Physical Education, Health and Social Sciences

https://journal-of-social-education.org

E-ISSN: 2958-5996

P-ISSN: 2958-5988

# Explainable AI-Based Early Sprint Risk Prediction in Agile Software Projects Using Machine Learning Models

**Attiya Afzal** [1]

[1] MPhil, Currently Studying, Department of Software Engineering University of Sargodha, Pakistan. Email: attiagondal555@gmail.com

**Abstract**

Agile software development is widely used in modern software projects because it supports flexibility, continuous feedback, and iterative delivery. However, Agile projects still face several sprint-level risks such as reduced sprint velocity, frequent backlog changes, increasing defect count, poor task completion ratio, and excessive team workload. These risks can negatively affect sprint success if they are not identified at an early stage. Traditional risk management methods mainly depend on manual monitoring and project manager experience, which may not be sufficient for timely and accurate risk detection. Moreover, many existing machine learning-based prediction systems work as black-box models and do not clearly explain the reasons behind their predictions.

To address this problem, this research proposes an Explainable Artificial Intelligence-based early sprint risk prediction framework for Agile software projects. The proposed approach combines machine learning models with explainability techniques to predict sprint risk levels and provide understandable explanations for project managers. The system uses sprint-related features such as sprint velocity, backlog churn rate, defect count, task completion ratio, team workload, and story points completed. The target variable is sprint risk level, classified as Low, Medium, or High.

In this study, multiple machine learning models including Logistic Regression, Random Forest, Support Vector Machine, and XGBoost are applied and compared. The dataset is preprocessed through missing value handling, normalization, and encoding. Model performance is evaluated using accuracy, precision, recall, F1-score, and ROC-AUC. For explainability, SHAP is used to identify the contribution of each feature in the prediction process.

The key contribution of this research is the development of a transparent and interpretable sprint risk prediction framework that not only predicts early sprint risks but also explains the main factors responsible for those risks. This can help Agile project managers make timely, data-driven, and understandable decisions to improve sprint planning, reduce project failure, and enhance overall software project performance.

## 2. Introduction

### 2.1 Background

Agile software development has become one of the most widely adopted approaches in software engineering because of its flexibility, iterative nature, and strong focus on customer collaboration. Unlike traditional development models such as the Waterfall model, Agile promotes adaptive planning, continuous improvement, and rapid delivery of working software in small increments (Beck et al., 2001). Agile methodologies are designed to respond effectively to changing customer requirements, which makes them especially suitable for dynamic and fast-changing project environments. Over the years, Agile has been increasingly adopted by organizations seeking to improve software quality, shorten development cycles, and enhance communication among team members (Dybå & Dingsøyr, 2008).

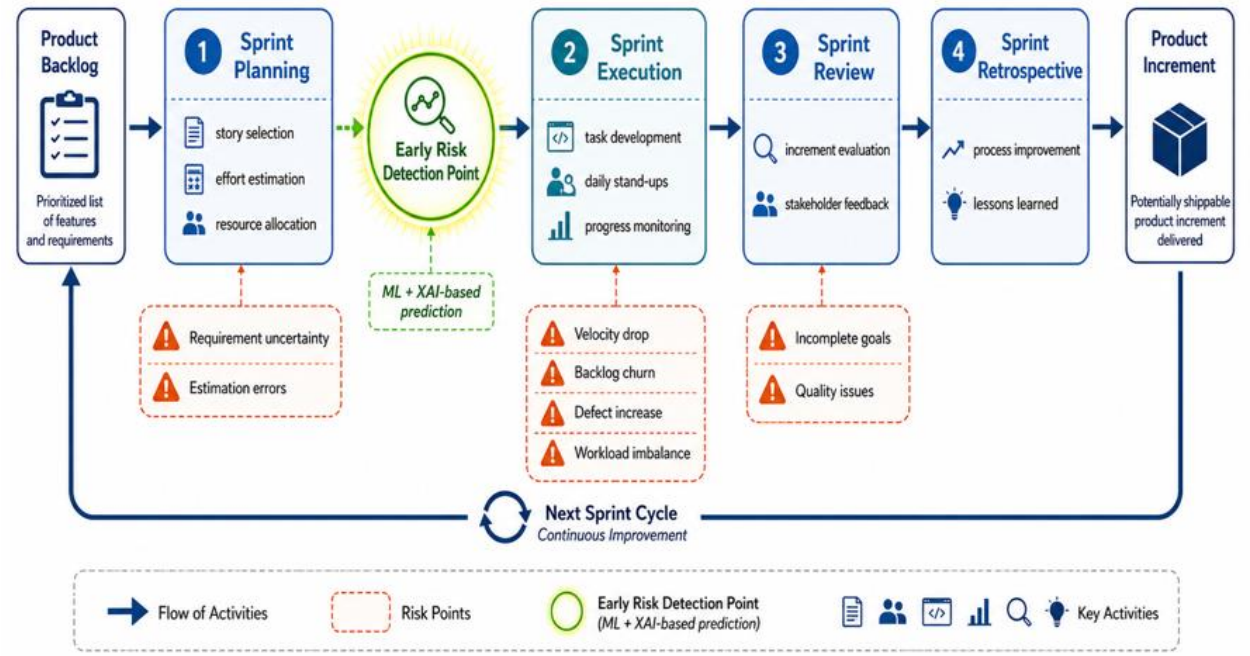
Among the different Agile frameworks, Scrum is one of the most commonly used. Scrum organizes software development into short, time-boxed iterations called sprints, which usually last between one and four weeks (Schwaber & Sutherland, 2020). Each sprint includes a set of planned tasks that are selected from the product backlog during sprint planning. At the end of the sprint, the team delivers a potentially shippable product increment. A typical sprint cycle includes sprint planning, sprint execution, daily stand-up meetings, sprint review, and sprint retrospective. This structured yet flexible process helps teams monitor progress regularly and adjust their work according to project needs.

The sprint concept is central to Agile project management because it allows teams to divide complex projects into manageable units of work. Each sprint has specific goals, deliverables, and deadlines. During sprint planning, the development team estimates story points, selects user stories, and allocates tasks according to team capacity. During execution, developers work collaboratively to complete the selected backlog items. After execution, sprint review and retrospective sessions are conducted to assess progress, identify challenges, and improve future performance (Schwaber & Sutherland, 2020). Because sprint outcomes directly affect overall project success, monitoring sprint health and detecting possible risks early are highly important.

Despite its advantages, Agile development is not free from risk. Agile projects often face several sprint-level risk issues that can negatively affect project performance if not identified and managed in time. Common risks include low sprint velocity, backlog churn, incomplete tasks, increasing defect count, poor task completion ratio, unrealistic effort estimation, and uneven team workload. In addition, frequent requirement changes and communication problems within cross-functional teams may further increase the probability of sprint failure (Boehm, 1991; Dybå & Dingsøyr, 2008). These risks can lead to missed sprint goals, delayed releases, reduced software quality, and dissatisfaction among stakeholders.

In many Agile teams, risk identification still depends largely on project manager experience, team discussions, and manual progress tracking. Although these traditional methods are useful, they may not be sufficient in data-rich and complex software environments. The increasing availability of sprint-related data, such as velocity, defect count, backlog changes, and task completion records, creates an opportunity to use data-driven methods for risk prediction. Machine learning techniques have shown strong potential in software engineering tasks because they can identify hidden patterns in historical data and generate predictive insights for better decision-making (Khoshgoftaar & Allen, 2003). Therefore, the use of machine learning in Agile sprint risk management can provide a more proactive and intelligent solution.

**Figure 1. Agile Software Development Lifecycle with Risk Points in Sprint Phases**




This figure illustrates the Agile sprint lifecycle, including sprint planning, sprint execution, and sprint review/retrospective, while highlighting key risk points that may arise during each phase. The figure also marks the early risk detection point, where machine learning-based prediction can be applied to identify potential sprint risks before they significantly affect project performance.

**2.2 Problem Statement**

Although Agile methodologies support iterative monitoring and continuous feedback, many software teams still struggle to identify sprint risks at an early stage. In practice, risks are often recognized only after they begin to affect team performance or sprint outcomes. This delayed identification reduces the ability of project managers to take timely corrective action. As a result, projects may experience reduced productivity, poor sprint outcomes, increased defects, and missed delivery commitments. Therefore, one major problem in Agile project management is the lack of early risk prediction.

