Application of Software Engineering Concepts in Conjunction with Explainable Machine Learning and Conformal Prediction

Student: Amr Alkhatib
University: KTH Royal Institute of Technology

1 Introduction

Machine learning algorithms have made significant progress in real-world applications, particularly in fields like medicine, biology, and chemistry. However, these algorithms often produce black-box models, which raise issues related to legality, ethics, and trustworthiness [5, 6]. Black-box models, which lack transparency in their decision-making, pose several challenges:

- **Lack of Trust:** Transparency is crucial for users to trust machine learning models. With an understanding of how a model arrives at its decisions, users may be able to rely on it [6]. This trust issue is especially critical in many domains, e.g., healthcare, finance, and law, where decisions can have profound consequences.
- **Accountability Problems:** In regulated industries like finance and healthcare, there are legal and ethical requirements for models to explain their decisions. This transparency ensures that decisions are fair and unbiased [5].
- **Bias and Fairness Problems:** Machine learning models can inherit biases from their training data, leading to discriminatory decisions [9]. Explainability can help identify and mitigate these biases by revealing which features influence predictions.
- **Robustness and Safety Problems:** Explainability aids in identifying a model's vulnerabilities, making it possible to improve its robustness. This is particularly important in safety-critical applications like autonomous vehicles and medical diagnosis.

To address these challenges, interpretable white-box models like decision trees and linear models can be used, but they often sacrifice predictive performance [11, 10]. This has led to the rise of explainable machine learning (XAI) methods that aim to provide transparency without compromising performance. Explanation techniques can be categorized as model-agnostic or model-specific, as well as local or global in scope. Model-Agnostic methods include Local Interpretable Model-agnostic Explanations (LIME) [13] and Shapley additive explanations (SHAP) [8]. They offer explanations for any black-box model without requiring prior knowledge of the algorithm or training data. Model-Specific methods target specific models and leverage their properties to generate explanations. Local methods explain specific predictions, while global techniques provide an overview of how the model behaves in general.

Explanations can take various forms, such as intuitive plots like Partial Dependence Plots (PDP) [2] and Accumulated Local Effects (ALE) [1] Plots, additive feature importance scores like those produced by LIME and SHAP, or rule-based explanations like Anchors [14]. However, using machine learning explanation methods presents its own set of challenges. Fidelity, which measures how accurately an explanation represents the underlying model's behavior, is a significant concern. Some methods produce explanations in the form of feature importance scores, which lack objective verification. Rule-based techniques offer more verifiable explanations but can have low fidelity due to complex underlying models.

In our work, we aim to use the conformal prediction framework [3, 15] to quantify the uncertainty of explanations, allowing for different explanation techniques to be evaluated with respect to their relative uncertainty. Develop computationally efficient explanation methods suitable for use in scenarios where timely explanations are necessary. We also investigate using the conformal prediction framework to provide explanations with predefined validity guarantees.

2 Software Design and Quality in XAI and Conformal Prediction

In this section of the essay we will explore how software design and software quality relate to Explainable Machine Learning (XAI) and Conformal Prediction.

Software quality is a concept that encloses reliability, maintainability, and scalability. In XAI, ensuring model quality is essential. A model may be interpretable and transparent but must also be reliable and accurate. Software engineers can apply software testing methodologies to machine learning models used in XAI. This involves testing the model's behavior under various scenarios and conditions, including edge cases. Rigorous testing helps uncover potential issues and uncertainties in the model's decision-making process. Additionally, continuous monitoring and evaluation of the model's performance in real-world settings are crucial for maintaining its quality over time, which involves designing software components that can collect and analyze data, comparing model predictions with actual outcomes, and triggering alerts when the model's performance deviates significantly.

In the context of Conformal Prediction, software quality can be tied to the reliability of prediction intervals and the framework's scalability. The reliability of prediction intervals directly impacts decision-making in applications like medical diagnosis and financial forecasting. Poorly calibrated intervals can lead to incorrect decisions and costly errors. To ensure reliability, software engineers can employ rigorous testing and validation techniques. They should also focus on robust implementation practices, including error handling and fault tolerance, to prevent software failures that could compromise the integrity of the prediction intervals. Scalability is also a critical consideration, particularly in large-scale applications. Conformal Prediction models can handle diverse data sources and adapt to changing conditions. Software engineers must design systems that can efficiently process large volumes of data and scale when needed.

