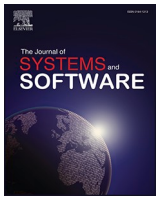
The Journal of Systems and Software 227 (2025) 112449

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss



Identifying inconsistent software defect predictions with symmetry metamorphic relation pattern

Pak Yuen Patrick Chan [a], Jacky Keung [a], Zhen Yang [b,*]

[a] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, China
[b] School of Computer Science and Technology, Shandong University, Qingdao, China

ARTICLE INFO

Editor: Marcos Kalinowski

Keywords:
Metamorphic testing
Metamorphic relation patterns
Software defect prediction
Explainable artificial intelligence

ABSTRACT

Determining inconsistent software defect predictions in machine learning-based systems poses a significant challenge. To address this issue, we propose the utilization of Metamorphic Testing (MT) incorporating the "symmetry" metamorphic relation pattern (MRP) to transform the training datasets for training follow-up systems. In contrast, original datasets are employed to train source systems. By comparing the occurrence of inconsistent predictions between source and follow-up systems and analysing the efficacy of this approach, we aim to shed light on its effectiveness. Additionally, Explainable Artificial Intelligence (XAI) is employed to explain the inconsistencies observed. The results demonstrate that the "symmetry" MRP can induce inconsistent predictions, and XAI techniques can effectively elucidate such inconsistencies. Moreover, we find that the ordering of small-sized and imbalanced datasets can contribute to inconsistencies when using the KMeans, Random Forests or Convolutional Neural Network algorithm for software defect prediction systems. To further advance this research, future studies can extend the proposed approach by incorporating additional MRPs in domains that utilize machine learning algorithms to identify and explain inconsistencies. Another promising research avenue involves investigating the relationship between data imbalance, dataset size, and MRPs to enhance the identification of inconsistencies and derive more robust MRs.

1. Introduction

Software defect prediction plays a crucial role in reducing repair costs by detecting defects early in the software development process. As a result, it has emerged as a significant area of research (Bala et al., 2022; Zhang et al., 2020). While machine learning techniques have been employed to enhance the performance of software defect prediction (Bala et al., 2022; Hemández-Molinos et al., 2021), it is imperative to ensure the accuracy of prediction results. To address this concern, our study delves into the application of Metamorphic Testing (MT) for identifying inconsistent predictions. We propose five Metamorphic Relations (MRs), which are derived from the symmetry Metamorphic Relation Patterns (MRPs), and conduct experiments on seven Software Defect Prediction Systems (SDPSs) to showcase the feasibility of MT in this context.

In this study, we employ MRs to generate follow-up datasets for training follow-up SDPSs, while the original datasets are used to train the source SDPSs. Our objective is to compare the occurrence of inconsistent predictions between the source and follow-up SDPSs and evaluate the effectiveness of this approach. Additionally, we leverage Explainable Artificial Intelligence (XAI) technology, specifically SHAP (SHapley Additive exPlanations), to provide explanations for the identified inconsistent predictions.

To guide our investigation, we formulate two research questions: "RQ1. Can MT identify inconsistent predictions?" and "RQ2. If so, can XAI provide additional explanations for the inconsistent predictions identified by MT?". The results of our study demonstrate that MT successfully identifies inconsistent predictions in SDPSs. Furthermore, we find that altering the order of data can lead to inconsistent predictions in four subject SDPSs. XAI, through the calculation of feature contributions using SHAP, provides reasonable explanations for the observed inconsistent predictions. These calculated contributions offer insights into the impact of changing the data order on SDPSs.

Furthermore, the utilization of the proposed MRs can serve as an effective validation tool for the K-Nearest-Neighbours (KNN), Support vector machine (SVM) and Logistic Regression (LR) algorithms. Incorporating MT in conjunction with XAI can effectively identify inconsistent predictions of the KMeans, Random Forests (RF) and Convolutional

* Corresponding author.
E-mail address: zhenyang@sdu.edu.cn (Z. Yang).

https://doi.org/10.1016/j.jss.2025.112449
Received 17 November 2024; Received in revised form 10 February 2025; Accepted 3 April 2025
Available online 4 April 2025

Neural Network (CNN) algorithms. This integrated approach aims to reduce uncertainty and provide a means for further examination and investigation.

In this paper, we conducted experiments on seven SDPSs utilizing various machine learning algorithms across eighteen publicly available datasets. The primary contributions of this study are as follows:

Identification and Explanation of Inconsistent Predictions: We successfully identify and provide explanations for the inconsistent predictions observed in the seven SDPSs, which implement diverse machine learning algorithms, using our proposed MRs. Notably, this is the first instance of combining MT and XAI techniques to detect and explain inconsistent predictions in SDPSs.

Effective Validation of SDPSs: Our proposed approach demonstrates its efficacy in effectively validating SDPSs. This signifies that the approach holds promise for extension to other machine learning-based systems, enabling the identification and explanation of inconsistent predictions in various domains.

The subsequent sections of this paper follow a structured framework. Section 2 delineates the background and related work of this research, encompassing three crucial aspects: software defect prediction, metamorphic testing, and explainable artificial intelligence. Section 3 expounds on the proposed approach employed in this study and furnishes an overview of the experiment's design and implementation process. In Section 4, the results will be presented and comprehensively discussed. The threats to the validity of this study are covered in Section 5. Finally, Section 6 encapsulates the conclusion and explores potential avenues for future research.

2. Background

This section briefly introduces software defection prediction, the preliminary knowledge of MT and Explainable Artificial Intelligence.

2.1. Software defect prediction

Recommending the likelihood of defects in programs is the primary function of software defect prediction (Wang et al., 2021; Xu et al., 2022), and the later the defect is discovered, the higher the correction cost; thus, applying software defect prediction at the early stage of software development is essential (Hemández-Molinos et al., 2021). Software defect prediction is an important and active research topic in software engineering (Bala et al., 2022; Zhang et al., 2020). One common approach is based on the features extracted from software modules to recommend the likelihood of defects, and "Code metrics" and "Process metrics" are the two main types of software features metrics (Bala et al., 2022; Sreedevi et al., 2019; Wang et al., 2021). The metrics become the training data of SDPSs, and various machine learning techniques, such as K-Nearest-Neighbours, K-means, Naive Bayes, Support Vector Machine, Random Forests, Logistic Regression and Convolutional Neural network, have been widely applied to SDPSs (Bala et al., 2022; Batool and Khan, 2022; He et al., 2015; Hemández-Molinos et al., 2021; Laradji et al., 2015; Pachouly et al., 2022).

Various advanced designs have been also proposed, such as an augmented-code property graph-based defect prediction method that uses code's graph representation combined with neural network to generate graph of defect region candidates to predict defects (Xu et al., 2022), supervised gated hierarchical long short-term memory networks extracted semantic features of abstract syntax trees of source code and features provided by the PROMISE repository to predict defects (Wang et al., 2021), unsupervised software defect prediction approach used threshold derivation for clustering to label the data points for defect prediction (Kumar et al., 2022). Ensemble different machine learning techniques and attention mechanisms to predict software defects can also provide more detailed information on the location of defects in code (Zhang et al., 2020). Cross-project defect prediction method (CPDP) is another software defect prediction approach that uses data from other source projects. This method is particularly beneficial when the target project has insufficient labelled data. Using defect knowledge from multiple source projects (Liu et al., 2022), applying transfer learning techniques to CPDP (Tang et al., 2021), or applying statement semantic learning and maximum mean discrepancy techniques to CPDP (Liu et al., 2021) can improve the performance and stability of defect prediction on the target project.

Although there are many advancements in software defect prediction technology, evaluating the performance of machine learning-based SDPSs is a test oracle problem because the systems can only predict at best with the given training data and can only find the local optimum rather than the global optimum. To address the test oracle problem, Metamorphic Testing is a commonly used technique (Sun et al., 2022; Xiao et al., 2022; Xie et al., 2020; Ying et al., 2021; Zhang et al., 2021; Zhou et al., 2020).

2.2. Metamorphic testing (MT)

Metamorphic Testing (MT) was introduced by Chen et al. (2020) as an approach to alleviate the problem of test oracles by verifying the relations between the inputs and outputs across multiple test executions of the programs or systems under examination. The major component of MT is metamorphic relations (MRs), which are the "expected" relationships between inputs and outputs across multiple software executions. If these MRs fail in various software executions, which means the output results are not as expected with inputs, then a fault of software is revealed (Segura et al., 2019; Zhang et al., 2021; Zhou et al., 2020).

For example, a mathematical property of the *sine* function is that "sin (x) = sin(π - x)". The corresponding MR is that if two function inputs x1 and x2 satisfy "x1 + x2 = π", then two function outputs should be equal, i.e., sin(x1) = sin(x2). Based on MR, x2 is constructed based on x1 and sin(x1). Therefore, we refer to x1 as a source input and x2 as a follow-up input. Such relations can be used to test the program as if the outputs of sin(x1) and sin(x2) are different. The implementation for this function "sin()" must have faults as it violates the above MR (Zhang et al., 2021).

MT has been applied to alleviate test oracle problems in many domains of applications and compilers (Segura et al., 2018), such as bioinformatic software (Stacy et al., 2022), machine learning classifiers (Ellis et al., 2021; Riccio et al., 2020; Saputra and Suharjito, 2019; Xie et al., 2011), cryptographic software (Pugh et al., 2019), online search engines (Segura et al., 2019), validating highly inter-related systems (Chua et al., 2021), and test order generation (Zhang et al., 2021). The concept of MT can also be applied to defined MRP and input patterns for creating multiple concrete MRs to enhance the understanding of systems (Zhou et al., 2020) or to assist in identifying MRs in testing query-based systems (Segura et al., 2019). Furthermore, an MR template with field operational test logs can generate MRs automatically for more robust testing in an autonomous robot vehicle case study (Cho et al., 2022; Deng et al., 2023; Luu et al., 2024).

2.3. Explainable artificial intelligence (XAI)

Although MT is a valuable tool in validating machine learning models, understanding the model's decision-making process is essential (Molnar, 2020). Despite applications of machine learning algorithms that have been implemented in various areas, there is still significant potential for improvement, and understanding the basis for the final decisions made by these models remains challenging (Lin et al., 2024; Zheng et al., 2022). This lack of transparency complicates the establishment of trust between humans and machines (Ajorloo et al., 2024; Lin et al., 2024; Zheng et al., 2022). In addition, the 2022 program solicitation "NSF22-502: National Artificial Intelligence (AI) Research Institutes: Accelerating Research, Transforming Society, and Growing the American Workforce" of U.S. National Science Foundation (NSF) has stated that AI system is trustworthy as "beyond reliability and

dependability, AI systems should exhibit transparency, accountability and explainability, among other attributes, in order to be deemed trustworthy" (NationalScienceFoundation, 2022). Therefore, ensuring the explainability of AI and machine learning systems is vital for fostering trust between humans and machines (Ajorloo et al., 2024; Lin et al., 2024; Zheng et al., 2022), and XAI plays a key role in clarifying the decision-making processes of these models (Bagheri et al., 2020; Zhang et al., 2022).