Another important challenge is the lack of explainability in predictive systems. While machine learning models can improve the accuracy of risk prediction, many of these models operate as black-box systems, meaning they provide predictions without clearly explaining the reasoning behind them (Adadi & Berrada, 2018). In software project management, decision-makers need not only accurate predictions but also understandable explanations. For example, if a sprint is predicted as high risk, the project manager must know whether the prediction is mainly influenced by low velocity, high backlog churn, increasing defects, or poor task completion ratio. Without such explainability, project managers may hesitate to trust or use the system's recommendations in real-world decision-making.

This study addresses both issues by proposing an approach that combines machine learning with Explainable Artificial Intelligence. By integrating predictive models with explainability methods such as SHAP, the proposed system aims to predict sprint risk levels early and provide transparent explanations for each prediction (Lundberg & Lee, 2017). In this way, the system can support more reliable, interpretable, and actionable Agile risk management.

**2.3 Research Objectives**

The main objective of this research is to develop an intelligent and explainable framework for early sprint risk prediction in Agile software projects. To achieve this aim, the study focuses on two major components: machine learning-based prediction and XAI-based explanation.

First, this research aims to apply machine learning techniques to classify sprint risk levels based on sprint-related features such as sprint velocity, backlog churn rate, defect count, task completion ratio, team workload, and story points completed. Multiple machine learning models, including Logistic Regression, Random Forest, Support Vector Machine, and XGBoost, are used to evaluate predictive performance and identify the most effective model for sprint risk prediction.
Second, the study aims to improve the interpretability of prediction results through Explainable Artificial Intelligence. For this purpose, SHAP is used to analyze feature importance and explain the contribution of each input variable to the predicted sprint risk level (Lundberg & Lee, 2017). This will help project managers understand why a sprint is considered low, medium, or high risk.
More specifically, the objectives of this research are as follows:

1. To analyze the role of sprint-related data in identifying Agile project risks.
2. To develop a machine learning-based model for early sprint risk prediction.
3. To compare the predictive performance of Logistic Regression, Random Forest, SVM, and XGBoost.
4. To apply SHAP-based explainability for interpreting model outputs.
5. To identify the most influential features contributing to sprint risk.
6. To support Agile project managers in making timely, transparent, and data-driven decisions.

Overall, this research seeks to bridge the gap between prediction accuracy and model interpretability in Agile software risk management. By combining machine learning with explainable AI, the study contributes toward a more practical and trustworthy approach for improving sprint-level decision-making and overall project success.

## 3. Literature Review

### 3.1 Agile Risk Management Studies

Agile software development has been widely studied as an adaptive and iterative approach for managing software projects in uncertain and changing environments. Agile methods focus on customer collaboration, short development cycles, continuous feedback, and frequent delivery of working software. Compared with traditional plan-driven models, Agile provides flexibility and allows teams to respond quickly to requirement changes (Dybå & Dingsøyr, 2008). However, flexibility does not remove risks completely. Instead, Agile projects face different types of sprint-level risks that must be identified and controlled throughout the project lifecycle.
Risk management is a critical part of software project success. Boehm (1991) emphasized that identifying and managing risks early in the software development process can reduce long-term project costs and prevent major software failures. In Agile projects, risks may appear in different forms, such as unclear requirements, unstable backlog items, poor effort estimation, low team velocity, communication issues, increased defects, and incomplete sprint goals. These risks can directly affect sprint performance and reduce the quality of the final product.
Previous studies show that Agile practices can help reduce some project risks because teams work in short iterations and receive regular feedback from customers and stakeholders. Sprint planning, daily stand-up meetings, sprint reviews, and retrospectives allow teams to monitor progress and identify problems more frequently (Schwaber & Sutherland, 2020). However, many Agile teams still depend on manual risk identification methods such as team discussions, project manager experience, and progress reports. These methods are useful but may not always detect hidden risks at an early stage.
In sprint-based development, risk can emerge during different sprint phases. During sprint planning, risks may occur because of poor requirement understanding, inaccurate effort estimation, or unrealistic sprint goals. During sprint execution, risks may include velocity drop, backlog churn, increasing defect count, and workload imbalance. During sprint review, incomplete user stories, quality issues, and stakeholder dissatisfaction may become visible. The problem is that many of these risks are recognized only after they have already affected sprint

outcomes. Therefore, early sprint risk prediction is necessary for improving Agile project control and decision-making.

Agile risk management studies highlight the need for proactive risk identification instead of reactive risk handling. Traditional risk management approaches usually identify risks after symptoms appear, while data-driven prediction methods can help project managers detect risk patterns before sprint failure occurs. This creates an opportunity to integrate sprint metrics with machine learning techniques for early prediction. However, existing Agile risk management research often focuses on general risk categories and management practices rather than automated sprint-level prediction systems. This shows a clear need for intelligent models that can analyze sprint data and predict risk levels before the sprint is completed.

**3.2 Machine Learning in Software Engineering**

Machine learning has become an important area in software engineering because it can analyze large amounts of project data and identify patterns that are difficult to detect manually. Machine learning techniques have been applied in several software engineering tasks, including defect prediction, effort estimation, software quality prediction, software maintenance, bug classification, vulnerability detection, and project management decision support (Zhang, 2003). These applications show that machine learning can support software teams by improving prediction accuracy and reducing dependence on manual judgment.

In software quality and defect prediction, machine learning models have been used to classify software modules as defective or non-defective based on historical data. Hall et al. (2012) conducted a systematic literature review on software fault prediction and found that prediction models can help direct testing effort, reduce cost, and improve software quality. Similar principles can be applied to Agile sprint risk prediction because sprint-level data also contains useful indicators of future performance. For example, sprint velocity, defect count, backlog churn, completion ratio, and team workload can be used as input features for predicting whether a sprint is likely to be low, medium, or high risk.

Several machine learning algorithms are commonly used in software engineering prediction tasks. Logistic Regression is often used as a baseline model because it is simple and interpretable. Random Forest is an ensemble learning method that can handle nonlinear relationships and feature interactions. Support Vector Machine is useful for classification problems where clear decision boundaries are needed. XGBoost is a powerful boosting algorithm that often provides strong predictive performance in structured datasets. These models can be compared using evaluation metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

Machine learning can be especially useful in Agile project environments because Agile teams generate continuous project data during every sprint. Sprint-related metrics such as story points completed, number of defects, team capacity, task completion percentage, and backlog changes can provide valuable signals about sprint health. By learning from past sprint records, machine learning models can classify current or future sprint risk levels. This can help project managers make timely decisions, such as adjusting workload, clarifying requirements, improving resource allocation, or controlling backlog changes.

However, the use of machine learning in Agile risk prediction also has limitations. First, model performance depends on the quality and availability of sprint data. Incomplete, inconsistent, or biased data can reduce prediction accuracy. Second, some machine learning models may provide high accuracy but limited explanation. For example, ensemble models such as Random Forest and XGBoost can produce strong predictions, but their internal decision-making process may be difficult for project managers to understand. Therefore, machine learning alone is not sufficient for practical Agile risk management. The prediction system must also be explainable so that project managers can understand why a sprint is classified as risky.

**3.3 Explainable AI in Predictive Systems**

Explainable Artificial Intelligence is an important research area that aims to make AI and machine learning systems more transparent, interpretable, and trustworthy. Many high-performing machine learning models are considered black-box models because they provide predictions without clearly explaining the reasoning behind those predictions. In critical decision-making domains, users need to understand why a model produced a specific output before they can trust and apply it (Adadi & Berrada, 2018).

In software project management, explainability is particularly important because project managers must justify their decisions to development teams, stakeholders, and clients. A risk prediction model that only says "High Risk" may not be enough for decision-making. The manager also needs to know the main reasons behind the high-risk prediction. For example, a sprint may be classified as high risk because of a sudden velocity drop, increased backlog churn, rising defect count, or low task completion ratio. If the model explains these contributing factors, the project manager can take targeted corrective action.