One of the basic principles of software engineering is requirements engineering. In XAI and Conformal Prediction projects, this involves documenting not only the functional requirements but also the interpretability and reliability requirements. Collaborating closely with domain experts and end-users is essential to ensure that the software meets their needs for transparency and reliability.

Software engineering can be applied to the entire lifecycle of machine learning models in XAI and Conformal Prediction projects, including version control for the deployed model and automated testing pipelines to ensure that models are developed and deployed reliably.

3 Sentiment Analysis, AutoML, and the Application of XAI and Conformal Prediction

In the evolving landscape of machine learning, Sentiment Analysis and AutoML (Automated Machine Learning) are examples where ensuring transparency, reliability, and predictive uncertainty become essential conditions. Explainable Machine Learning and Conformal Prediction techniques offer valuable tools to enhance sentiment analysis models and AutoML systems.

3.1 Application of XAI in Sentiment Analysis

XAI methods, e.g., LIME and SHAP, can be applied to sentiment analysis models to provide interpretable explanations of predictions. For instance, if a model labels a tweet as "negative", XAI can highlight the words or phrases contributing to this prediction, which helps users understand why the model made a particular sentiment prediction, enhancing trust in the system. XAI can also reveal feature importance, indicating which words or features were most influential in determining sentiment. Such information can guide data preprocessing and feature engineering efforts, allowing for selecting features that are not only predictive but also interpretable. XAI can assist in the selection of the most suitable sentiment analysis model. It can provide insights into the possible strengths and weaknesses of different models, helping practitioners choose models that align with the desired level of transparency and accuracy.

Conformal prediction can also help assess the reliability of sentiment predictions. Instead of providing a single sentiment label, the model can output prediction intervals that convey the uncertainty of the sentiment classification. For example, if the model predicts a "neutral" sentiment with a wide prediction interval, this indicates a high degree of uncertainty, potentially signaling the need for further analysis or human review. This dynamic adaptation enhances the reliability of sentiment predictions.

3.2 AutoML Complements XAI and Conformal Prediction in Sentiment Analysis

AutoML automates the process of selecting, training, and tuning sentiment analysis models, which accelerates the development of interpretable models that can be further enhanced with XAI and Conformal Prediction techniques. AutoML tools can optimize hyperparameters to improve model performance and interpretability simultaneously. For instance, it can prioritize hyperparameters that control feature selection or influence the

trade-off between model complexity and interpretability. AutoML also leverages ensemble methods, combining multiple models for improved accuracy. XAI can help interpret ensemble decisions, while Conformal Prediction can provide reliable confidence measures for ensemble predictions.

4 Software Quality Assurance and Software Architecture and Design

Software Quality Assurance is a systematic approach to ensure that software products meet predefined quality standards. It contains different processes, activities, and techniques to prevent and address quality issues throughout the software development lifecycle. Software Quality Assurance is essential for delivering reliable, maintainable, and robust software systems.

Software Architecture and Design involve the process of defining the structures and components of a software system to meet specific requirements, as well as the decisions that will affect the behavior and the overall quality of software. It serves as the blueprint for the construction and evolution of software systems [4, 16].