Two commonly used XAI technologies in explaining machine learning models are Local Interpretable Model-Agnostic Interpretation (LIME) and Shapley explanations (SHAP) (Molnar, 2020; Poth et al., 2020). The idea of LIME is to create an understudy of the target machine learning model and train it to approximately explain individual predictions of the target model (Molnar, 2020). SHAP is based on coalitional game theory, which explains prediction by "assuming that each feature value of the instance is a player in a game where the prediction is the payout" (Molnar, 2020), to measure each feature's contribution towards the prediction and calculates "Shapley values", and this value can explain the decision by the features' contributions (Molnar, 2020; Poth et al., 2020). XAI has been applied in explaining landslides prediction models (Pradhan et al., 2023), prediction models in the medical domain (Hauser et al., 2022; Loh et al., 2022; Sheu and Pardeshi, 2022), and malicious network traffic identification models (Zebin et al., 2022). Together with MT, XAI has also been applied to evaluate interpretation methods of the Deep Neural Network model (Fan et al., 2022), to search for regions of misclassified images detected by the Deep learning model (Torikoshi et al., 2023), and to search for defects caused by the incorrect decision of Deep Neural network (Yuan et al., 2022).

Thus, this study aims to apply MT to identify inconsistent predictions of SDPSs and XAI to explain identified inconsistent predictions. The goal is to gain a deeper understanding of why the system produces inconsistent predictions. We demonstrate the feasibility and practicality of this proposed approach through a comprehensive scaled experiment.

3. Methodology

This section describes the setup of the experiment. It defines the research objective and research questions of our experiment. Then, it describes the subject systems and data used in the experiment. Thereafter, it discusses the detailed experimental design, including environment configuration, measurement, experimental procedure, dataset preparation, and parameter setting.

3.1. Research objective and questions

Traditionally, MT identifies system faults through MR violations (Segura et al., 2019; M. Zhang et al., 2021; Zhou et al., 2020). We aim to explore if these violations clarify inconsistent predictions. Therefore, the main objective of this experiment is to establish the feasibility of MT by implementing the proposed MRs, which are derived from the Zhou et al.'s (2020) symmetry MRP, outlined in Section 3.2. By applying these MRs, we aim to identify inconsistent predictions within SDPSs. After we identify the inconsistent predictions, we apply XAI technology for further explanations and understanding. To achieve these goals, we address the following two research questions:

RQ1: Can MT identify inconsistent predictions? We evaluate the capacity of the proposed MRs by counting the occurrence of inconsistent predictions related to proposed MRs. This study assesses the efficacy of MRs by quantifying inconsistent prediction occurrences aligned with MRs.

RQ2: If so, can XAI provide additional explanations for the inconsistent predictions identified by MT? If inconsistencies arise, we select such cases for XAI-based explanations, in order to assess the capacity of XAI for further explanations of inconsistencies.

3.2. Proposed metamorphic relations (MRs)

This study adopts Zhou et al.'s (2020) symmetry MRP of MT to identify inconsistent predictions of SDPSs. Zhou et al. (2020) explore innovative applications of MRs that go beyond their conventional use in MT. They introduced the symmetry MRP and this pattern facilitates the generation of multiple specific MRs, which can uncover previously unknown failures in widely-used applications. The symmetry MRP "refers to the existence of different viewpoints from which the computer system appears the same" (Zhou et al., 2020, p. 1126). For example, regardless of what query languages are used for the inputs of the system, as long as the meanings of the queries are preserved, and from this perspective, the system should behave and perform the same. This study specifically examines the symmetry MRP, focusing on the concept of symmetry. It operates on the premise that predictions should remain unchanged as long as the semantic meanings of the data are preserved. In essence, systems should produce similar predictions using both source and follow-up datasets, provided that the contents of the datasets remain intact. If the MRs are not upheld, it indicates inconsistency.

Based on the symmetry MRP, we derive our MRs only change the order of data (not the contents) and assume the same results should be returned. We further explore whether different data ordering can cause inconsistent predictions. We select reverse, ascending and descending ordering and create five symmetric MRs that are inspired by Zhou et al.'s (2020) idea of symmetry, in order to understand the behavior of systems (Zhou et al., 2020). If there is violation of MRs, this violation can highlight the inconsistent predictions caused by the order of data. Five MRs are as follows:

MR1 - Reverse Order: This pattern reverses the order of source data from its original order and expects source and follow–up outputs to contain the same set of results. As only the order of the data is altered reversely, there is no change in the contents or meaning of data; thus, SDPSs with the source and follow-up training should produce identical outputs. For example, if we consider the training datasets "1, 2, 3, 4, 5, 6, 7, 8, 9, 10" and "10, 9, 8, 7, 6, 5, 4, 3, 2, 1," both sets contain the same content, and SDPSs, trained on either dataset should predict identical results with the same testing datasets.

For the following four MRs, they are using ascending and descending ordering. We use numeric features in ascending and descending order, where only the order of the data is altered. The contents and meaning of the data remain unchanged; therefore, SDPSs with the source and follow-up training should yield identical outputs.

As we are changing the ascending and descending ordering of source data, we need to select numeric features, and the order of those features should be independent of other features. Thus, we select the feature "bug number" (bug) and feature "line of codes" (loc) to manipulate the order of the training datasets. We chose these two features because they are the only ones that do not use the code content for measurements, as they only measure how many lines of code and how many bugs there are in the instance, unlike the other features, which measure the cohesion and coupling between codes (Jureczko and Madeyski, 2010). We believe that arranging the order of the datasets based on the values of features "loc" and "bugs", which do not use the code content for measurements, will not affect the contents of the training datasets. For example, training datasets such as "9, 6, 4, 3, 2, 5, 1, 7, 10, 8," "10, 9, 8, 7, 6, 5, 4, 3, 2, 1," and "1, 2, 3, 4, 5, 6, 7, 8, 9, 10" are all seen as having the same content by the SDPSs. As a result, SDPSs, trained on any of these datasets should yield identical results with the same testing datasets.

MR2.1 - Ascending first order by feature 'bug number' (bug) and ascending second order by feature 'line of codes' (loc): This pattern sorts the order of source data by firstly using the feature 'bug number' and secondly using the feature 'line of codes' in ascending order. If the values in the first-order feature are the same, we use the second-order feature to sort the data. It expects source and follow–up outputs to contain similar results.

MR2.2 - Descending first order by feature 'bug number' (bug)

and descending second order by feature 'line of codes' (loc): This pattern sorts the order of source data by firstly using the feature 'bug number' and secondly using the feature 'line of codes' in descending order. If the values in the first-order feature are the same, we use the second-order feature to sort the data. It expects source and follow–up outputs to contain similar results.

MR3.1 - Ascending first order by feature 'line of code' (loc) and ascending second order by feature 'bug number' (bug): This pattern sorts the order of source data by firstly using the feature 'line of code' and secondly using the feature 'bug number' in ascending order. If the values in the first-order feature are the same, we use the second-order feature to sort the data. It expects source and follow–up outputs to contain similar results.

MR3.2 - Descending first order by feature 'line of code' and descending second order by feature 'bug number': This pattern sorts the order of source data by firstly using the feature 'line of code' and secondly using the feature 'bug number' in descending order. If the values in the first-order feature are the same, we use the second-order feature to sort the data. It expects source and follow–up outputs to contain similar results.

3.3. Experiment design

Our experiment includes six steps.

Step 1. We randomly select twenty per cent of data of source datasets as test datasets and the rest as train datasets.
Step 2. This study uses MRs of MT for transforming the training datasets to train follow-up SDPSs. We apply the proposed MRs (Section 3.2) to create five follow-up train datasets for each source train dataset. This study has eighteen source datasets, creating ninety follow-up datasets.
Step 3. We train SDPSs with source datasets to create source SDPSs and train SDPSs with follow-up datasets to create follow-up SDPSs.
Step 4. We use source SDPSs and follow-up SDPSs to predict test datasets.
Step 5. We compare the prediction results and select inconsistent predictions from each test dataset.
Step 6. We use SHAP to explain and analyze the inconsistent predictions (Fig. 1).

Seven subject SDPSs are trained with eighteen subject source datasets, so there are one hundred and twenty-six source SDPSs. Five MRs have been applied to eighteen subject source datasets to generate ninety follow-up datasets. With seven subject SDPSs, there are six hundred and thirty follow-up SDPSs. We conducted one test trial with each associated test dataset for associated source SDPSs and follow-up SDPSs. Thus, there are one hundred and twenty-six test trials for source outputs and six hundred and thirty trials for follow-up outputs. Conclusively, we conducted seven hundred and fifty-six test trials in total.

3.4. Shapley XAI technology (SHAP)

This research selects SHAP to explain the identified inconsistent predictions of SDPSs because SHAP provided features' contributions (Molnar, 2020; Poth et al., 2020), which makes comparison possible. By comparing the features' contributions between source and follow-up SDPSs, we can analyse the relationship between features' contributions and inconsistent predictions, and the relationship between the MRs and features' contributions.

3.5. Subject under test (SUTs)

This experiment selected the following different and commonly used machine learning models for software defect prediction (Arora and Saha, 2018; Bala et al., 2022; Batool and Khan, 2022; Elish and Elish, 2008; He et al., 2015; Hemández-Molinos et al., 2021; Jureczko and Madeyski, 2010; Laradji et al., 2015; Pachouly et al., 2022). All SUTs used the standard configurations to implement in this study (Section 3.7).

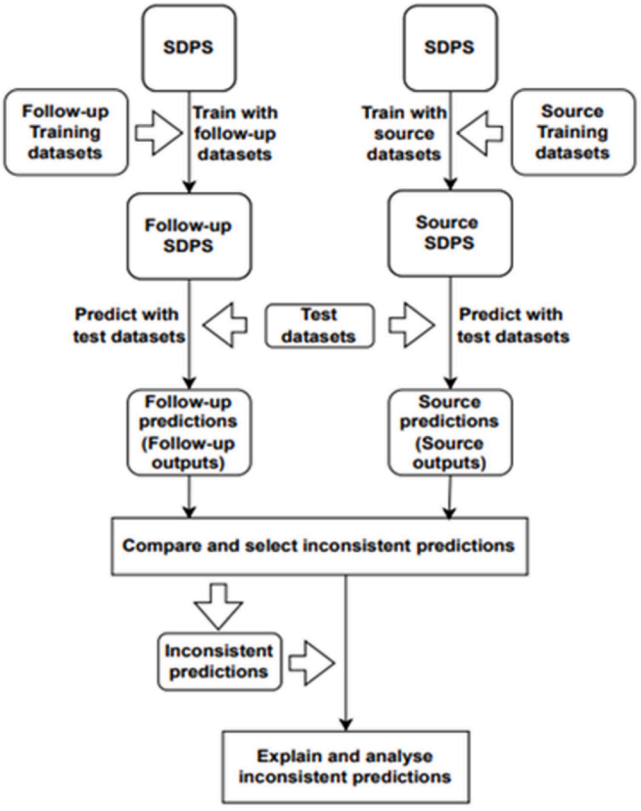


Fig. 1. Illustration of Step 3 to Step 6.

K-Nearest-Neighbors (KNN) classifies data into certain classes based on the parameter "k", which is the minimum neighborhood size of considered nearest neighbors. It uses the distance function to measure the similarity between instances and assigns the most common class of processing instance's k nearest neighbors to the processing instance (Chen et al., 2022; Jiang et al., 2007; Sathe and Adamuthe, 2021); thus, "k" is the most crucial hyper-parameter in KNN as this could affect the performance of the model (Chen et al., 2022).