Several explainability methods have been proposed for interpreting machine learning predictions. LIME explains individual predictions by creating a simple interpretable model around a specific prediction (Ribeiro et al., 2016). SHAP is another widely used method based on Shapley values from cooperative game theory. SHAP explains predictions by assigning an importance value to each feature, showing how much that feature contributes to the final prediction (Lundberg & Lee, 2017). In the context of sprint risk prediction, SHAP can identify whether features such as velocity, defect count, workload, backlog churn, or completion ratio are pushing the prediction toward low, medium, or high risk.

Explainable AI can provide both global and local explanations. Global explanations show which features are generally most important across the whole dataset. Local explanations show why a specific sprint received a particular risk prediction. For Agile project management, both types of explanations are useful. Global explanations can help organizations understand the common drivers of sprint risk, while local explanations can help teams take action in a specific sprint.

Recent software engineering research has also emphasized the importance of XAI for increasing the trust and adoption of AI-based tools. Mohammadkhani et al. (2023) found that XAI in software engineering is still an emerging area, with many opportunities for applying explainability methods beyond traditional defect prediction tasks. This is important because many existing XAI applications in software engineering focus on maintenance and defect prediction, while less attention has been given to software project management and Agile sprint risk prediction.

Therefore, combining machine learning with explainable AI can make sprint risk prediction more useful and practical. Machine learning can provide predictive power, while XAI can provide transparency and trust. This combination allows Agile managers to not only know the predicted risk level but also understand the main factors that caused that prediction.

**3.4 Research Gap**

The reviewed literature shows that Agile software development has strong potential for managing changing requirements and improving project flexibility. However, Agile teams still face sprint-level risks such as backlog instability, inaccurate estimation, velocity reduction, defect increase, incomplete goals, and workload imbalance. Existing Agile risk management studies mainly focus on risk identification, risk categories, and management practices. They provide useful guidelines, but many of them do not offer automated early prediction at the sprint level.

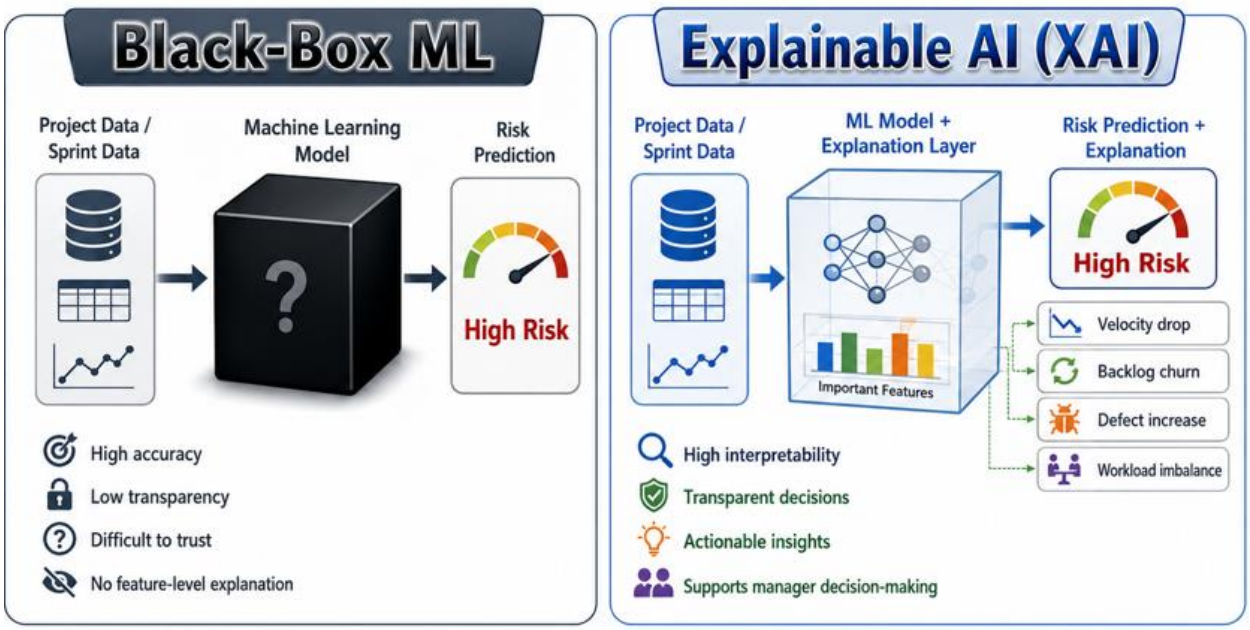
The literature on machine learning in software engineering shows that ML models have been successfully used for defect prediction, quality prediction, and effort estimation. These studies prove that software engineering data can be used for predictive decision-making. However, most existing ML-based studies focus on code-level or defect-level prediction rather than sprint-level Agile risk prediction. There is still limited research on using sprint metrics such as velocity,

backlog churn, defect count, task completion ratio, team workload, and story points completed for early sprint risk classification.

Another major gap is the lack of explainability. Many machine learning models can provide accurate predictions, but they do not clearly explain why a specific prediction was made. In Agile project management, this is a serious limitation because project managers need understandable explanations before taking corrective actions. A black-box model may predict that a sprint is high risk, but without explanation, the project manager may not know whether the problem is related to defects, team workload, incomplete tasks, or requirement changes.

Based on these gaps, this research proposes an Explainable AI-based early sprint risk prediction framework for Agile software projects. The proposed study combines machine learning models with SHAP-based explanation to classify sprint risk levels and identify the most influential sprint risk factors. This research fills the gap by focusing specifically on sprint-level prediction, early risk detection, and explainable decision support for Agile project managers.

**Figure 2. Comparison of Black-Box ML vs Explainable AI in Software Project Prediction**




| Aspect | Black-Box ML | vs. | Explainable AI (XAI) |
|---|---|---|---|
| Output | Prediction only | ⟷ | Prediction + Explanation |
| Trust | Low trust | ⟷ | High trust |
| Transparency | Opaque (Black Box) | ⟷ | Transparent (Interpretable) |

This figure compares traditional black-box machine learning prediction with Explainable AI-based prediction in software project management. The black-box model provides only the predicted risk level, while the XAI-based model provides both the predicted risk level and feature-level explanations such as velocity drop, backlog churn, defect increase, and workload imbalance.

**4. Research Gap and Contribution**

Previous studies in software engineering have shown that machine learning models can achieve good prediction performance in areas such as defect prediction, software quality estimation, and project risk analysis. Models such as Random Forest, Support Vector Machine, and XGBoost are often effective because they can learn complex patterns from historical software project data (Hall et al., 2012). However, many of these models mainly focus on improving prediction accuracy and

do not provide clear explanations for their decisions. As a result, they often work as black-box systems, where the output is available but the reasoning behind the output is difficult to understand (Adadi & Berrada, 2018).

In Agile software projects, prediction accuracy alone is not sufficient. Agile project managers need interpretable and actionable information because they must make timely decisions during sprint planning and sprint execution. For example, if a sprint is predicted as high risk, the manager should know whether the risk is caused by velocity drop, backlog churn, defect increase, low task completion ratio, or workload imbalance. Without such explanation, managers may not fully trust the prediction model or may not know which corrective action should be taken.

The major research gap identified in this study is the limited availability of sprint-level Explainable AI systems for Agile risk prediction. Existing studies mostly focus on general software defect prediction, effort estimation, or overall project risk management, but fewer studies focus specifically on early sprint-level risk prediction using both machine learning and explainability. Moreover, many existing prediction models do not integrate SHAP-based feature explanation to show how each sprint metric contributes to the final risk classification.

Therefore, this research contributes by proposing an Explainable AI-based early sprint risk prediction framework for Agile software projects. The proposed framework combines machine learning models with SHAP explanations to classify sprint risk levels as Low, Medium, or High and to identify the most influential risk factors. This contribution can support Agile managers in making transparent, data-driven, and timely decisions to reduce sprint failure and improve software project performance (Lundberg & Lee, 2017).

## 5. Methodology

This section explains the methodology used to develop the proposed Explainable AI-based early sprint risk prediction framework for Agile software projects. The main purpose of this methodology is to predict sprint risk levels at an early stage and explain the reasons behind each prediction. The proposed framework combines machine learning models with Explainable Artificial Intelligence techniques so that Agile project managers can receive both prediction results and understandable explanations. The methodology consists of five major stages: system architecture, dataset description, data preprocessing, machine learning model training, and SHAP-based explanation.

