to reach a more detailed description of the necessary tasks and activities.

Anomaly detection is one prominent type of machine learning applications. Various machine learning algorithms have been modified to be able to distinguish between normal and anomalous data instances. Even though anomaly detection has been studied for decades, there is no common process model to facilitate the development of these systems. A lot of research is dedicated to develop new or enhance existing algorithms. In addition, there are many publications that deal with the application of an anomaly detection algorithm on a specific problem. There is a gap between these two extremes that we address now.

While there are clearly individual project related obstacles and tasks for each anomaly detection problem that originate from the application domain, there are also many similarities. The requirements on an anomaly detection system are often very similar though the definition of an anomaly depends on the application. Consequently, the same questions have to be answered for most anomaly detection projects. This includes the definition of anomalous and normal in relation to the data, the required tradeoff between sensitivity and specificity, what amount of data has to be processed by the system in which time and many more. Usually, the used algorithms do not depend on the application domain but on the texture of the data, as well as further requirements like the level of model interpretation that is needed. For the selection of a suitable algorithms the anomaly type (point, contextual or collective anomaly) is important. Test cases can often be derived from the applied algorithms. Since all kind of data instances are mapped to be either anomalous or normal, one can test with these two classes instead of examining the application data itself. Of course, the origin of certain test results has to be interpreted using the original data. The deployment also raises similar questions for all kinds of anomaly detection systems. What has to be done with detected anomalies? How can user feedback be used to decrease the rate of false positives? How can one be sure that no anomalies have been overlooked? Many of these properties of anomaly detection systems are not shared by other applications.

The CRISP-DM process model itself is not detailed enough to cover the questions that are raised by the anomaly detection context. An example of the CRISP-DM task "Assess model" demonstrates the lack of practical assistance this process model can offer. During this task the activity "evaluate with evaluation criteria" should be performed. There are no further information given what the evaluation criteria should or might be. Furthermore, it is not clear how the evaluation process should be conducted. From the perspective of the CRISP-DM developers, this makes sense, since they claimed to develop a generic standard process model for all kinds of data mining applications. The authors of the CRISP-DM process model encourage developers to refine their generic model by adjusting the tasks based on the application context. We follow this call and argue that a process model on the problem domain level can provide much richer support for the developers. In our opinion, this will reduce the uncertainty during the development of anomaly detection applications leading to a more stable development process and predictable results.

3 A PRACTICAL PROCESS MODEL FOR ANOMALY DETECTION SYSTEMS

We develop a practical process model to support developers during the engineering of anomaly detection systems (ADS). This model will refine the CRISP-DM process model by mapping general tasks and activities to specific recommendations for action for the development of ADS. While our model clarifies how steps in the generic CRISP-DM process model can be mapped to a tailored process model for anomaly detection settings, it is still independent from the application domain. This is possible through the abstraction of concrete problems to a set of common anomaly detection tasks. The detection of cancerous tissue in magnetic resonance tomography (MRT) images and the detection of forest fires in satellite images can be tackled by the same algorithms. Both problems can be generalised to the problem of finding contextual (spatial) anomalies in images. A taxonomy of algorithms can guide developers to find the best suitable methods quickly. By separating technical characteristics and activities of anomaly detection problems from the application domain specific questions, the development of an engineering process for ADS, that focuses on the details of the technical aspects of anomaly detection problems, becomes possible.