K-Means (KMeans) uses squared Euclidean distance as the similarity measure to select instances with the minimal sum of squared distances between them for building clusters, and the number of clusters is set by the user (Venkatkumar and Shardaben, 2016; Z. Xu et al., 2021). In the "scikit-learn" library, there are two expectation-maximization (EM) algorithm options to choose (SciKit-Learn, 2022), and we treat each K-Means model with different EM algorithm options as two separated SDPSs. The first K-Means model uses the "Lloyd" EM algorithm option (KMeans(Lloyd)), which is based on observation to optimize one cluster given the other (SciKit-Learn, 2022). The second K-Means model uses the "Elkan" EM algorithm (KMeans(Elkan)), which uses triangular inequality to simplify distance calculations for clustering (SciKit-Learn, 2022).

Convolutional Neural network (CNN) is one of the typical deep learning algorithms used in software defect prediction (Batool and Khan, 2022; Pachouly et al., 2022). CNN is constructed from convolutional, pooling, merge, and fully-connected layers. The convolutional layers learn defect features from multiple views of instruction sequences. The pooling layers perform down-sampling operations to gather extracted information. The merge layers combine feature vectors of all views before feeding to the fully connected layers for computing the final class scores (Phan and Le Nguyen, 2017).

Random Forests (RF) is one of the mostly used clustering-based algorithms for software defect predictions. RF uses hyperbagging or averaging methods of different decision trees to give prediction results. These methods can help to improve performance and reduce the chance of overfitting the problems (Bala et al., 2022; Hemández-Molinos et al., 2021; Laradji et al., 2015).

Support vector machine (SVM) is a supervised learning model with associated learning algorithms typically used for classification and regression analysis by finding the optimal hyperplane that maximally separates samples into two different classes. The predictions for the data are based on which side they fall on. The functions which perform this mapping into the space are called kernel functions (Arora and Saha, 2018; Elish and Elish, 2008; He et al., 2015).

Logistic Regression (LR) is a probabilistic statistical regression model for categorical prediction. It works by fitting data to a logistic curve and is particularly useful for predicting a binary response from a binary predictor. This method helps predict the outcome of a categorical dependent variable based on one or more features (He et al., 2015; Laradji et al., 2015).

Therefore, this experiment employed seven SDPSs using different machine learning algorithms: KNN, SVM, RF, LR, CNN, KMeans(Lloyd) and KMeans(Elkan).

3.6. Dataset

This study has chosen the publicly available and widely used software defect data repository "PROMISE" (Bala et al., 2022) as the subject dataset. PROMISE data repository is designed for "research to predict effort and defect of a software" (Karim et al., 2017), which serves the intent of this study. We selected all eighteen datasets from PROMISE. As this study applies MRs to the data, only the numeric features are used, so twenty-one features of each dataset are chosen. As this is a two-class classification experiment, we have labelled the data as "defect" if the defect number is more than zero and "non-defect" if the defect number is zero.

3.7. Environment

This experiment environment included using an "Intel Core" i7 CPU with 64 GB RAM, "Windows 10" operating systems, "Jupyter Notebook" development platform and "Python" programming language and related libraries, such as "scikit-learn" and "tensorflow", for machine learning algorithms and "shap" for XAI. Standard parameters and options were applied to all subject systems. The option of cluster number was set to two because this is a "two classes" classification experiment. The option of the minimum number of neighbors to form a class was set to seven because it is one-third of the smallest test dataset (log4j-1.1) in this study. In addition, we use SVM with the Radial Basis Function (RBF) kernel (SciKit-Learn, 2024) and set the "random seed" parameters to a fixed number, the "batch size" parameter for CNN to thirty-two and the "epoch" parameter for CNN to ten, in order to reduce the randomness in the program. However, there is no option for the KMeans to set a fixed random seed (SciKit-Learn, 2022).

3.8. Measurement

This experiment measures the differences between the prediction outcomes of SDPSs trained by source datasets and SDPSs trained by follow-up datasets (more details in Section 3.3). Let the predictions of source SDPS be $P_{source} = [y_{s1}, y_{s2}, \ldots y_{sn}]$. Let the predictions of follow-up SDPS be $P_{follow\text{-}up} = [y_{p1}, y_{p2}, \ldots y_{pn}]$. We consider the difference between P_{source} and $P_{follow\text{-}up}$ as "Inconsistent prediction" and let $P_{inconsistent}$ be the number of predictions that differ between P_{source} and $P_{follow\text{-}up}$.

This study uses "Inconsistent Prediction Trial Count" (*IPC*) to count how many test trials contain one or more inconsistent predictions. If any test trial has $P_{inconsistent}$ larger than zero, we add one to *IPC*. We also use "Inconsistent Prediction Rate" (*IPR*) to calculate the percentage of inconsistent prediction in the total number of predictions in each test trial as Eq. (1). If there is no difference between source output and follow-up output, *IPC* and *IPR* are zero, and the SDPS is insensitive to the associated MR. Otherwise, these two measurements can show the inconsistent prediction caused by MRs.

$$IPR = \frac{P_{inconsistent}}{Total\ number\ of\ predictions} \quad , where\ P_{inconsistent} = P_{source} - P_{follow-up} \tag{1}$$

This study also uses prediction accuracy rate (ACC) and Matthew correlation coefficient (MCC) (Chicco and Jurman, 2020). These measurements can show any relationship between MRs and prediction accuracy. Let T_p be the actual positives that are correctly predicted, T_n be the actual negatives that are correctly predicted, F_p be the actual negatives that are wrongly predicted as positives, and F_n be the actual positives that are wrongly predicted negatives. ACC in this study is calculated as Eq. (2), and MCC is calculated as Eq. (3). Although ACC and MCC can be affected by class imbalance problems, this study mainly focuses on inconsistent predictions, not prediction performance, so ACC and MCC are supplementary measurements.

$$ACC = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{2}$$

$$MCC = \frac{T_p \times T_n - F_p \times F_n}{\sqrt{(T_p + F_p) \times (T_p + F_n) \times (T_n + F_p) \times (T_n + F_n)}} \tag{3}$$

For the statistical analysis, we use the "Paired T-test" to compare the features' contributions, interpreted by the XAI technology, between source and follow-up SDPSs. This study chooses the alpha level of 0.01 as the threshold for statistical significance. It means that if the probability value of the *t*-test is below 0.01, the *t*-test result is considered statistically significant (Moore et al., 2009). In other words, the difference between source and follow-up SDPSs is significant. In this experiment, we also use the "Cohen's d" formula (Sullivan and Feinn, 2012) to measure the effect size, which is the "raw difference between group means" (Sullivan and Feinn, 2012, p. 279). It is calculated as Eq. (4). This measurement is applied to the *t*-test results, which are very close to 0.01. In this way, if the effect size is around or larger than 0.8, we will also consider the result significant even if the statistical difference is on the edge of significance (Sullivan and Feinn, 2012).

$$Cohen's\ d\ effect\ size = (M1 - M2)/s \tag{4}$$

, where (M1 - M2) is the difference between the group means (M), and (s) is the standard deviation of either group.

4. Result and discussion

In this section, we study the violations with all subject SDPSs and MRs to assess the efficacy of MRs (RQ1). Then, we assess the capacity of XAI for further explanations with the selected inconsistent cases (RQ2). Finally, we analyze the findings further to gain insights from the experiment.

4.1. Violation with MRs

In this section, we discuss the violation with the proposed MRs. No SDPS violates one or two MRs. Subject SDPSs either violated all MRs or none (Tables 1 and 2). KMeans(Elkan) had the same results as KMeans (Lloyd) (Tables 1–4). KMeans(Lloyd) and KMeans(Elkan) had the highest average IPRs in this study, even though their IPCs are smaller than CNN (Tables 1 and 2). Table 2 shows that KMeans(Lloyd) and KMeans(Elkan) had average IPRs of around 38.55 % to 55.65 %, meaning that each test trial had at least 38 % of predictions considered

Table 1
List of IPC in percentage.

SDPS	MR1	MR2.1	MR2.2	MR3.1	MR3.2
KNN	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
SVM	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
LR	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
RF	6.00 %	6.00 %	11.00 %	11.00 %	6.00 %
KMeans(Lloyd)	50.00 %	50.00 %	61.11 %	61.11 %	66.67 %
KMeans(Elkan)	50.00 %	50.00 %	61.11 %	61.11 %	66.67 %
CNN	77.78 %	72.22 %	72.22 %	72.22 %	72.22 %

Table 2
List of average IPR.

SDPS	MR1	MR2.1	MR2.2	MR3.1	MR3.2
KNN	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
SVM	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
LR	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
RF	0.36 %	0.36 %	0.52 %	0.52 %	0.36 %
KMeans(Lloyd)	38.55 %	40.08 %	55.65 %	44.72 %	55.34 %
KMeans(Elkan)	38.55 %	40.08 %	55.65 %	44.72 %	55.34 %
CNN	4.60 %	3.84 %	2.68 %	3.80 %	2.53 %

Table 3
List of average ACC.

SDPS	Source	MR1	MR2.1	MR2.2	MR3.1	MR3.2
KNN	89.39 %	89.39 %	89.39 %	89.39 %	89.39 %	89.39 %
SVM	94.43 %	94.43 %	94.43 %	94.43 %	94.43 %	94.43 %
LR	98.10 %	98.10 %	98.10 %	98.10 %	98.10 %	98.10 %
RF	98.77 %	99.13 %	99.13 %	98.97 %	98.97 %	99.13 %
KMeans (Lloyd)	60.67 %	56.11 %	66.28 %	32.89 %	65.00 %	48.11 %
KMeans (Elkan)	60.67 %	56.11 %	66.28 %	32.89 %	65.00 %	48.11 %
CNN	85.22 %	86.11 %	84.78 %	84.61 %	83.22 %	85.33 %

Table 4
List of average MCC.

SDPS	Source	MR1	MR2.1	MR2.2	MR3.1	MR3.2
KNN	66.65 %	66.65 %	66.65 %	66.65 %	66.65 %	66.65 %
SVM	82.74 %	82.74 %	82.74 %	82.74 %	82.74 %	82.74 %
LR	94.25 %	94.25 %	94.25 %	94.25 %	94.25 %	94.25 %
RF	95.77 %	96.52 %	96.52 %	95.95 %	95.95 %	96.52 %
KMeans (Lloyd)	14.77 %	2.11 %	17.48 %	−20.55 %	11.94 %	−6.56 %
KMeans (Elkan)	14.77 %	2.11 %	17.48 %	−20.55 %	11.94 %	−6.56 %
CNN	33.84 %	41.55 %	28.45 %	31.28 %	27.38 %	34.07 %

inconsistent on average. Tables 3 and 4 show that the average ACCs and MCCs of KMeans(Lloyd) and KMeans(Elkan) are the lowest in this study, and KMeans(Lloyd) had inconsistent predictions with at least one MR of all eighteen datasets (Table 6). All results show that KMeans(Lloyd) and KMeans(Elkan) are sensitive to all the proposed MRs.

CNN had the highest IPCs in the study, as Table 1 shows that over 70 % of all test trials had one or more inconsistent predictions. Table 5 shows that CNN had no inconsistent predictions in four out of eighteen datasets but had inconsistent predictions with all five MRs in twelve

Table 5
IPC of CNN against each dataset.