### 5.1 System Architecture

The proposed system architecture is designed to predict sprint-level risk in Agile software projects using machine learning and Explainable AI. Agile projects generate continuous sprint-related data during planning, execution, review, and retrospective phases. These data points can provide useful indicators about the health of a sprint. For example, low sprint velocity, high defect count, frequent backlog changes, and poor task completion ratio can indicate possible sprint failure or project delay. Previous software engineering research has shown that historical project data can support predictive decision-making and improve software quality management (Hall et al., 2012). Therefore, this research uses sprint-level project data as input for risk prediction.

The architecture of the proposed framework follows a sequential pipeline: input sprint data → data preprocessing → machine learning models → SHAP explanation → output dashboard. In the first stage, sprint-related data is collected from Agile project management sources such as Jira, GitHub, project tracking tools, or a simulated Agile sprint dataset. The input data includes sprint velocity, backlog churn rate, defect count, task completion ratio, team workload, and story points completed. These features are selected because they are closely related to sprint performance and Agile project risk.

After collecting the input data, the second stage is data preprocessing. In real software projects, raw data is often incomplete, inconsistent, or not directly suitable for machine learning models. Therefore, the dataset is cleaned by handling missing values, normalizing numerical values, and

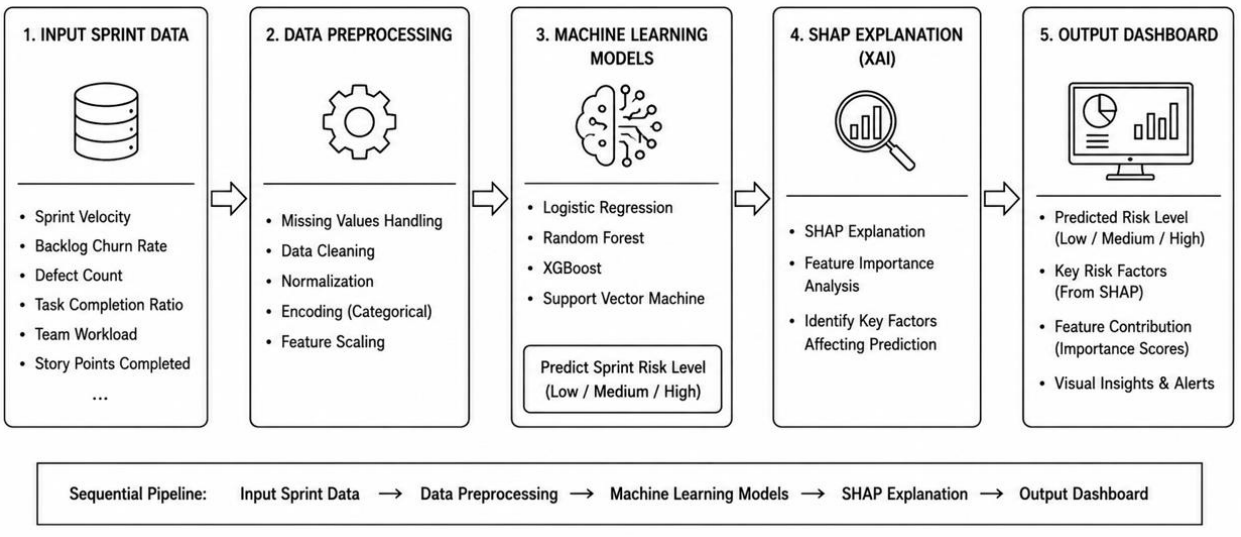
encoding categorical variables. This step is important because machine learning algorithms require clean and structured data to produce reliable predictions.

The third stage of the architecture is machine learning model development. In this study, multiple machine learning models are trained and compared, including Logistic Regression, Random Forest, XGBoost, and Support Vector Machine. These models are used to classify sprint risk into three categories: Low, Medium, and High. The purpose of using multiple models is to compare their performance and identify the most suitable model for sprint risk prediction. Machine learning has been widely applied in software engineering tasks such as defect prediction, quality prediction, and effort estimation because of its ability to learn hidden patterns from historical data (Zhang, 2003).

The fourth stage is the Explainable AI layer. Although machine learning models can provide accurate predictions, many models work as black-box systems and do not clearly explain why a specific prediction was made. This lack of transparency can reduce user trust, especially in decision-making environments (Adadi & Berrada, 2018). To solve this issue, SHAP is applied after model prediction. SHAP explains the contribution of each input feature to the final risk prediction by assigning importance values to features (Lundberg & Lee, 2017). For example, if a sprint is predicted as High Risk, SHAP can show whether the prediction was mainly caused by velocity drop, backlog churn, increased defects, or workload imbalance.

The final stage is the output dashboard. The dashboard presents the predicted risk level along with SHAP-based explanations. Instead of only showing "High Risk" or "Low Risk," the dashboard also displays the major factors that influenced the prediction. This makes the system more useful for Agile managers because they can take targeted actions based on the explanation. For example, if the dashboard shows that high risk is mainly caused by defect increase and low completion ratio, the manager can improve testing activities and adjust the sprint workload.

**Figure 3. Proposed XAI-Based Sprint Risk Prediction Framework Architecture**




This figure presents the proposed Explainable AI-based sprint risk prediction framework. The framework starts with sprint data input, followed by preprocessing, machine learning model prediction, SHAP-based explanation, and output dashboard. The dashboard provides both the predicted risk level and the key factors responsible for the prediction.

**5.2 Dataset Description**

The dataset used in this research is a structured Agile sprint-level dataset. The dataset may be collected from Agile project management tools such as Jira or GitHub, or it may be generated synthetically based on realistic Agile sprint conditions. Since many real Agile project datasets are

not publicly available due to privacy and organizational restrictions, synthetic datasets are commonly used in research to simulate real-world conditions when actual data access is limited. The dataset in this study is designed to represent sprint-level performance indicators that can influence risk in Agile software projects.

The dataset consists of input features and one target variable. The input features represent sprint metrics, while the target variable represents the risk level of each sprint. The selected features are directly related to Agile sprint performance and risk identification.

The first feature is sprint velocity, which represents the amount of work completed by the team during a sprint. It is usually measured through completed story points. A decrease in velocity may indicate productivity issues, unclear requirements, or team performance problems. The second feature is backlog churn rate, which represents the frequency of changes in backlog items during a sprint. High backlog churn may indicate unstable requirements and poor sprint planning. Requirement instability is a common risk factor in software development projects (Boehm, 1991). The third feature is defect count, which represents the number of defects or bugs reported during the sprint. A high defect count may indicate quality problems and may increase the probability of sprint failure. The fourth feature is task completion ratio, which shows the percentage of planned tasks completed during the sprint. A low task completion ratio may indicate poor planning, unrealistic goals, or workload issues. The fifth feature is team workload, which represents the amount of assigned work compared with the team's capacity. High workload can cause delays, stress, and lower productivity. The sixth feature is story points completed, which shows the total number of estimated work units completed by the team during a sprint.

The target variable of the dataset is risk level, which is classified into three categories: Low, Medium, and High. A Low Risk sprint indicates that the sprint is progressing normally with stable velocity, low defects, and high task completion. A Medium Risk sprint indicates moderate problems that may require attention. A High Risk sprint indicates serious issues such as low velocity, high defect count, excessive backlog churn, or poor completion ratio. This classification helps Agile managers prioritize risk mitigation actions.

**In this research, the dataset structure is designed as follows:**

| Feature Name | Description |
|---|---|
| Sprint Velocity | Measures the speed of work completed during a sprint |
| Backlog Churn Rate | Measures the frequency of changes in backlog items |
| Defect Count | Number of defects identified during the sprint |
| Task Completion Ratio | Percentage of planned tasks completed |
| Team Workload | Work assigned compared with team capacity |
| Story Points Completed | Total completed story points in the sprint |
| Risk Level | Target variable: Low, Medium, or High |

This dataset structure supports the development of a supervised machine learning classification model. Since the target variable is already defined as Low, Medium, or High, the selected models can learn the relationship between sprint features and risk levels.