4.1 Areas of Opportunity

As machine learning systems have become integral to many applications, there is a growing need to integrate software quality assurance practices seamlessly into machine learning development pipelines. Software quality assurance can play a crucial role in testing the robustness of machine learning models against various adversarial attacks and in assessing bias and fairness in predictive models. Research in this area can lead to the development of new testing methodologies and tools that help identify and mitigate bias, contributing to more robust and reliable predictive models. There also is a need for quality assurance metrics that not only assess model accuracy but also provide insights into model explainability and interpretability. Research can focus on defining and quantifying metrics that evaluate how well a model's decisions can be understood and trusted. Quality assurance can help develop ethical guidelines and legal regulations for privacy, transparency, and fairness in deployed machine learning models. This area of interest can also go the other way around, where we can benefit from the advances in machine learning algorithms to determine the quality of the software. Applying machine learning and explainable machine learning techniques in software quality prediction offers, for instance, the ability to identify and mitigate flaws early in the software development process. Machine learning models can be trained to analyze vast amounts of historical data, including code repositories, bug reports, and testing results, to detect patterns that humans may encounter. This predictive capability allows developers to proactively address potential issues before they face critical problems at production time, which is very helpful in reducing the cost and effort required for post-release bug fixes. Furthermore, ML-based quality prediction can enhance the overall software development process by providing insights into the most error-prone areas of the codebase. Developers can prioritize their efforts based on these insights, allocating resources more effectively and optimizing testing and code review processes. This targeted approach can lead to shorter development cycles and improved time to deploy software products. Explainable machine learning can also play a role here by explaining why certain code segments are low quality or need improvement. When a model suggests that a specific module of code is likely to contain defects, XAI can provide explanations such as the code complexity or past defect history. This information empowers developers to make informed decisions about whether and how to address the predicted issues.

Developing software architectures that facilitate the integration of explainable machine learning methods into systems in production is a critical challenge. Researchers can explore patterns that allow for the seamless incorporation of interpretability and transparency into the design of machine learning systems. With machine learning predictive model applications growing in complexity and scale, there is also a need for software architectures that can efficiently handle large datasets and distributed processing. Effective software architecture should consider the versioning and management of models. Developing tools for version control, model deployment, and model management can improve the robustness and maintainability of machine learning systems. In many applications, real-time decisions are critical, e.g., autonomous driving cars, and designing systems that support low-latency models is a significant challenge.

4.1.1 The First Paper: Software quality Estimation Using Machine Learning

The first paper [12] discusses the possibility of enhancing the accuracy of software quality estimation using machine learning. The paper assessed eight different machine learning algorithms to understand how well the algorithms work for reliability and maintainability prediction. Where reliability means how many problems there are in a system, and for maintainability, we look at how frequently changes are made to the system. The

authors experimented with eight different algorithms, where Random Forest stood out as the top performer. It achieved an AUC (Area Under the ROC Curve, a measure of prediction accuracy) of more than 0.8 for both reliability and maintainability predictions.

In reliability prediction in software development, six machine learning methods were employed alongside a set of Object-Oriented (OO) metrics. In the following, we provide an overview of the reliability experiment:

- The data used for reliability prediction were 3,189 instances, with 1,236 instances labeled as having defects.
- The primary goal was to assess the classifiers' ability to identify defective instances, so instances were labeled as either defective or not defective for classification.
- The names of the projects were removed from the dataset to prevent classifiers from relying on this as a factor, particularly for projects with very few defects.
- The compared algorithms are decisions trees, Random Forests, Naïve Bayes, Bayesian belief's network, Rule-based classifiers, Nearest neighbor classifiers, SVM (Support Vector Machine), and Multi-Layer Perceptron (MLP).

For the maintainability prediction in software systems. The same set of machine learning algorithms used for defect prediction are also used for maintainability prediction. Here is a summary of the maintainability experiment:

- The datasets were collected from two commercial systems.
- The measure of maintenance effort was the number of line changes made per class.
- 110 instances were collected, where each instance in the dataset is a class in the system.
- The objective is to evaluate if the classification model can accurately identify the severity of maintenance changes.

The evaluation results show that Random Forest is an efficient classifier for both reliability and maintainability prediction with a reasonable AUC value. There were also other algorithms with generally good performance, e.g., KNN and the Rule-based model. Bayesian networks performed very well in defect prediction but also had poor performance with mainability prediction.

4.1.2 The Second Paper: Software Architecture Challenges

The paper in [7] discusses the challenges and concerns introduced by machine learning components for software architecture and design. The challenges include data-dependent behaviors, the need to detect and address changes over time, and the timely acquisition of accurate data for retraining. The authors compiled the challenges from workshops, interviews with practitioners, and engagements within the field. The paper also outlines four categories of challenges in software architecture for machine learning that need to be tackled to facilitate the development and maintenance of machine learning systems. The four categories are: software architecture practices, architecture patterns and tactics, monitorability quality attributes, and co-architecting and co-versioning. Additionally, the paper highlights the need to explore architectural principles relevant to ML systems. It calls for a deeper examination of monitorability.