Dataset	Size	MR1	MR2.1	MR2.2	MR3.1	MR3.2
ant-1.7	149	1	1	1	1	1
camel-1.0	68	0	0	0	0	0
camel-1.6	193	1	1	1	1	1
data_arc	45	0	0	0	0	0
data_ivy-2.0	71	1	0	0	0	1
data_prop-6	129	1	1	1	1	1
data_redaktor	35	0	0	0	0	0
jedit-3.2	55	1	1	1	1	1
jedit-4.2	74	1	1	1	1	1
log4j-1.1	22	1	1	1	1	0
lucene-2.0	39	1	1	1	1	1
poi-2.0	63	1	1	1	1	1
synapse-1.0	32	0	0	0	0	0
synapse-1.2	52	1	1	1	1	1
velocity-1.6	46	1	1	1	1	1
xalan-2.4	145	1	1	1	1	1
xerces-1.2	88	1	1	1	1	1
xerces-1.3	91	1	1	1	1	1
Total		**14**	**13**	**13**	**13**	**13**

Table 6
IPC of KMeans(Lloyd) against each dataset.

Dataset	Size	MR1	MR2.1	MR2.2	MR3.1	MR3.2
ant-1.7	149	1	1	1	1	1
camel-1.0	68	1	1	0	1	0
camel-1.6	193	1	1	0	1	1
data_arc	45	0	0	1	0	0
data_ivy-2.0	71	0	1	1	1	1
data_prop-6	129	1	1	0	1	1
data_redaktor	35	0	0	1	0	1
jedit-3.2	55	1	0	0	1	1
jedit-4.2	74	1	1	0	1	1
log4j-1.1	22	0	0	1	1	1
lucene-2.0	39	0	0	1	0	0
poi-2.0	63	0	1	0	0	1
synapse-1.0	32	0	0	1	1	1
synapse-1.2	52	0	1	1	1	0
velocity-1.6	46	1	0	0	0	0
xalan-2.4	145	1	0	1	1	1
xerces-1.2	88	1	0	1	0	1
xerces-1.3	91	0	1	1	0	0
Total		**9**	**9**	**11**	**11**	**12**

datasets. Even Table 2 shows CNN had average IPRs around 2.68 % to 4.60 %, which means each test trial had less than 5 % of predictions considered inconsistent on average, CNN is considered sensitive to the proposed MRs.

RF had 6 % IPCs with MR1, MR2.1 and MR3.2, whereas 11 % with MR2.2 and MR3.1. It is because RF only had inconsistent cases with two datasets, “data_redaktor” and “velocity-1.6” (Table 7). Table 2 shows that RF had average IPRs of around 0.36 % to 0.52 %, meaning that each test trial had less than or equal to 0.52 % of predictions considered inconsistent on average. Although Tables 3 and 4 show that the average ACCs and MCCs of RF are the highest in this study, RF is considered sensitive to the MRs.

This study expected that the proposed MRs would yield similar results (Section 3.2). However, MR1, MR2.1, MR2.2, MR3.1, and MR3.2 produced different IPCs, IPRs, ACCs and MCCs (Tables 1–4). This indicates that the results were not as consistent as expected.

The findings reveal that altering the order of training data, which are the proposed MRs, can lead to inconsistencies in KMeans(Lloyd), KMeans(Elkan), RF and CNN. Furthermore, all the inconsistent cases across the various SDPSs can be identified by the proposed MRs because this experiment measures the differences in prediction outcomes between SDPSs trained on source datasets and SDPSs trained on follow-up datasets, both of which use the same test dataset for comparison.

Table 7
IPC of RF against each dataset.

Dataset	Size	MR1	MR2.1	MR2.2	MR3.1	MR3.2
ant-1.7	149	0	0	0	0	0
camel-1.0	68	0	0	0	0	0
camel-1.6	193	0	0	0	0	0
data_arc	45	0	0	0	0	0
data_ivy-2.0	71	0	0	0	0	0
data_prop-6	129	0	0	0	0	0
data_redaktor	35	0	0	1	1	0
jedit-3.2	55	0	0	0	0	0
jedit-4.2	74	0	0	0	0	0
log4j-1.1	22	0	0	0	0	0
lucene-2.0	39	0	0	0	0	0
poi-2.0	63	0	0	0	0	0
synapse-1.0	32	0	0	0	0	0
synapse-1.2	52	0	0	0	0	0
velocity-1.6	46	1	1	1	1	1
xalan-2.4	145	0	0	0	0	0
xerces-1.2	88	0	0	0	0	0
xerces-1.3	91	0	0	0	0	0
Total		**1**	**1**	**2**	**2**	**1**

Consequently, the proposed MRs from the MT approach are capable of identifying test cases that result in inconsistent predictions.

Answering RQ1: Results show that proposed MRs of the MT approach can induce and identify inconsistent predictions of SDPSs using KMeans(Lloyd), KMeans(Elkan), RF and CNN algorithms. These identified inconsistencies allow for examining whether XAI can be used to explain the inconsistencies further in the next section.

4.2. Inconsistent predictions with XAI-explanation

In this section, we use XAI to interpret the inconsistencies with each of the machine learning algorithms that had violated cases.

4.2.1. CNN

As CNN has violated fourteen datasets, we select one inconsistent case from each of the violated datasets (Table 5) and use SHAP to explain the features' contributions to inconsistent prediction between source SDPS and follow-up SDPSs (Table 8). Then, we use multiple paired T-tests to examine the differences between the features' contributions (Table 9).

SHAP explained the features' contributions of source and follow-up SDPSs on the inconsistent prediction of the selected test instance number 57 of the "poi-2.0" dataset (Table 8). Table 9 shows the multiple paired T-test results between the lists of features' contributions of source and follow-up SDPSs, and all of them are not significantly different, which can be interpreted as the prediction result of test instance number 57 of the "poi-2.0" dataset can differ with a slight change in the features' values, which is caused by the order change in training data.

Table 9
Multiple Paired T-test results of features' contributions of predicting test case #57 of "poi-2.0" between source and follow-up SDPSs using CNN.

Paired T-test group	p-value = $P(T \leq t)$ two-tail (alpha=0.01, df=20)	t-score (df=20)	Reject H0: mean difference == 0 (p-value ⟨ 0.01 or t-score ⟩ 2.8453[a])
Source vs MR1	0.9271	0.0927	No
Source vs MR2.1	0.3160	1.0286	No
Source vs MR2.2	0.8990	0.1286	No
Source vs MR3.1	0.8645	0.1729	No
Source vs MR3.2	0.5734	0.5725	No

[a] t-critical(alpha=0.01, df=20) = 2.8453, where it is paired t-test with 21 features of dataset PROMISE, thus degree of freedom is 20 (21–1).

Table 10 shows that the multiple paired T-test results of features' contributions of all the selected test instances with inconsistent predictions from eighteen datasets are insignificant as all the p-values are larger than the alpha level (0.01). These results matched the high IPC and low IPR of CNN because the differences in features' contributions are minor, so there are only several inconsistent predictions in each violated dataset. Thus, the inconsistent prediction caused by the order of data can be difficult to detect. In addition, CNN had over 83 % average ACC and over 27 % average MCC (Tables 3 and 4), which can be misinterpreted as the model is stable. Therefore, MT and XAI are valuable tools for CNN as they explain that the sequence of the data is critical to CNN and changing the order of data could affect the features' contributions, even slightly, and lead to inconsistent predictions.

We then examine the differences in the contribution of each feature. Table 11 displays the differences in the percentage of features' contributions between source and follow-up. The result indicates that the proposed MRs significantly impact the feature "dam", which stands for "Data Access Metric" (Ferenc et al., 2018; Jureczko and Madeyski, 2010), in the "poi-2.0" dataset from PROMISE (Table 11). This can be

Table 8
Illustration of SHAP explains inconsistent prediction of test case #57 of "poi-2.0" with source and follow-up SDPSs using CNN by features' contributions.

Feature	Source	MR1	MR2.1	MR2.2	MR3.1	MR3.2
wmc	−5.19E−04	3.93E−03	−1.85E−03	−3.71E−04	−1.75E−02	2.94E−03
dit	6.83E−02	1.23E−04	−3.13E−03	−3.06E−02	−1.13E−02	−1.44E−02
noc	−1.77E−03	−7.67E−04	−3.79E−04	−2.42E−03	−3.35E−03	−4.17E−04
cbo	2.10E−02	6.46E−03	3.43E−03	−1.01E−03	−3.10E−02	9.47E−03
rfc	2.38E−02	−9.24E−04	7.91E−03	−7.17E−03	1.00E−02	5.59E−03
lcom	3.98E−02	−5.02E−03	1.18E−03	1.54E−03	5.80E−02	5.08E−03
ca	1.25E−02	−6.41E−04	−4.61E−03	9.57E−04	3.16E−01	9.73E−04
ce	2.89E−03	4.16E−03	−3.71E−03	−2.31E−04	−5.05E−03	−9.06E−03
npm	−2.92E−02	−3.11E−04	−2.27E−03	1.03E−03	−5.50E−02	3.18E−03
lcom3	−2.75E−02	−3.80E−03	−5.95E−03	−4.94E−03	−3.29E−02	−6.68E−03
loc	−8.62E−03	5.04E−03	1.46E−03	5.79E−04	5.82E−03	−6.64E−03
dam	−1.09E−04	1.92E−02	−1.54E−03	−1.77E−02	−6.96E−02	−2.31E−02
moa	1.38E−02	−9.01E−03	−8.55E−03	−3.91E−03	3.97E−03	−6.27E−03
mfa	−1.20E−01	−1.97E−02	−1.78E−02	−2.19E−02	−1.06E−02	−2.15E−02
cam	3.29E−02	−2.64E−04	−5.73E−03	−4.72E−03	−1.64E−02	−1.40E−02
ic	2.43E−02	9.94E−03	−7.48E−03	−1.56E−02	−5.12E−03	−3.16E−02
cbm	1.40E−02	−1.08E−02	5.72E−03	−5.22E−03	−1.50E−02	−7.49E−03
amc	−1.66E−02	−5.33E−03	−1.39E−03	−4.98E−03	−1.56E−02	−9.55E−03
max_cc	−2.38E−01	1.38E−02	−1.86E−02	5.76E−02	−1.18E−01	2.25E−02
avg_cc	−9.12E−02	−3.47E−01	2.42E−03	−2.23E−01	−2.96E−01	−6.39E−02
bug	−1.08E−01	−9.77E−03	−2.64E−02	−5.83E−02	−3.15E−03	−3.98E−02

Table 10
Multiple Paired T-test results of features' contributions of predicting all selected inconsistent cases between source and follow-up SDPSs using CNN[a].