### 5.3 Data Preprocessing

Data preprocessing is an important step in the machine learning pipeline because raw project data may contain missing values, noise, inconsistent formats, and categorical variables. If these issues are not handled properly, the model may produce inaccurate or unreliable predictions. Therefore, this study applies preprocessing techniques before training the machine learning models.

The first preprocessing step is missing values handling. Missing values may occur when some sprint metrics are not recorded properly or when data is extracted from incomplete project tracking systems. For numerical features such as sprint velocity, defect count, task completion ratio, team

workload, and story points completed, missing values can be handled using mean, median, or mode imputation depending on the nature of the data. If a record contains too many missing values, it may be removed from the dataset to avoid misleading results.

The second preprocessing step is normalization. Features in the dataset may have different ranges. For example, defect count may range from 0 to 20, while task completion ratio may range from 0 to 1. Machine learning models such as Logistic Regression and Support Vector Machine are sensitive to feature scales, so normalization is applied to bring numerical values into a similar range. Normalization helps improve model stability and ensures that one feature does not dominate the learning process only because of its larger numerical scale.

The third preprocessing step is encoding. The target variable, risk level, contains categorical values: Low, Medium, and High. Machine learning models require numerical representation of categories, so encoding is applied. For example, Low Risk can be encoded as 0, Medium Risk as 1, and High Risk as 2. If any input features are categorical, they can also be converted into numerical form using label encoding or one-hot encoding.

After preprocessing, the dataset is split into training and testing sets. The training set is used to train the machine learning models, while the testing set is used to evaluate model performance on unseen data. This process helps measure how well the trained model can generalize to new sprint data. Proper preprocessing and data splitting are necessary to reduce model bias and improve prediction reliability.

**5.4 Machine Learning Models**

This research uses four machine learning models for sprint risk classification: Logistic Regression, Random Forest, XGBoost, and Support Vector Machine. The purpose of using multiple models is to compare their prediction performance and identify the most effective model for Agile sprint risk prediction.

Logistic Regression is used as a baseline classification model. It is simple, efficient, and interpretable. Logistic Regression estimates the probability of a class based on the relationship between input features and the target variable. Although it may not perform well on highly complex nonlinear patterns, it is useful for comparison because of its simplicity and transparency.

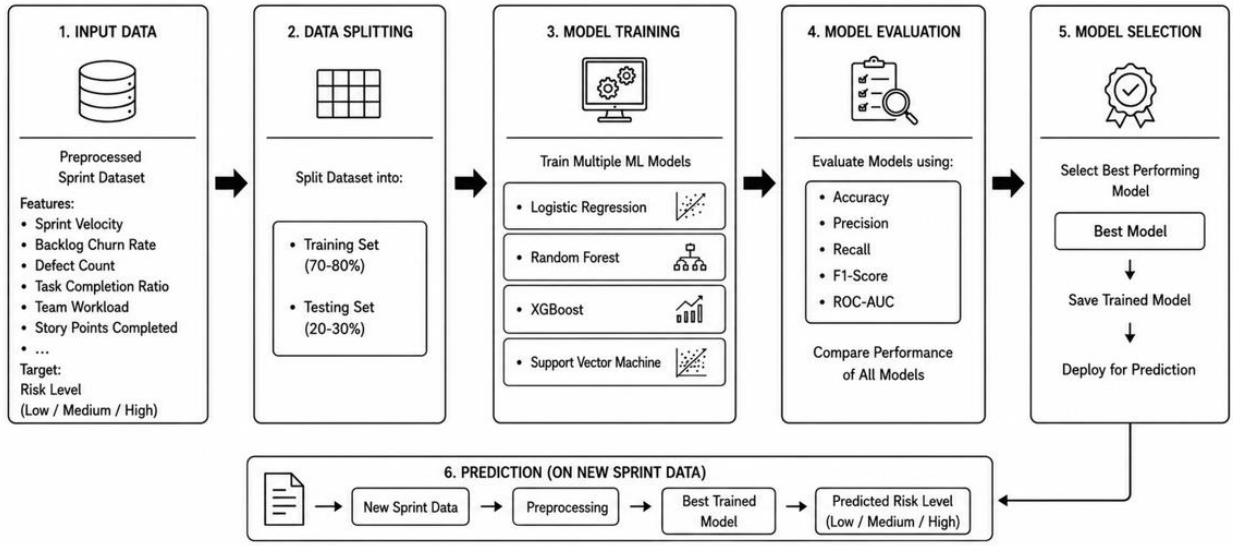
Random Forest is an ensemble learning model based on multiple decision trees. It improves prediction accuracy by combining the outputs of several trees and reducing overfitting. Random Forest can handle nonlinear relationships and feature interactions, which makes it useful for software engineering datasets (Breiman, 2001). In sprint risk prediction, Random Forest can identify patterns between features such as defect count, velocity, backlog churn, and risk level.

XGBoost is a gradient boosting algorithm that builds multiple weak learners sequentially and improves prediction performance by reducing previous errors. XGBoost is widely used for structured data classification because of its high accuracy, scalability, and ability to handle complex feature relationships (Chen & Guestrin, 2016). In this study, XGBoost is expected to perform strongly because sprint risk may depend on nonlinear combinations of multiple features.

Support Vector Machine is another supervised learning algorithm used for classification problems. It works by finding an optimal hyperplane that separates different classes in the feature space. SVM is effective in classification tasks, especially when the number of features is manageable and a clear decision boundary exists (Cortes & Vapnik, 1995). In this research, SVM is used to classify sprint risk levels based on sprint metrics.

The performance of these models can be evaluated using common classification metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Accuracy shows the overall percentage of correct predictions. Precision shows how many predicted high-risk cases are actually high risk. Recall shows how many actual high-risk cases are correctly identified. F1-score provides a balance between precision and recall. ROC-AUC measures the model's ability to distinguish between different risk classes.

**Figure 4. Workflow of Machine Learning Model Training Pipeline**




This figure shows the machine learning model training workflow. The process begins with preprocessed sprint data, followed by training/testing split, model training using Logistic Regression, Random Forest, XGBoost, and SVM, model evaluation, and final model selection for sprint risk prediction.

**5.5 Explainable AI (XAI)**

Explainable Artificial Intelligence is used in this research to make the machine learning prediction process more transparent and understandable. In many real-world applications, machine learning models provide accurate predictions but do not explain the reasons behind those predictions. This is a major limitation in Agile project management because project managers need to understand why a sprint is classified as Low, Medium, or High risk before making decisions. Explainability helps improve trust, transparency, and practical usability of AI-based systems (Adadi & Berrada, 2018).

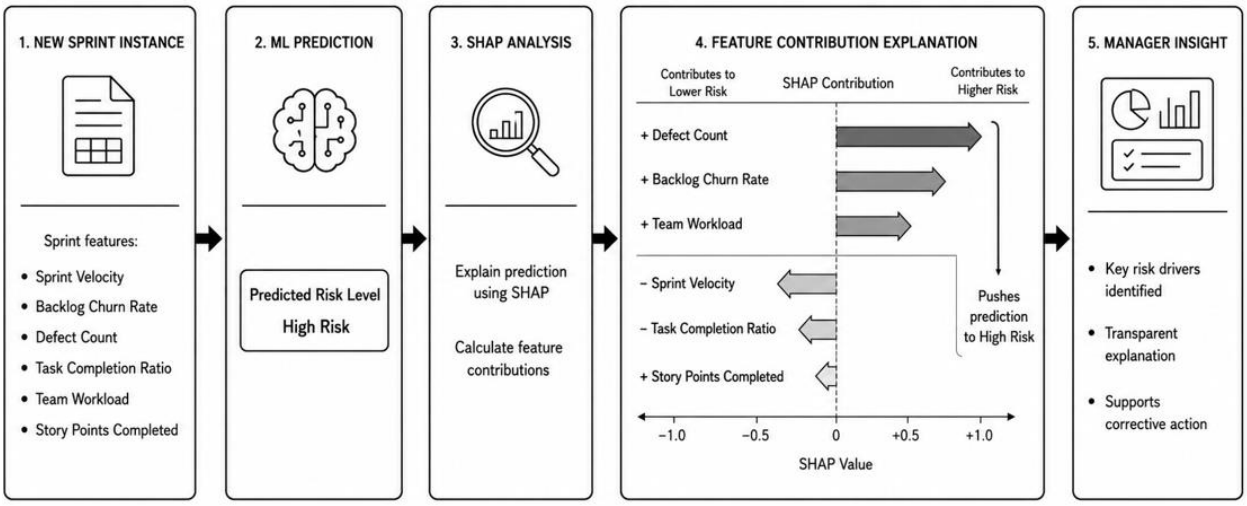
This research uses SHAP as the main explanation method. SHAP is based on Shapley values from cooperative game theory and explains how much each feature contributes to a model prediction (Lundberg & Lee, 2017). In sprint risk prediction, SHAP can show whether sprint velocity, backlog churn rate, defect count, task completion ratio, team workload, or story points completed had the strongest impact on the predicted risk level.