While the CRISP-DM model clarifies how tasks should be performed in form of generic actions only, our model offers concrete recommendations how certain actions can be performed in an anomaly detection setting. This is done by presenting common strategies for all kinds of anomaly detection problem types including the state of the art algorithms, evaluation metrics and visualisation techniques, common pitfalls and guidelines for the transformation of prototypes towards real applications. During the modelling phase the CRISP-DM model asks to generate a test design to evaluate the model performance. Beside mentioning that it is required to split data into a training-, validation- and test-set the model leaves it to the developer how an extensive test suite should look like. Even the distribution of data into a training- and a test-set is not obvious in the case of anomaly detection systems when there are no labeled anomalous data instances. The performance of a classification model is often measured by its accuracy. While this is an easily understandable metric for all stakeholders, it is usually not suitable for anomaly detection problems. Due to the huge imbalance between the anomalous and normal class, the accuracy value misses critical properties of the model like the number of unidentified anomalies. A mixture of sensitivity and specificity like in the receiver operating characteristic (ROC) curve provides a more comprehensive evaluation of the performance of an anomaly detection system. Since the best evaluation metric depends on the problem domain, as well as on the algorithms used, it is comprehensibly that a generic model is not able to provide this kind of details without becoming overly complex. We believe that only a detailed process model provides the level of support that is needed to ensure a reliable development process that is able to guarantee a constantly high software quality. The complexity of such a detailed model can be controlled by restricting its scope of application to a single problem domain. In the case of generating a test suite, the model can address problems like the typically missing knowledge about all kinds of anomalies in a given setting. Obviously, in this case it is not possible to achieve a complete test coverage. Instead, the definition of a normal instance has to be analysed intensively to be able to make statements on the assumptions of what kind of patterns are assumed to be normal by the model. Anomaly detection techniques like the single class support vector machine (SC-SVM) or clustering algorithms with specialised metrics can be used to tackle this problem. Since the process model is limited to ADS, it is possible to provide concrete guidelines how to perform this type of analysis for a set of representational anomaly detection techniques. For the practical use of an anomaly detection system the sensitivity-specificity tradeoff plays a major role. This has to be considered during the requirements engineering/business understanding phase of the development cycle.

Of course, it is not possible to treat all kinds of anomaly detection problems equally. There are different classes of ADS that differ in their type, mode, notion of an anomaly, nature of data and existing techniques. Anomaly detection techniques exists for supervised (data instances are labeled as normal or anomalous), semisupervised (only normal data instances are labeled) and unsupervised (no labeled data) settings. The literature distinguishes three types of anomalies, these are point, contextual and collective anomalies. Point anomalies can be identified based on conspicuous values of their attributes compared to normal data instances. On the other hand, contextual anomalies may have inconspicuous attribute values compared to other data instances, but their values are anomalous for their context. Collective anomalies detect anomalous sequences of data instances. In this case no instance is anomalous on its own. An example for this is an intrusion detection system where system calls are tracked. While each system call might be executed in a normal workflow, the combination of certain calls can indicate an attack on the system. It might appear natural to distinguish anomaly detection systems based on these three categories. However, looking at each category in detail opens up other possibilities to distinguish between different ADS. For instance, contextual anomalies can be further distinguished between temporal and spatial anomalies. Defining a certain context for each data instance reduces the contextual to a point anomaly detection problem. Finding an optimal separation of different paths through the anomaly detection development process is a challenge that we tackle.

Unlike many other data science process models, we focus especially on the conversion of a prototype into a stable system component. We believe that this is a critical part of the development cycle, as it often requires a detailed information exchange between software developer and data scientist. A clear documentation what information need to be communicated from the data scientist to the software developer is necessary to guarantee a successful model deployment. Assumptions that hold for a prototyping environment might not hold for the application context. Different needs for data pre-processing like data cleaning, fallback decisions for model failures and emerging anomalies have to be addressed. As one step towards a better communication between data scientist and software engineer we include a common glossary in our process model to facilitate a joint discussion.

Another interesting possibility we would like to investigate during the development of our process model is the tuning of models based on a problem domain. Many machine learning algorithms have a large set of hyper parameters that need to be tuned for each

application. For a random forest that is used for classification between multiple classes the number of trees, the height of each tree, the weighting of the different trees, the number of leaves in each tree and many more attributes have to be determined. This kind of model can also be used to classify between normal and anomalous instances in a supervised setting. The detection of anomalies brings new properties with it that need to be noted during model training. Typically, there are far less labeled anomalies than normal data instances. This causes new difficulties a developer should be aware of and that need to be overcome. We investigate if there are model configurations that already take care of anomaly detection related difficulties and can be used by developers as a starting point for further individual parameter tuning.

We plan to evaluate our process model from various perspectives. This includes case studies and experience reports from projects with industrial partners to prove the applicability of the process itself. In addition, the benefit of our concrete descriptions about how to perform the required activities in an anomaly detection setting will be investigated. Studies with software engineers are used to show the effect of a practical problem based process model compared to generic process models. For that, the developer's performances for different tasks of the anomaly detection development process are compared.