Dataset	Paired T-test group p-value = $P(T \leq t)$ two-tail (alpha=0.01, df=20)					Reject H0: mean difference == 0 (p-value < 0.01)
	Source vs MR1	Source vs MR2.1	Source vs MR2.2	Source vs MR3.1	Source vs MR3.2	
ant-1.7	0.6231	0.3629	0.3756	0.1579	0.4083	No
camel-1.6	0.6835	0.5485	0.6138	0.5314	0.8792	No
data_ivy-2.0	0.6038	Nil	nil	nil	0.2675	No
data_prop-6	0.8196	0.7256	0.9405	0.3549	0.5106	No
jedit-3.2	0.4266	0.6545	0.6035	0.7409	0.4160	No
jedit-4.2	0.5625	0.9757	0.6568	0.9193	0.5598	No
log4j-1.1	0.2717	0.8345	0.7099	nil	0.6851	No
lucene-2.0	0.7662	0.8107	0.4689	0.2053	0.7786	No
poi-2.0	0.9271	0.3160	0.8990	0.8645	0.5734	No
synapse-1.2	0.7685	0.4583	0.6382	0.6369	0.4207	No
velocity-1.6	0.6404	0.6431	0.7157	0.9559	0.1082	No
xalan-2.4	0.5574	0.4124	0.6856	0.7596	0.7260	No
xerces-1.2	0.9245	0.6545	0.8797	0.9093	0.8780	No
xerces-1.3	0.8187	0.7454	0.1270	0.9061	0.4166	No

[a] "nil" means there is no inconsistent case.

Table 11
Illustration of SHAP explains inconsistent prediction of test case #57 of "poi-2.0" with source and follow-up SDPSs using CNN by the differences in percentage of features' contributions.

Feature	Diff(%) Source vs MR1	Diff(%) Source vs MR2.1	Diff(%) Source vs MR2.2	Diff(%) Source vs MR3.1	Diff(%) Source vs MR3.2
wmc	856.65 %	257.23 %	28.52 %	3275.53 %	665.51 %
dit	99.82 %	104.59 %	144.75 %	116.59 %	121.07 %
noc	56.67 %	78.59 %	36.89 %	89.38 %	76.44 %
cbo	69.19 %	83.65 %	104.80 %	248.02 %	54.82 %
rfc	103.88 %	66.75 %	130.13 %	57.91 %	76.51 %
lcom	112.60 %	97.04 %	96.12 %	45.65 %	87.23 %
ca	105.12 %	136.79 %	92.36 %	2423.35 %	92.23 %
ce	43.91 %	228.34 %	107.99 %	274.71 %	413.39 %
npm	98.94 %	92.24 %	103.50 %	88.11 %	110.86 %
lcom3	86.17 %	78.37 %	82.03 %	19.48 %	75.70 %
loc	158.44 %	116.93 %	106.72 %	167.53 %	22.90 %
dam	17,711.01 %	1308.26 %	16,180.73 %	63,725.69 %	21,048.62 %
moa	165.09 %	161.75 %	128.26 %	71.33 %	145.33 %
mfa	83.61 %	85.23 %	81.81 %	91.15 %	82.14 %
cam	100.80 %	117.45 %	114.35 %	149.81 %	142.71 %
ic	59.13 %	130.78 %	164.31 %	121.06 %	229.94 %
cbm	177.13 %	59.16 %	137.27 %	206.86 %	153.51 %
amc	67.96 %	91.62 %	70.05 %	6.34 %	42.59 %
max_cc	105.80 %	92.16 %	124.25 %	50.48 %	109.47 %
avg_cc	280.34 %	102.65 %	144.35 %	224.45 %	29.91 %
bug	90.99 %	75.68 %	46.29 %	97.10 %	63.33 %

interpreted as changing the order of training dataset can significantly influence how CNN calculates the contribution of feature "dam", which is "the ratio of the number of private (protected) attributes to the total number of attributes declared in the class" (Jureczko and Madeyski, 2010, p.3).

This impact is especially pronounced with MR3.1, which is sorting the training data with features "loc" and "bug". Thus, MT with XAI can give further explanations with the selected inconsistent cases (#57 of the "poi-20" dataset) as the sorting of training data can affect how CNN consider the feature "dam", consequently leading to changes in its predictions.

4.2.2. KMeans

We selected the same test instance number 57 of the "poi-2.0" dataset used in CNN for SHAP to explain (Table 8). Tables 12 and 13 present the multiple paired T-test results comparing the features' contributions of source and follow-up SDPSs using KMeans(Lloyd) and KMeans(Elkan). The results show that KMeans(Elkan) and KMeans (Lloyd) are significantly different between source, MR2.1 and MR3.2 (Tables 12 and 13) as they have inconsistent predictions in the "poi-2.0" test dataset with MR2.1 and MR3.2 only. With the same inconsistent case, these results show that KMeans(Lloyd) and KMeans(Elkan), which are highly sensitive to the order of data, considered test instance number 57 as a case with different features' contributions.

To understand further, we select one inconsistent case from each of the eighteen datasets and use SHAP to explain the features' contributions to inconsistent predictions between source SDPSs and follow-up SDPSs. Then, we use multiple paired T-tests to examine the differences between the features' contributions. Tables 14 and 15 show the multiple paired T-test results of features' contributions of all the selected inconsistent test cases between source and follow-up SDPSs.

Almost all the results are statistically significant, as most p-values are below the alpha level of 0.01. Thus, the chance of inconsistent prediction caused by the order of data is significant, and KMeans is sensitive to changes in the order of data. Thus, MT and XAI reveal that the order of data can bring inconsistencies for KMeans(Lloyd) and KMeans(Elkan).

Tables 16 and 17 show the differences in the percentage of features' contributions between source, MR2.1 and MR3.2. As KMeans(Lloyd) and KMeans(Elkan) have inconsistent predictions in the "poi-2.0" test dataset with MR2.1 and MR3.2 only, we only calculate the differences between source, MR2.1 and MR3.2. However, the features' contributions explained by XAI do not provide deeper insights, as all the differences in weightings are quite similar. While XAI reveals notable variations in the contributions of all features between the source, MR2.1, and MR3.2, we did not identify any significant differences in the contribution of any individual feature when comparing the source and the follow-up.

4.2.3. RF

Since RF only has violations with datasets "data_redaktor" and "velocity-1.6", we select one inconsistent case from each of the violated datasets (Table 7) and used SHAP to explain the features' contributions to inconsistent prediction between source SDPS and follow-up SDPSs (Table 18). Then, we performed multiple paired T-tests to examine the differences in features' contributions.

SHAP explained the features' contributions of source and follow-up SDPSs on the inconsistent prediction of test instance number 44 of the "velocity-1.6" dataset (Table 18). Table 19 displays the multiple paired T-test results between the lists of features' contributions of source and follow-up SDPSs. None of the differences were statistically significant, suggesting that the prediction for test instance number 44 from the "velocity-1.6" dataset can vary with only slight changes in feature values, a result attributed to the change in order of the training data.

Table 20 presents the multiple paired T-test results of all the selected test instances with inconsistent predictions from two datasets between source and follow-up. The results are insignificant as all the p-values are exceeding the alpha level of 0.01. These results matched the low IPC and

Table 12
Multiple paired T-test results of features' contributions of predicting test case #57 of "poi-2.0" between source and follow-up SDPSs using KMeans(Lloyd).

Paired T-test group	p-value = $P(T \leq t)$ two-tail (alpha=0.01, df=20)	t-score (df=20)	Reject H0: mean difference == 0 (p-value 〈 0.01 or t-score 〉 2.8453[1])
Source vs MR2.1	0.0003	4.3061	Yes
Source vs MR3.2	0.0003	4.3780	Yes

Table 13
Multiple paired T-test results of features' contributions of predicting test case #57 of "poi-2.0" between source and follow-up SDPSs using KMeans(Elkan).

Paired T-test group	p-value = $P(T \leq t)$ two-tail (alpha=0.01, df=20)	t-score (df=20)	Reject H0: mean difference == 0 (p-value 〈 0.01 or t-score 〉 2.8453[1])
Source vs MR2.1	0.0002	4.4682	Yes
Source vs MR3.2	0.0002	4.5415	Yes

low IPR of RF because the differences in all feature contributions are insignificant, so there are only several inconsistent predictions in each violated dataset.

Thus, the inconsistent predictions caused by the order of data can be difficult to detect. Furthermore, RF achieved the highest average ACCs and MCCs in the experiment (Tables 3 and 4), which could mislead one into thinking that the model is stable. Consequently, MT and XAI prove to be valuable for RF, clarifying that the sequence of data is crucial; even minor changes in the order can slightly influence feature contributions, leading to inconsistent predictions.

Table 18 illustrates the differences in features' contributions between source and follow-up. The result shows that proposed MRs significantly impact one feature: "bug", which refers to the total number of bugs (Jureczko and Madeyski, 2010). In contrast, the contributions of all other features become negligible. This finding suggests that the order of the training dataset can significantly influence RF model's calculation of the "bug" feature's contributions. Thus, MT with XAI can provide additional explanations for the selected inconsistent cases (#44 of the "velocity-1.6" dataset) as the sorting of training data affects how RF evaluates the feature's contribution of "bug" and lead to a change in the prediction of RF.

Answering RQ2: The results demonstrate that XAI can offer deeper explanations and insights into the inconsistent predictions of SDPSs using CNN and RF by identifying the features' contributions towards the inconsistent predictions. Thus, our proposed approach with MT and XAI to identify and explain inconsistencies of SDPSs is feasible.

4.3. Further analysis

In this section, we further analyse the findings with "zero violation with MRs, " characteristic of algorithms", and "characteristic of datasets".

Table 14
Multiple paired T-test results of features' contributions of predicting all selected inconsistent test cases between source and follow-up SDPSs using KMeans(Lloyd)[a].

Dataset	Paired T-test group p-value = $P(T \leq t)$ two-tail (alpha=0.01, df=20)					Reject H0: mean difference == 0 (p-value< 1.00E-02)
	Source vs MR1	Source vs MR2.1	Source vs MR2.2	Source vs MR3.1	Source vs MR3.2	
ant-1.7	4.93E–04	4.05E–04	5.19E–04	4.11E–04	6.26E–04	Yes
camel-1.0	5.28E–03	6.01E–03	5.61E–03	Nil	Nil	Yes
camel-1.6	5.73E–05	6.88E–05	Nil	1.78E–04	8.43E–08	Yes
data_arc	Nil	nil	8.85E–07	Nil	Nil	Yes
data_ivy-2.0	Nil	1.85E-03	2.78E–09	2.60E–09	3.51E–09	Yes
data_prop-6	4.47E–07	5.57E–07	Nil	3.98E–07	1.42E–02	Yes
data_redaktor	nil	nil	2.05E–06	nil	1.71E–06	Yes
jedit-3.2	3.72E–02	nil	Nil	4.04E–02	3.51E–02	Yes
jedit-4.2	7.95E–08	1.95E–07	Nil	8.44E–08	9.31E–08	Yes
log4j-1.1	nil	nil	4.66E–07	2.35E–07	1.74E–07	Yes
lucene-2.0	nil	nil	2.27E–09	nil	Nil	Yes
poi-2.0	nil	7.43E–03	Nil	nil	6.83E–03	Yes
synapse-1.0	nil	Nil	1.85E–02	3.16E–06	4.03E–06	Yes
synapse-1.2	nil	3.00E–04	3.91E–04	3.44E–04	Nil	Yes
velocity-1.6	1.47E–05	nil	Nil	Nil	Nil	Yes
xalan-2.4	2.64E–07	nil	2.68E–07	6.47E–03	2.72E–07	Yes
xerces-1.2	2.40E–09	nil	1.80E–09	nil	2.36E–09	Yes
xerces-1.3	Nil	6.68E–09	6.88E–09	nil	nil	Yes

[a] "nil" means there is no inconsistent case.

Table 15
Multiple paired t-test results of features' contributions of predicting all selected inconsistent test cases between source and follow-up SDPSs using KMeans(Elkan)[3].