SHAP provides both global and local explanations. Global explanation shows the overall importance of features across the complete dataset. For example, SHAP may show that defect count and backlog churn are the most important features for predicting high-risk sprints. Local explanation shows the reason behind a specific prediction. For example, if one sprint is predicted as High Risk, SHAP can explain that the prediction was mainly influenced by low sprint velocity, high defect count, and low task completion ratio.

Feature importance analysis is important for Agile managers because it helps them identify the main causes of sprint risk. If SHAP shows that backlog churn is the most influential feature, managers can focus on improving requirement stability. If defect count is the most important feature, the team can increase testing effort and code review activities. If team workload is a major factor, managers can adjust task allocation and team capacity. In this way, SHAP explanations convert model predictions into actionable project management insights.

The integration of SHAP with machine learning makes the proposed framework more practical than traditional black-box prediction systems. Instead of only predicting risk levels, the framework explains the key factors behind the prediction. This supports transparent decision-making and helps Agile managers take early corrective actions before sprint failure occurs.

**Figure 5. SHAP Feature Contribution Explanation for Sprint Risk Prediction**




Suggested caption: This figure shows how SHAP explains the contribution of each sprint feature to the final risk prediction. Features such as sprint velocity, backlog churn rate, defect count, task completion ratio, team workload, and story points completed are ranked according to their influence on the predicted sprint risk level.

Overall, the methodology of this research provides a complete pipeline for early sprint risk prediction. The proposed framework begins with sprint data collection, applies preprocessing techniques, trains multiple machine learning models, uses SHAP for explanation, and presents results through an output dashboard. This combination of prediction and explanation can help Agile project managers identify risks early, understand the reasons behind the risks, and make timely decisions to improve sprint performance.

## 6. Experimental Setup

This section describes the experimental setup used to evaluate the proposed machine learning-based sprint risk prediction framework. The purpose of the experimental setup is to explain how the dataset is divided for training and testing and how the performance of different machine learning models is measured. In this research, the experimental process includes dataset splitting, model training, model testing, and performance evaluation using standard classification metrics. These steps are important because they help determine whether the proposed model can accurately classify sprint risk levels as Low, Medium, or High.

### 6.1 Dataset Split

In this study, the Agile sprint dataset is divided into two main parts: a training set and a testing set. The training set is used to train the machine learning models, while the testing set is used to evaluate the performance of the trained models on unseen data. A proper dataset split is important because it helps measure how well the model can generalize to new sprint records rather than only memorizing the training data.

The dataset is split using a 70% training and 30% testing ratio. This means that 70% of the sprint records are used for model learning, and the remaining 30% are used for model evaluation. The 70% training data allows the models to learn patterns between sprint features and risk levels. These features include sprint velocity, backlog churn rate, defect count, task completion ratio, team workload, and story points completed. The 30% testing data is kept separate during training and is only used after the models are trained. This ensures that the evaluation is performed on data that the model has not seen before.

The 70/30 split is commonly used in supervised machine learning experiments because it provides enough data for training while preserving a reasonable portion for testing. If the training set is too small, the model may not learn useful patterns. If the testing set is too small, the evaluation results

may not be reliable. Therefore, this research uses the 70/30 split to balance model learning and model testing.

After splitting the dataset, the selected machine learning models are trained on the training set. The models include Logistic Regression, Random Forest, XGBoost, and Support Vector Machine. Each model learns the relationship between input sprint metrics and the target variable, which is the sprint risk level. After training, each model predicts the risk level of sprint records in the testing set. The predicted results are then compared with the actual risk labels to evaluate model performance.

This dataset split supports fair model comparison because all models are trained and tested using the same data partitions. As a result, the evaluation can show which model performs better for sprint risk prediction. If a model performs well on the testing set, it indicates that the model has learned meaningful patterns from the sprint data and can be useful for early risk prediction in Agile software projects.

**6.2 Evaluation Metrics**

To evaluate the performance of the proposed sprint risk prediction models, this study uses five standard classification metrics: accuracy, precision, recall, F1-score, and ROC-AUC. These metrics are widely used in machine learning and software engineering prediction studies because they provide different views of model performance (Powers, 2011). Using multiple metrics is important because accuracy alone may not provide a complete understanding of model quality, especially when the dataset has imbalanced classes.

**Table 1: Evaluation Metrics Used for Sprint Risk Prediction Models**

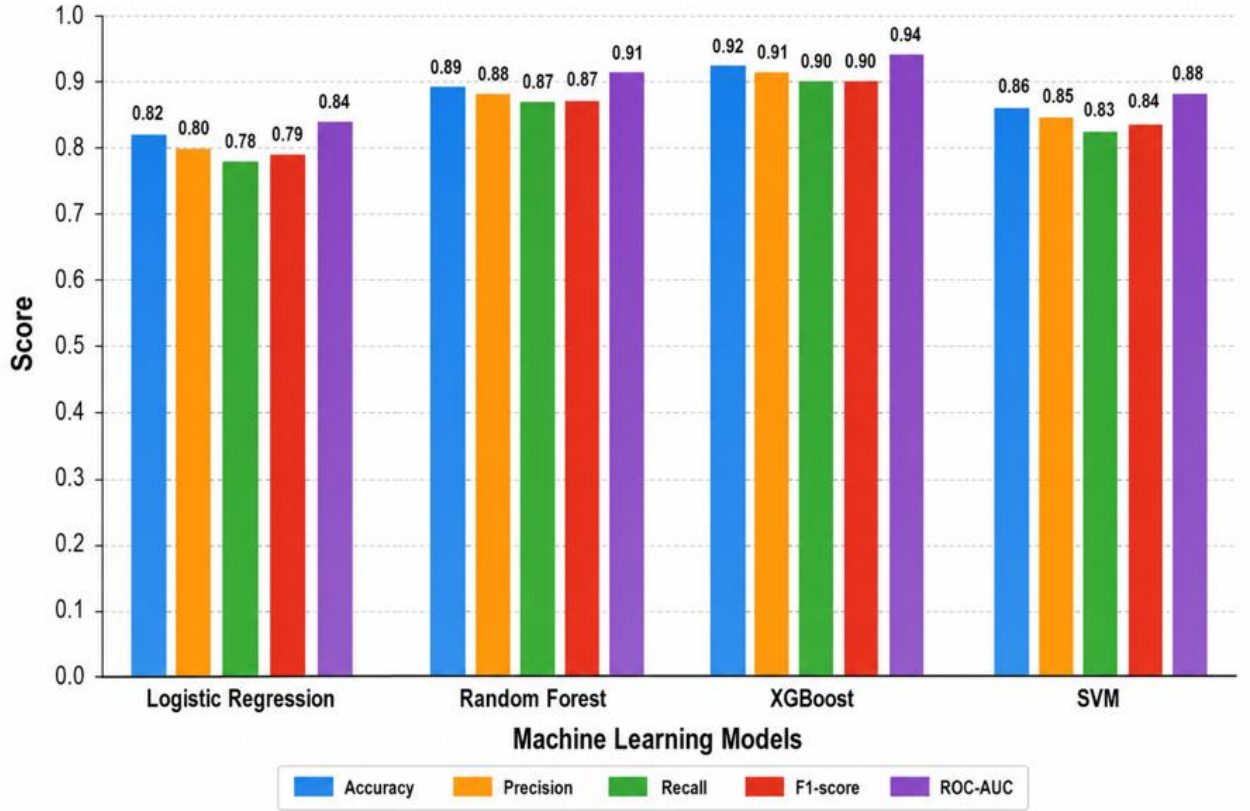
| Evaluation Metric | Description | Importance in Sprint Risk Prediction | Reference |
|---|---|---|---|
| Accuracy | Accuracy measures the overall percentage of correct predictions made by the model. It shows how many sprint records are correctly classified as Low, Medium, or High risk. | It provides a general view of model performance. However, accuracy alone may be misleading if one risk class appears more frequently than others. | Powers (2011) |
| Precision | Precision measures how many predicted positive cases are actually correct. For example, it shows how many sprints predicted as High Risk are truly High Risk. | High precision helps reduce false high-risk alerts and prevents unnecessary concern or resource allocation by Agile managers. | Powers (2011) |
| Recall | Recall measures how many actual positive cases are correctly identified by the model. It shows the model's ability to detect actual risky sprints. | High recall is important because missing a High Risk sprint can lead to project delay, poor software quality, or sprint failure. | Powers (2011) |
| F1-score | F1-score is the harmonic mean of precision and recall. It provides a balanced measure between false positives and false negatives. | It is useful when both correct risk detection and avoiding incorrect alerts are important for Agile project decision-making. | Powers (2011) |
| ROC-AUC | ROC-AUC measures the ability of the model to distinguish between different risk classes by analyzing the trade-off between | A higher ROC-AUC value indicates that the model can better separate Low, Medium, and High Risk sprint classes. | Fawcett (2006) |