5 Future Trends

The future of Software Engineering is expected to evolve in order to cope with the new needs, especially in the context of machine learning systems that continue to be integrated into almost every domain. The demand for explainability and interpretability will also grow. XAI methods will play a role in ensuring that machine learning systems are trustworthy and can be understood.

The ethical aspects of machine learning are becoming increasingly important. Researchers must be aware of ethical considerations, biases, and fairness when developing machine learning systems. Ethical guidelines and standards for machine learning development will likely become more strict, and XAI can help mitigate many of these problems. Researchers will focus on building more efficient algorithms, improving model robustness, and addressing domain-specific challenges.

Software engineers will need to integrate machine learning modules seamlessly into existing software products, which requires knowledge of traditional software engineering and machine learning development. Machine learning will be used to automate various aspects of software development, from code generation to testing and bug detection. Software engineers will need to adapt to these changes and collaborate with machine learning tools to enhance their productivity. In the coming 5-10 years, machine learning engineering will likely become more central to the software development process.

References

[1] Daniel W. Apley and Jingyu Zhu. "Visualizing the effects of predictor variables in black box supervised learning models". English (US). In: *Journal of the Royal Statistical Society. Series B: Statistical Methodology* 82.4 (Sept. 2020), pp. 1059–1086. ISSN: 1369-7412. DOI: 10.1111/rssb.12377.

[2] Jerome Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29 (Oct. 2001), pp. 1189–1232. DOI: 10.2307/2699986.

[3] A. Gammerman, V. Vovk, and V. Vapnik. "Learning by Transduction". In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. UAI'98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., 1998, pp. 148–155. ISBN: 155860555X.

[4] DAVID GARLAN and MARY SHAW. "AN INTRODUCTION TO SOFTWARE ARCHITECTURE". In: *Advances in Software Engineering and Knowledge Engineering*, pp. 1–39. DOI: 10.1142/9789812798039_0001.

[5] Bryce Goodman and Seth Flaxman. "European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"". In: *AI Magazine* 38.3 (2017), pp. 50–57.

[6] Himabindu Lakkaraju et al. "Interpretable & Explorable Approximations of Black Box Models". In: *CoRR* abs/1707.01154 (2017).

[7] Grace A. Lewis, Ipek Ozkaya, and Xiwei Xu. "Software Architecture Challenges for ML Systems". In: *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2021, pp. 634–638. DOI: 10.1109/ICSME52107.2021.00071.

[8] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[9] Ninareh Mehrabi et al. "A Survey on Bias and Fairness in Machine Learning". In: *ACM Comput. Surv.* 54.6 (2021). ISSN: 0360-0300.

[10] Toshiki Mori and Naoshi Uchihira. "Balancing the trade-off between accuracy and interpretability in software defect prediction". In: *Empirical Software Engineering* 24.2 (2019), pp. 779–825. ISSN: 1573-7616.

[11] Emmanuel Pintelas, Ioannis E. Livieris, and Panagiotis Pintelas. "A Grey-Box Ensemble Model Exploiting Black-Box Accuracy and White-Box Intrinsic Interpretability". In: *Algorithms* 13.1 (2020). ISSN: 1999-4893.

[12] Sandeep Reddivari and Jayalakshmi Raman. "Software Quality Prediction: An Investigation Based on Machine Learning". In: *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. 2019, pp. 115–122. DOI: 10.1109/IRI.2019.00030.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 2016, pp. 1135–1144.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-Precision Model-Agnostic Explanations". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[15] Craig Saunders, Alexander Gammerman, and Volodya Vovk. "Transduction with Confidence and Credibility". In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. IJCAI '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 722–726. ISBN: 1558606130.

[16] Zhiyuan Wan et al. *Software Architecture in Practice: Challenges and Opportunities*. Aug. 2023.