Dataset	Paired T-test group p-value = $P(T \leq t)$ two-tail (alpha=0.01, df=20)					Reject H0: mean difference == 0 (p-value < 1.00E-02)
	Source vs MR1	Source vs MR2.1	Source vs MR2.2	Source vs MR3.1	Source vs MR3.2	
ant-1.7	1.06E−01	1.03E−01	1.06E−01	1.11E−01	7.04E−02	No
camel-1.0	2.18E−06	2.87E−06	Nil	3.31E−06	Nil	Yes
camel-1.6	7.19E−05	6.10E−05	Nil	2.03E−04	1.07E−07	Yes
data_arc	nil	nil	1.45E−06	nil	Nil	Yes
data_ivy-2.0	nil	1.55E−03	4.94E−06	5.22E−06	4.16E−06	Yes
data_prop-6	2.71E−07	1.93E−07	Nil	2.67E−07	1.45E−02	Yes
data_redaktor	nil	nil	3.27E−09	nil	2.09E−09	Yes
jedit-3.2	3.35E−02	nil	Nil	3.77E−02	3.34E−02	No
jedit-4.2	7.53E−08	1.44E−07	Nil	1.47E−07	1.13E−07	Yes
log4j-1.1	nil	nil	1.13E−04	6.44E−05	9.19E−05	Yes
lucene-2.0	nil	nil	4.56E−04	Nil	Nil	Yes
poi-2.0	nil	2.79E−04	Nil	nil	3.25E-04	Yes
synapse-1.0	nil	Nil	1.25E−02	9.94E−07	6.87E−07	Yes
synapse-1.2	nil	3.13E−05	1.92E−05	1.58E−04	nil	Yes
velocity-1.6	7.71E−07	nil	Nil	Nil	nil	Yes
xalan-2.4	5.08E−07	nil	5.32E−07	7.07E−03	3.96E−07	Yes
xerces-1.2	5.78E−09	nil	4.19E−09	nil	7.54E−09	Yes
xerces-1.3	nil	4.86E−05	8.07E−05	nil	Nil	Yes

Table 16
Illustration of SHAP explains inconsistent prediction of test case #57 of "poi-2.0" with source and follow-up SDPSs using KMeans(Lloyd) by the differences in percentage of features' contributions[a].

Feature	Diff(%) Source vs MR2.1	Diff(%) Source vs MR3.2
Wmc	203.11 %	197.84 %
Dit	199.05 %	191.80 %
Noc	nil	100.00 %
Cbo	nil	nil
Rfc	100.00 %	nil
Lcom	180.44 %	100.00 %
Ca	nil	nil
Ce	nil	nil
Npm	164.61 %	197.85 %
lcom3	100.00 %	100.00 %
Loc	nil	nil
Dam	nil	100.00 %
Moa	100.00 %	100.00 %
Mfa	193.82 %	194.13 %
Cam	100.00 %	100.00 %
Ic	186.50 %	195.98 %
Cbm	202.31 %	201.35 %
Amc	100.00 %	186.45 %
max_cc	239.64 %	201.72 %
avg_cc	192.00 %	230.54 %
Bug	258.88 %	194.58 %

[a] "nil" means the feature's contribution of "source" and "follow-up" are both zero. "100 %" means either the feature's contribution of "source" or "follow-up" is zero.

Table 17
Illustration of SHAP explains inconsistent prediction of test case #57 of "poi-2.0" with source and follow-up SDPSs using KMeans(Elkan) by the differences in percentage of features' contributions[4].

Feature	Diff(%) Source vs MR2.1	Diff(%) Source vs MR3.2
Wmc	142.45 %	239.30 %
Dit	210.17 %	212.02 %
Noc	nil	nil
Cbo	184.89 %	nil
Rfc	100.00 %	nil
Lcom	458.61 %	100.00 %
Ca	nil	nil
Ce	nil	nil
Npm	185.97 %	156.71 %
lcom3	nil	nil
Loc	nil	nil
Dam	100.00 %	nil
Moa	nil	nil
Mfa	213.89 %	193.51 %
Cam	nil	nil
Ic	198.24 %	191.71 %
Cbm	198.73 %	210.76 %
Amc	232.05 %	100.00 %
max_cc	262.28 %	150.36 %
avg_cc	216.57 %	225.60 %
Bug	184.88 %	206.68 %

4.3.1. Zero violation with MRs

KNN, SVM, and LR demonstrated zero IPC and zero IPR across all five MRs, which means there was no inconsistent prediction (Tables 1 and 2). Their average ACCs and MCCs with all five MRs are all the same, which shows that the predictions are all stable (Tables 3 and 4). Since KNN, SVM, and LR had no inconsistent prediction with all MRs, data can be updated in any order to train them without causing inconsistent prediction. Hence, the proposed MRs can be served as an effective and valuable validation tool for these algorithms. If any inconsistent prediction occurs with our MRs, this could infer that the SDPS using KNN, SVM, or LR contains errors that require further investigation, as these algorithms are insensitive to our MRs (Segura et al., 2019; M. Zhang et al., 2021; Zhou et al., 2020).

Moreover, the average ACCs and MCCs of KNN, SVM and LR were high in this study. MRs did not influence the performance of KNN, SVM, and LR, demonstrating that changing the order of data did not cause inconsistent predictions in KNN, SVM, and LR. This result confirms that KNN, SVM and LR are excellent choices for SDPSs. Therefore, the findings recommend applying KNN, SVM and LR algorithms as SDPS due to their high accuracy rate, along with implementing the proposed MRs as the effective validation tool to ensure the correctness of SDPSs' outcomes.

Table 18
Illustration of SHAP explains inconsistent prediction of test case #44 of "velocity-1.6" with source and follow-up SDPSs using RF by features' contributions.

Feature	Source	MR1	MR2.1	MR2.2	MR3.1	MR3.2
Wmc	−0.1001	0.0000	0.0000	0.0000	0.0000	0.0000
Dit	0.0217	0.0000	0.0000	0.0000	0.0000	0.0000
Noc	−0.0045	0.0000	0.0000	0.0000	0.0000	0.0000
Cbo	−0.0226	0.0000	0.0000	0.0000	0.0000	0.0000
Rfc	−0.1242	0.0000	0.0000	0.0000	0.0000	0.0000
Lcom	−0.0588	0.0000	0.0000	0.0000	0.0000	0.0000
Ca	−0.0086	0.0000	0.0000	0.0000	0.0000	0.0000
Ce	−0.0401	0.0000	0.0000	0.0000	0.0000	0.0000
Npm	−0.0691	0.0000	0.0000	0.0000	0.0000	0.0000
lcom3	−0.0091	0.0000	0.0000	0.0000	0.0000	0.0000
Loc	−0.1075	0.0000	0.0000	0.0000	0.0000	0.0000
Dam	−0.0469	0.0000	0.0000	0.0000	0.0000	0.0000
Moa	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Mfa	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Cam	−0.0937	0.0000	0.0000	0.0000	0.0000	0.0000
Ic	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Cbm	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Amc	−0.0667	0.0000	0.0000	0.0000	0.0000	0.0000
max_cc	0.0209	0.0000	0.0000	0.0000	0.0000	0.0000
avg_cc	−0.0276	0.0000	0.0000	0.0000	0.0000	0.0000
Bug	**0.4036**	**0.6667**	**0.6667**	**0.6667**	**0.6667**	**0.6667**

Table 19
Multiple Paired T-test results of features' contributions of predicting test case #44 of "velocity-1.6" between source and follow-up SDPSs using RF.

Paired T-test group	p-value $= P(T \leq t)$ two-tail (alpha=0.01, df=20)	t-score (df=20)	Reject H0: mean difference == 0 (p-value $\langle$ 0.01 or t-score $\rangle$ 2.8453[1])
Source vs MR1	0.2323	1.2318	No
Source vs MR2.1	0.2335	1.2286	No
Source vs MR2.2	0.2333	1.2291	No
Source vs MR3.1	0.2374	1.2179	No
Source vs MR3.2	0.2365	1.2205	No

4.3.2. Characteristic of algorithms

Table 2 shows that RF, KMeans(Lloyd), KMeans(Elkan) and CNN had non-zero IPRs. KMeans(Lloyd) and KMeans(Elkan) gave the worst IPC, IPR, ACC and MCC in this study (Tables 1–4), especially since both KMeans models had over 38 % IPR with all MRs. The above results can be explained by the random initialization of the KMeans algorithm (Xu et al., 2021). KMeans selects instances as initial center points randomly and assigns other instances to clusters where the center point is at nearest to that instance (Xu et al., 2021). Changing the order of data would affect the initial points selection, leading to inconsistent results, and this randomness of KMeans cannot be removed in the Python programs (SciKit-Learn, 2022).

RF consists of multiple decision trees, each trained on a randomly selected sample of the data (Sathe and Adamuthe, 2021). As a result, changing the order of the data can impact the sample selection for RF, leading to inconsistencies in predictions.

CNN require sufficient training data to make accurate predictions

Table 20
Multiple paired T-test results of features' contributions of predicting all selected inconsistent cases between source and follow-up SDPSs using RF[a].

Dataset	Paired T-test group p-value = $P(T \leq t)$ two-tail (alpha=0.01, df=20)					Reject H0: mean difference == 0 (p-value < 0.01)
	Source vs MR1	Source vs MR2.1	Source vs MR2.2	Source vs MR3.1	Source vs MR3.2	
data_redaktor	nil	Nil	0.2451	0.3292	nil	No
velocity-1.6	0.2323	0.2335	0.2333	0.2374	0.2365	No

[a] "nil" means there is no inconsistent case.

(Pachouly et al., 2022). However, our datasets were relatively small (Table 5). It is reasonable to conclude that altering the order of this limited training data can influence prediction consistency. Our results indicate that using descending-ordered MRs leads to fewer inconsistent predictions compared to reverse-ordered and ascending-ordered MRs (Table 2). Therefore, different arrangements of small training datasets can result in inconsistent predictions in CNN.

In summary, the impact of data ordering is closely tied to the characteristics of the algorithms. Maintaining the correct sequence of data is essential for ensuring consistent predictions in SDPS using KMeans (Lloyd), KMeans(Elkan), RF, or CNN.

4.3.3. Characteristic of datasets

As CNN has the highest IPC in the experiment (Table 1), we further analyse the identified inconsistent cases of CNN with the features' contributions and all the datasets to seek further understanding of the influence of the characteristics of datasets.

Twelve datasets with inconsistent cases of CNN in all five MRs (Table 5) are selected. We ran multiple paired T-tests between each feature's contributions to source and follow-up datasets. We found that six features' contributions are significantly different[7] or almost significantly different between source and follow-up datasets (Tables 21 and 22). These results show that these six features contributed the most to inconsistent predictions (Table 22).

As each of the six identified features was significantly different or almost significantly different between source and five MRs (Table 22), we ran a correlation analysis with all features in the twelve datasets. The results do not show that the six features are highly correlated with the features that are used in MRs (Section 3.2). Thus, the correlation is not the reason behind the significant differences.