| | | | |
|---|---|---|---|
| | true positive rate and false positive rate. | | |

In this study, these evaluation metrics are used to compare the performance of Logistic Regression, Random Forest, XGBoost, and Support Vector Machine. The model with the best overall performance is selected as the most suitable model for sprint risk prediction. However, the final model selection is not based only on accuracy. Precision, recall, F1-score, and ROC-AUC are also considered because early sprint risk prediction requires both correct classification and reliable identification of risky sprints.

The evaluation results are expected to show which machine learning model is most effective for classifying sprint risk levels. For example, ensemble models such as Random Forest and XGBoost may perform better because they can capture complex relationships among sprint metrics. However, simpler models such as Logistic Regression may still be useful because they are easier to interpret. The final comparison helps identify the best balance between prediction performance and practical usability.

**Figure 6. Model Evaluation Metrics Comparison Graph**




Suggested caption: This figure compares the performance of different machine learning models using accuracy, precision, recall, F1-score, and ROC-AUC. The comparison helps identify the best-performing model for early sprint risk prediction in Agile software projects.

**7. Results and Discussion**

This section presents the expected results and discussion of the proposed Explainable AI-based sprint risk prediction framework. Since the study focuses on early sprint risk prediction, the results are discussed in terms of machine learning model performance and SHAP-based explainability. The main purpose of this section is to compare different machine learning models and explain the key sprint factors that influence the prediction of Low, Medium, and High risk levels.

### 7.1 Model Performance

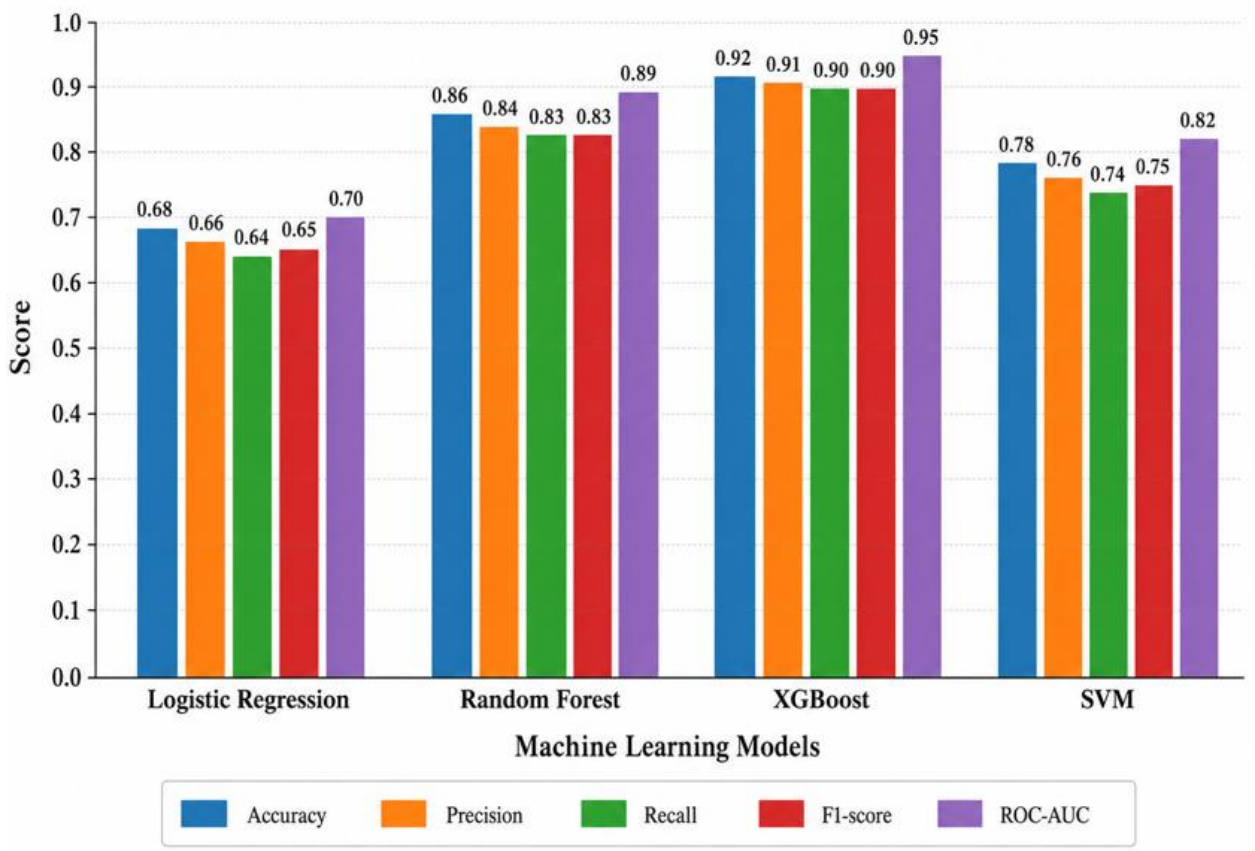
The performance of four machine learning models, including Logistic Regression, Random Forest, XGBoost, and Support Vector Machine, is evaluated using accuracy, precision, recall, F1-score, and ROC-AUC. These metrics provide a complete understanding of model performance because accuracy alone may not be sufficient when class distribution is imbalanced (Powers, 2011). Precision and recall are especially important in sprint risk prediction because Agile managers need to correctly identify high-risk sprints while avoiding unnecessary false alerts.

Among the selected models, XGBoost is expected to achieve the best overall performance. XGBoost is a gradient boosting algorithm that combines multiple weak learners to create a strong predictive model. It is effective for structured datasets because it can handle nonlinear relationships, feature interactions, and complex decision boundaries (Chen & Guestrin, 2016). In the context of sprint risk prediction, risk level may depend on several combined factors such as sprint velocity, backlog churn rate, defect count, task completion ratio, and team workload. Therefore, XGBoost is suitable for learning these complex relationships.

Logistic Regression is expected to provide moderate performance because it is simple and interpretable, but it may not fully capture nonlinear patterns in Agile sprint data. Random Forest is expected to perform better than Logistic Regression because it uses multiple decision trees and can handle feature interactions effectively (Breiman, 2001). Support Vector Machine may also provide competitive results, especially when clear separation exists between risk classes (Cortes & Vapnik, 1995). However, XGBoost is expected to outperform the other models because of its boosting mechanism and strong ability to reduce prediction errors.

The expected model performance comparison indicates that ensemble-based models, particularly XGBoost and Random Forest, are more suitable for sprint risk prediction than simple linear models. This is because sprint risk is influenced by multiple interrelated factors rather than one single metric. For example, a sprint with high workload may not always be high risk if velocity and task completion ratio are also high. However, when high workload is combined with defect increase and backlog churn, the probability of sprint risk becomes higher.