4 RELATED WORK

The cross-industry standard process for data mining (CRISP-DM) is the standard industry process model for data mining tasks [8]. CRISP-DM was conceived to establish a standard generic process model for data mining applications. Three years later, it was first published at the 4th CRISP-DM SIG workshop in Brussels in 1999 [3]. CRISP-DM is a hierarchical process model with 4 layers (s. figure 1). The first two layers describe the phases (layer 1) of the model and the associated generic tasks for each phase (layer 2). From that, specialised tasks (layer 3) have to be derived from the generic tasks by the model's user. Any combination of specialised tasks with a set order forms a process instance (layer 4). The transition between the upper two layers (CRISP-DM process model) to the lower two layers (CRISP-DM process) is called a mapping. Only layer 1 and 2 are described in detail in the CRISP-DM methodology. Each task is broken down into several activities that explain how to perform this task. Further information on how to realise an activity are not given. A process cycle of the CRISP-DM reference model shows the order in which the different process phases are passed. CRISP-DM distinguishes between mappings for a single usage, that is the creation of a process for a concrete project, and mappings for specialised process models for certain application or technical contexts.

Marbán et al. recognise parallel in the history of software engineering and data mining process development. They compare the CRISP-DM methodology with the two most used software engineering processes ISO 12207 and IEEE 1074 in their work "Toward data mining engineering: A software engineering approach" [9]. They discover that CRISP-DM focuses on the execution of the project plan but lacks project management and organisational processes. This includes checks for the project completeness and quality, as well as an enhancement process for a more effective organisation.

The knowledge discovery and data mining (KDDM) community has also developed process models for data mining at about the same time the CRISP-DM model was created. Fayyad et al. published the first KDDM process model in 1996 [6]. It consists of nine phases, that can be mapped to the six phases of the CRISP-DM model [7]. Cios et al. developed a six phase process model similar to the CRISP-DM model [5]. In contrast to the CRISP-DM process model, they suggest a concrete set of techniques to accomplish the various data mining tasks.

5 CONCLUSION

We motivated the use of process models during the development of anomaly detection systems. The state of the art process model for data mining applications CRISP-DM lacks the ability to provide concrete support on how to perform required activities due to its universal approach. We presented our vision of a specialisation of the CRISP-DM process model for the development of anomaly detection systems. Furthermore, we demonstrated the benefits of a practical process model that is based on a machine learning problem type. The support for the realisation of activities based on best practices and concrete guidelines, as well as the precise description of artefacts are the biggest advantages over generic approaches.

In our future work we plan to document and present our process model in detail and evaluate it with multiple studies. One focus will be on the performance of novice developers, because they benefit the most from a practical process model. Another interesting direction would be to examine other data mining problem types like recommender systems or reinforcement learning methods for their differences in the development process.

6 ACKNOWLEDGEMENTS

The European Union supported this work through project CPS.HUB NRW, EFRE No. 0-4000-17.

REFERENCES

[1] Charu C Aggarwal. 2015. Outlier analysis. In *Data mining*. Springer.
[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 15.
[3] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. 1999. The CRISP-DM user guide. In *4th CRISP-DM SIG Workshop in Brussels in March*, Vol. 1999.
[4] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. 2000. CRISP-DM 1.0 Step-by-step data mining guide. (2000).
[5] Krzysztof J Cios and Lukasz A Kurgan. 2005. Trends in data mining and knowledge discovery. In *Advanced techniques in knowledge discovery and data mining*. Springer, 1–26.
[6] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. 1996. The KDD process for extracting useful knowledge from volumes of data. *Commun. ACM* 39, 11 (1996), 27–34.
[7] Lukasz A Kurgan and Petr Musilek. 2006. A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review* 21, 1 (2006), 1–24.
[8] Óscar Marbán, Gonzalo Mariscal, and Javier Segovia. 2009. A data mining & knowledge discovery process model. In *Data Mining and Knowledge Discovery in Real Life Applications*. InTech.
[9] Oscar Marbán, Javier Segovia, Ernestina Menasalvas, and Covadonga Fernández-Baizán. 2009. Toward data mining engineering: A software engineering approach. *Information systems* 34, 1 (2009), 87–107.
[10] Leon Osterweil. 1987. Software processes are software too. In *Proceedings of the 9th international conference on Software Engineering*. IEEE Computer Society Press, 2–13.