We also ran an analysis on skewness and kurtosis on the twelve datasets, and the results show that averaged skewness and kurtosis are high in twelve datasets (Table 23), which means the datasets are asymmetry distributed (Hair Jr et al., 2021). In other words, they are imbalanced datasets. However, the six identified features are not the highest or the lowest in averaged skewness and kurtosis, so the skewness and kurtosis are also not the reason behind the significant differences. However, changing orders in imbalanced datasets could impact the accuracy of the machine learning model (Mao et al., 2022). Thus, the inconsistent predictions of CNN can be caused by imbalanced datasets.

Lastly, results show that using MRs can help understand the systems,

Table 21
Illustration of SHAP explains inconsistent predictions with the contributions of feature "rfc" in source and follow-up SDPSs (MR2.2) using CNN.

Dataset	Source	MR2.2
ant-1.7	0.0084	0.0001
camel-1.6	0.0424	−0.0038
data_prop-6	0.0054	−0.0004
jedit-3.2	0.0028	0.0102
jedit-4.2	0.0120	−0.0399
lucene-2.0	0.0180	−0.0021
poi-2.0	0.0238	−0.0072
synapse-1.2	−0.0074	−0.0233
velocity-1.6	−0.0449	−0.1428
xalan-2.4	−0.0281	−0.0421
xerces-1.2	0.0043	0.0020
xerces-1.3	0.0047	0.0088

Table 22
Multiple Paired T-test results of features that are significantly different in contributions between source and follow-up SDPSs using CNN.

Feature	Paired T-test group	p-value = $P(T \leq t)$ two-tail (alpha=0.01, df=11)	t-score (df=11)	effect size	Reject H0: mean difference == 0 (p-value 〈 0.01 or t-score 〉 3.1058)[a]
Bug	Source vs MR1	0.0316	2.4623	0.6446	No
Amc	Source vs MR2.1	0.0579	2.1170	−0.4730	No
Rfc	Source vs MR2.2	0.0196	2.7281	0.6912	No
avg_cc	**Source vs MR2.2**	**0.0186**	**2.7560**	**0.7978**	**Yes**[b]
Mfa	**Source vs MR3.1**	**0.0287**	**2.5155**	**−0.8716**	**Yes**[b]
Cbo	Source vs MR3.2	0.0589	2.1067	0.7166	No

[a] t-critical(alpha=0.01, df=11) = 3.1058, where it is paired t-test with 12 related datasets, so the degree of freedom is 11(12–1).

[b] Although the p-values are over 0.01, they are near 0.01 so we also calculate the effect sizes. The effect sizes are at 0.8 (Sullivan and Feinn, 2012) and we consider these features have significantly differences in contributions between source and follow-up SDPSs using CNN.

Table 23
Averaged Skewness and Kurtosis of features in twelve datasets that had inconsistent cases with all five MRs.

Feature	Skewness	Kurtosis
Wmc	4.2108	30.4518
Dit	1.0205	0.8528
Noc	9.5928	115.3016
Cbo	5.3190	54.6473
Rfc	3.3880	19.0036
Lcom	8.8197	110.5362
Ca	6.6293	72.6300
Ce	3.3428	21.3809
Npm	3.5506	22.4920
lcom3	0.6876	1.0955
Loc	5.7192	51.2587
Dam	−0.0065	−1.5677
Moa	4.2383	28.7767
Mfa	1.4254	19.7763
Cam	1.0219	2.4305
Ic	1.5147	2.3278
Cbm	2.8273	11.6413
Amc	6.0651	66.3136
max_cc	7.2371	82.5291
avg_cc	4.7383	42.0620
Bug	4.7014	34.6854

which agrees with (Zhou et al., 2020), and our proposed approach is a valuable tool for CNN, especially when trained datasets are scarce and only small-sized training data or imbalanced data are available. The proposed MRs can identify inconsistent predictions without being misled by high ACC and MCC, which is particularly beneficial for unsupervised machine learning algorithms in scenarios where no labelled dataset is available. In contrast, XAI can explain the inconsistent predictions with the features' contributions to reflect the level of inconsistency for further examination.

5. Threats to validity

This section discusses the threats to validity that are faced by this study.

5.1. Internal validity

The randomness of the systems is an internal threat that can cause result variations, and this study fixed random seeds to a fixed number for all SUTs to keep results reproducible. Another internal threat is the parameter configuration, as different input parameters cause different results. In this experiment, standard parameters and configurations were applied to all SUTs with cluster number was set to two, and the rest options were set to default values (Section 3.7). We build subject SUTs by "scikit-learn" library of Python and use Microsoft Excel to create the follow-up datasets. These technologies should be considered reliable since they have been widely used. White-box testing has been conducted to examine the correctness of the source codes. Finally, this study focused on identifying inconsistent results of SUTs, and the measurements (Section 3.8) of all SUTs were listed and checked to ensure accuracy.

5.2. External validity

We applied MRs inspired by Zhou et al. (2020)'s proven approach, so the potential threat to the external validity of this study is low. Another external threat is the generality of our approach. In this experiment, our MRs are generic and can be applied to any numeric dataset. This study used eighteen different software defect datasets of PROMISE, which are varied in size. Seven different SDPSs were used in this study, which ran seven hundred and fifty-six test trials (Section 3.3). The results should be generalized.

5.3. Construct validity

The independent variables of this study are the prediction outcomes of the same test datasets with different SDPSs that were trained by different follow-up datasets, which are the follow-up outputs, and the prediction outcomes of the same test datasets with SDPSs that were trained by source dataset, which are the source outputs. The dependent variable is the differences between the source outputs and follow-up outputs, as this study focuses on identifying the inconsistent predictions after data transformations. If there is no difference between source outputs and follow-up outputs, there is no inconsistent prediction, and the order of data does not cause inconsistent prediction in SDPSs. If there are inconsistent predictions after data transformations, this shows that changing the order of data can cause inconsistency in SDPSs. This correctly measures the intent of this study. Thus, the construct validity of the independent variables and dependent variables of this article should be acceptable.

6. Conclusion and future direction

The performance of SDPSs holds significant importance in the realm of software development, making it crucial to ensure the accuracy and correctness of the prediction outcomes. This study effectively demonstrated the practicality and feasibility of MT in identifying inconsistent predictions within SDPSs, while also showcasing the explanatory capabilities of XAI in elucidating the extent of inconsistency present in SDPSs. Moreover, our research findings suggest that the MRs proposed in this study have the potential to serve as an effective and valuable validation tool for SDPSs using SVM, LR and KNN algorithms. Additionally, our approach proves to be a valuable tool for SDPSs using RF, KMeans(Lloyd), KMeans(Elkan), and CNN algorithms, enabling the identification of inconsistent predictions and explaining the inconsistencies by showing the features' contributions that warrant further examination. Notably, we observed that varying the order of data within small-sized training datasets and imbalanced data can lead to inconsistent predictions in SDPSs due to the characteristics of KMeans (Lloyd), KMeans(Elkan), RF and CNN algorithms.

6.1. Limitations

This study applied XAI to explain the inconsistent predictions with features' contributions, so this study cannot use dimension reduction techniques, such as principal component analysis, to improve the speed of the process. Furthermore, the limited computer resources cannot support XAI in analysing the models with a large dataset for interpretation. Another limitation is that if the training data are time-ordered (sequential), the proposed MRs are not applicable as the MRs are related to changing the order of data and only apply to the data that can be altered in order.

Due to the MRs being only related to the order of data, this study uses the numeric features of datasets. Furthermore, the findings indicate that the sequence of the data is crucial for certain SDPS, which limits the effectiveness of the proposed MRs, as changing the order of training data does not demonstrate the concept of symmetry. Consequently, those SDPSs trained on source and follow-up training datasets cannot produce similar predictions with the same testing dataset, even if the contents of the training datasets remain intact. Thus, our approach is not effective and cannot exhibit the concept of symmetry because of the nature of the combination of dataset and SDPSs using KMeans, CNN and RF.

6.2. Future directions

For future investigations, it is recommended to extend our approach by incorporating additional or different MRPs into domains that employ various machine learning algorithms for identifying and explaining inconsistent predictions. As mentioned in Section 6.1, it is important to first study the characteristics of machine learning algorithms and datasets before choosing the MRPs to extend our approach.

Furthermore, exploring the relationship between data imbalance, dataset size, and MRPs could yield valuable insights to enhance the identification of inconsistent predictions and derive more effective MRPs. By addressing these avenues for future research, we can advance the understanding and application of our approach, improving the identification and explanation of inconsistent predictions in software defect prediction systems.

Declaration of generative AI in scientific writing

The authors declare that they have not used any generative artificial intelligence (AI) and AI-assisted technologies in the writing process.

CRediT authorship contribution statement

Pak Yuen Patrick Chan: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Data curation, Conceptualization, Formal analysis, Software. **Jacky Keung:** Writing – review & editing, Validation, Supervision. **Zhen Yang:** Writing – review & editing, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong and the research funds of the City University of Hong Kong (6000796, 9229109, 9229098, 9220103, 9229029), and the Natural Science Foundation of Shandong Province under Grant ZR2024QF093.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jss.2025.112449.

Data availability

The PROMISE datasets used in this study are publicly available at https://figshare.com/articles/dataset/Software_Defect_Prediction_Dataset/13536506?file=25981097, and other related data will be made available on request.

References

Ajorloo, S., Jamarani, A., Kashfi, M., Kashani, M.H., Najafizadeh, A., 2024. A systematic review of machine learning methods in software testing. Appl. Soft Comput. 162, 111805.

Arora, I., Saha, A., 2018. Software defect prediction: a comparison between artificial neural network and support vector machine. In: Paper presented at the Advanced Computing and Communication Technologies: Proceedings of the 10th ICACCT, 2016.

Bagheri, H., Kang, E., Mansoor, N., 2020. Synthesis of assurance cases for software certification. In: Paper presented at the 2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER).

Bala, Y.Z., Samat, P.A., Sharif, K.Y., Manshor, N., 2022. Current software defect prediction: a systematic review. In: Paper presented at the 2022 Applied Informatics International Conference (AiIC).

Batool, I., Khan, T.A., 2022. Software fault prediction using data mining, machine learning and deep learning techniques: a systematic literature review. Comput. Electr. Eng. 100, 107886.

Chen, H., Zhang, Z., Yin, W., Zhao, C., Wang, F., Li, Y., 2022. A study on depth classification of defects by machine learning based on hyper-parameter search. Measurement 189, 110660.

Chen, T.Y., Cheung, S.C., & Yiu, S.M. (2020). Metamorphic testing: a new approach for generating next test cases. *arXiv [Cs.SE]*. Retrieved from http://arxiv.org/abs/2002.12543.

Chicco, D., Jurman, G., 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genom. 21 (1), 6. https://doi.org/10.1186/s12864-019-6413-7.

Cho, E., Shin, Y.-J., Hyun, S., Kim, H., Bae, D.-H., 2022. Automatic generation of metamorphic relations for a cyber-physical system-of-systems using genetic algorithm. In: Paper presented at the 2022 29th Asia-Pacific Software Engineering Conference (APSEC).

Chua, K.K.A., Bae, D.-H., Jee, E., 2021. Metamorphic testing for reliability in system of systems. In: Paper presented at the 2021 28th Asia-Pacific Software Engineering Conference (APSEC).

Deng, Y., Zheng, X., Zhang, T., Liu, H., Lou, G., Kim, M., Chen, T.Y., 2023. A declarative metamorphic testing framework for autonomous driving. IEEE Trans. Softw. Eng. 49 (4), 1964–1982.

Elish, K.O., Elish, M.O., 2008. Predicting defect-prone software modules using support vector machines. J. Syst. Softw. 81 (5), 649–660.

Ellis, J.D., Iqbal, R., Yoshimatsu, K., 2021. Verification of the neural network training process for spectrum-based chemical substructure prediction using metamorphic testing. J. Comput. Sci. 55, 101456.

Fan, M., Wei, J., Jin, W., Xu, Z., Wei, W., Liu, T., 2022. One step further: evaluating interpreters using metamorphic testing. In: Paper presented at the Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis.

Ferenc, R., Tóth, Z., Ladányi, G., Siket, I., Gyimóthy, T., 2018. A public unified bug dataset for java. In: Paper presented at the Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering.

Hair Jr, J.F., Hult, G.T.M., Ringle, C.M., Sarstedt, M., 2021. A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM). SAGE Publications.

Hauser, K., Kurz, A., Haggenmüller, S., Maron, R.C., von Kalle, C., Utikal, J.S., Sergon, M., 2022. Explainable artificial intelligence in skin cancer recognition: a systematic review. Eur. J. Cancer 167, 54–69.

He, P., Li, B., Liu, X., Chen, J., Ma, Y., 2015. An empirical study on software defect prediction with a simplified metric set. Inf. Softw. Technol. 59, 170–190.

Hemández-Molinos, M.J., Sánchez-García, Á.J., Barrientos-Martínez, R.E., 2021. Classification algorithms for software defect prediction: a systematic literature review. In: Paper presented at the 2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT).

Jiang, L., Cai, Z., Wang, D., Jiang, S., 2007. Survey of improving k-nearest-neighbor for classification. In: Paper presented at the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007).

Jureczko, M., Madeyski, L., 2010. Towards identifying software project clusters with regard to defect prediction. In: Paper presented at the Proceedings of the 6th International Conference on Predictive Models in Software Engineering.

Karim, S., Warnars, H.L.H.S., Gaol, F.L., Abdurachman, E., Soewito, B., 2017. Software metrics for fault prediction using machine learning approaches: a literature review with PROMISE repository dataset. In: Paper presented at the 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom).

Kumar, R., Chaturvedi, A., Kailasam, L., 2022. An unsupervised software fault prediction approach using threshold derivation. IEEE Trans. Reliab. 71 (2), 911–932.

Laradji, I.H., Alshayeb, M., Ghouti, L., 2015. Software defect prediction using ensemble learning on selected features. Inf. Softw. Technol. 58, 388–402.

Lin, H., Han, J., Wu, P., Wang, J., Tu, J., Tang, H., Zhu, L., 2024. Machine learning and human-machine trust in healthcare: a systematic survey. CAAI Trans. Intell. Technol. 9 (2), 286–302.

Liu, X., Li, Z., Zou, J., Tong, H., 2022. An empirical study on multi-source cross-project defect prediction models. In: *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, pp. 318–327.

Liu, W., Zhu, Y., Chen, X., Gu, Q., Wang, X., Gu, S., 2021. S ^{2} LMMD: cross-project software defect prediction via statement semantic learning and maximum mean discrepancy. In: *2021 28th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 369-379). IEEE..

Loh, H.W., Ooi, C.P., Seoni, S., Barua, P.D., Molinari, F., Acharya, U.R., 2022. Application of explainable artificial intelligence for healthcare: a systematic review of the last decade (2011–2022). Comput. Methods Progr. Biomed. 226, 107161.

Luu, Q.-H., Liu, H., Chen, T.Y., Vu, H.L., 2024. A sequential metamorphic testing framework for understanding autonomous vehicle's decisions. IEEE Trans. Intell. Veh.

Mao, Y., Gupta, V., Wang, K., Liao, W.-k., Choudhary, A., Agrawal, A., 2022. To shuffle or not to shuffle: mini-batch shuffling strategies for multi-class imbalanced classification. In: Paper presented at the 2022 International Conference on Computational Science and Computational Intelligence (CSCI).

Molnar, C., 2020. Interpretable Machine Learning. Lulu. com.

Moore, D.S., McCabe, G.P., Craig, B.A., 2009. Introduction to the Practice of Statistics, 4. WH Freeman, New York.

NationalScienceFoundation. (2022). NSF 22-502: national Artificial Intelligence (AI) Research Institutes accelerating: research, transforming society, and growing the American workforce - Program solicitation. Retrieved from https://new.nsf.gov/funding/opportunities/national-artificial-intelligence-research-institutes/nsf22-502/solicitation.

Pachouly, J., Ahirrao, S., Kotecha, K., Selvachandran, G., Abraham, A., 2022. A systematic literature review on software defect prediction using artificial intelligence: datasets, data validation methods, approaches, and tools. Eng. Appl. Artif. Intell. 111, 104773.

Phan, A.V., Le Nguyen, M., 2017. Convolutional neural networks on assembly code for predicting software defects. In: Paper presented at the 2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES).

Poth, A., Meyer, B., Schlicht, P., Riel, A., 2020. Quality assurance for machine learning – an approach to function and system safeguarding. In: Paper presented at the 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS).

Pradhan, B., Dikshit, A., Lee, S., Kim, H., 2023. An explainable AI (XAI) model for landslide susceptibility modeling. Appl. Soft Comput. 142, 110324.

Pugh, S., Raunak, M.S., Kuhn, D.R., Kacker, R., 2019. Systematic testing of post-quantum cryptographic implementations using metamorphic testing. In: Paper presented at the 2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET).

Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., Tonella, P., 2020. Testing machine learning based systems: a systematic mapping. Empir. Softw. Eng. 25 (6), 5193–5254.

Saputra, A., Suharjito, 2019. Fraud detection using machine learning in e-commerce. Int. J. Adv. Comput. Sci. Appl. 10 (9), 332–339.

Sathe, M.T., Adamuthe, A.C., 2021. Comparative study of supervised algorithms for prediction of students' Performance. Int. J. Mod. Educ. Comput. Sci. 13 (1), 1–21.

SciKit-Learn. (2022). sklearn.Cluster.KMeans. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

SciKit-Learn. (2024). 1.4. Support Vector machines. Retrieved from https://scikit-earn.org/stable/modules/svm.html.

Segura, S., Durán, A., Troya, J., Ruiz-Cortés, A., 2019. Metamorphic relation patterns for query-based systems. In: Paper presented at the 2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET).

Segura, S., Towey, D., Zhou, Z.Q., Chen, T.Y., 2018. Metamorphic testing: testing the untestable. IEEE Softw. 37 (3), 46–53.

Sheu, R.-K., Pardeshi, M.S., 2022. A survey on medical explainable AI (XAI): recent progress, explainability approach, Human interaction and scoring system. Sensors 22 (20), 8068.
Sreedevi, E., PremaLatha, V., Sivakumar, S., Nayak, S.R., 2019. A comparative study on new classification algorithm using NASA MDP datasets for software defect detection. In: Paper presented at the 2019 International Conference on Intelligent Sustainable Systems (ICISS).
Stacy, B., Hauzel, J., Lindvall, M., Porter, A., Pop, M., 2022. Metamorphic testing in Bioinformatics software: a case study on metagenomic assembly. In: Paper presented at the 2022 IEEE/ACM 7th International Workshop on Metamorphic Testing (MET).
Sullivan, G.M., Feinn, R., 2012. Using effect size—Or why the P value is not enough. J. Grad. Med. Educ. 4 (3), 279–282. https://doi.org/10.4300/jgme-D-12-00156.1.
Sun, C.-a., Liu, B., Fu, A., Liu, Y., Liu, H., 2022. Path-directed source test case generation and prioritization in metamorphic testing. J. Syst. Softw. 183, 111091.
Tang, S., Huang, S., Zheng, C., Liu, E., Zong, C., Ding, Y., 2021. A novel cross-project software defect prediction algorithm based on transfer learning. Tsinghua Sci. Technol. 27 (1), 41–57.
Torikoshi, Y., Nishi, Y., Takahashi, J., 2023. Sensitive region-based Metamorphic testing framework using explainable AI. IEEE, pp. 25–30.
Venkatkumar, I.A., Shardaben, S.J.K., 2016. Comparative study of data mining clustering algorithms. In: Paper presented at the 2016 International Conference on Data Science and Engineering (ICDSE).
Wang, H., Zhuang, W., Zhang, X., 2021. Software defect prediction based on gated hierarchical LSTMs. IEEE Trans. Reliab. 70 (2), 711–727.
Xiao, D., LIU, Z., Yuan, Y., Pang, Q., Wang, S., 2022. Metamorphic testing of deep learning compilers. Proc. ACM Meas. Anal. Comput. Syst. 6 (1), 15. Article.
Xie, X., Ho, J.W.K., Murphy, C., Kaiser, G., Xu, B., Chen, T.Y., 2011. Testing and validating machine learning classifiers by metamorphic testing. J. Syst. Softw. 84 (4), 544–558.
Xie, X., Zhang, Z., Chen, T.Y., Liu, Y., Poon, P.-L., Xu, B., 2020. METTLE: a metamorphic testing approach to assessing and validating unsupervised machine learning systems. IEEE Trans. Reliab. 69 (4), 1293–1322.
Xu, J., Ai, J., Liu, J., Shi, T., 2022. ACGDP: an augmented code graph-based system for software defect prediction. IEEE Trans. Reliab. 71 (2), 850–864.
Xu, Z., Li, L., Yan, M., Liu, J., Luo, X., Grundy, J., Zhang, X., 2021. A comprehensive comparative study of clustering-based unsupervised defect prediction models. J. Syst. Softw. 172, 110862.
Ying, Z., Towey, D., Bellotti, A., Zhou, Z.Q., Chen, T.Y., 2021. Preparing SQA professionals: metamorphic relation patterns, exploration, and testing for big data. In: Proceedings of the International Conference on Open and Innovation Education (ICOIE 2021), pp. 22–30.
Yuan, Y., Pang, Q., Wang, S., 2022. Unveiling hidden DNN defects with decision-based metamorphic testing. In: Paper presented at the 37th IEEE/ACM International Conference on Automated Software Engineering.
Zebin, T., Rezvy, S., Luo, Y., 2022. An explainable ai-based intrusion detection system for dns over https (doh) attacks. IEEE Trans. Inf. Forens. Sec. 17, 2339–2349.
Zhang, M., Keung, J.W., Chen, T.Y., Xiao, Y., 2021. Validating class integration test order generation systems with Metamorphic Testing. Inf. Softw. Technol. 132, 106507.
Zhang, T., Du, Q., Xu, J., Li, J., Li, X., 2020. Software defect prediction and localization with attention-based models and ensemble learning. In: Paper presented at the 2020 27th Asia-Pacific Software Engineering Conference (APSEC).
Zhang, X., Chan, F.T., Yan, C., Bose, I., 2022. Towards risk-aware artificial intelligence and machine learning systems: an overview. Decis. Support Syst. 159, 113800.
Zheng, W., Shen, T., Chen, X., Deng, P., 2022. Interpretability application of the just-in-time software defect prediction model. J. Syst. Softw. 188, 111245.
Zhou, Z.Q., Sun, L., Chen, T.Y., Towey, D., 2020. Metamorphic relations for enhancing system understanding and use. IEEE Trans. Softw. Eng. 46 (10), 1120–1154.