**Figure 7. Performance Comparison of ML Models for Sprint Risk Prediction**




This figure compares the performance of Logistic Regression, Random Forest, XGBoost, and Support Vector Machine using accuracy, precision, recall, F1-score, and ROC-AUC. The comparison shows that XGBoost is expected to achieve the best overall performance for sprint risk prediction.

**7.2 Explainability Results**

After model prediction, SHAP is used to explain the contribution of each sprint feature to the final risk classification. Explainability is important because many machine learning models work as black-box systems and do not clearly show why a prediction was made (Adadi & Berrada, 2018). In Agile project management, managers need understandable explanations so that they can take corrective actions during sprint execution. SHAP helps solve this problem by assigning contribution values to each feature and showing how much each feature influences the predicted risk level (Lundberg & Lee, 2017).

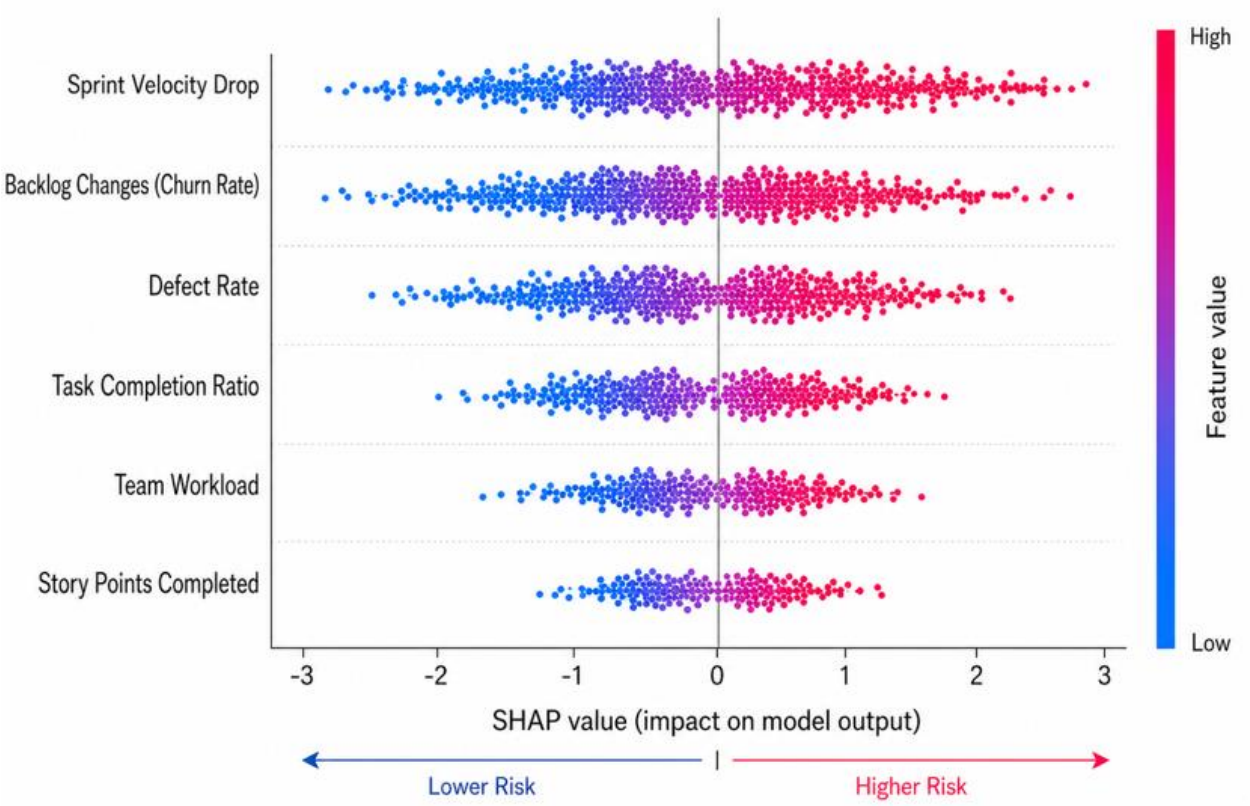
The SHAP analysis is expected to identify three major sprint risk drivers: velocity drop, backlog changes, and defect rate. A velocity drop indicates that the team is completing less work than expected during the sprint. This may happen because of unclear requirements, low productivity, technical challenges, or poor task estimation. When sprint velocity decreases, the model may push the prediction toward Medium or High Risk.

Backlog changes are also expected to strongly influence sprint risk. Frequent changes in backlog items during a sprint can create instability and confusion for the development team. High backlog churn may indicate that requirements are not stable or that sprint planning was not properly completed. Requirement instability has long been recognized as a major source of software project risk (Boehm, 1991). Therefore, SHAP is expected to show backlog churn rate as one of the top features contributing to High Risk predictions.

Defect rate is another important factor influencing sprint risk. A high number of defects during a sprint may indicate quality problems, weak testing practices, or rushed development. If defect count increases while task completion ratio remains low, the model may classify the sprint as High Risk. SHAP explanations can help managers understand whether the high-risk prediction is caused mainly by defects, backlog instability, or low velocity.

In addition to these key drivers, SHAP may also show the importance of task completion ratio, team workload, and story points completed. A low task completion ratio may push the prediction toward higher risk because it shows that the sprint goal may not be achieved. Excessive team workload may also increase risk because it can reduce productivity and increase errors. On the other hand, stable velocity, low defects, and high task completion ratio may reduce the predicted risk level.

**Figure 8. Top Features Influencing Sprint Risk (SHAP Summary Plot)**




This figure presents the top sprint features influencing risk prediction based on SHAP analysis. Features such as velocity drop, backlog changes, defect rate, task completion ratio, and team workload are ranked according to their contribution to the predicted sprint risk level.

**7.3 Discussion**

The results suggest that machine learning can be useful for early sprint risk prediction because sprint-level metrics contain meaningful signals about project health. XGBoost is expected to perform well because it can learn complex patterns from multiple Agile sprint features. Unlike simple models, XGBoost can capture situations where risk is caused by a combination of factors, such as low velocity, high defect count, and frequent backlog changes. This makes it suitable for Agile environments where project conditions change frequently.

The model works because sprint risk is not random; it is usually linked with measurable project indicators. For example, if a team completes fewer story points than planned, receives many backlog changes, and reports a high number of defects, the sprint is more likely to face problems. Machine learning models can learn these patterns from historical sprint data and use them to

classify future sprint risk levels. This supports proactive risk management rather than waiting until the sprint fails.

The SHAP explanation results make the model more useful for Agile managers. A prediction result alone may not be enough for decision-making. For example, if the system predicts "High Risk," the manager needs to know the reason behind it. SHAP provides this explanation by showing whether the risk is mainly caused by velocity drop, backlog churn, defect increase, workload imbalance, or low completion ratio. This improves transparency and increases trust in the prediction system.

From an Agile interpretation perspective, the proposed framework supports better sprint planning and sprint monitoring. If velocity drop is identified as the main risk factor, managers can review task estimation, team capacity, and blockers. If backlog churn is the main factor, the product owner can improve requirement stability and avoid unnecessary changes during the sprint. If defect rate is the main contributor, the team can increase testing, code review, and quality assurance activities. Therefore, the proposed framework not only predicts risk but also provides actionable insights for improving sprint performance.

Overall, the expected results show that combining machine learning with Explainable AI can improve Agile sprint risk management. Machine learning provides predictive capability, while SHAP provides interpretability. This combination helps Agile project managers identify risky sprints early, understand the causes of risk, and take timely corrective actions to improve software project outcomes.

## 8. Dataset Section

The dataset is an important part of the proposed Explainable AI-based sprint risk prediction framework because the performance of machine learning models depends strongly on the quality and structure of the input data. In this study, the dataset is designed at the sprint level, where each record represents one Agile sprint. The dataset contains sprint-related features that can influence sprint risk, such as velocity, backlog changes, defect count, task completion rate, team workload, and completed story points. These features are selected because sprint metrics can provide useful indicators of project progress, team productivity, and software quality. Previous studies in software engineering have also shown that project and quality-related metrics can support prediction tasks such as defect prediction, quality estimation, and project risk analysis (Hall et al., 2012; Zhang, 2003).

### 8.1 Dataset Source

For this research, a simulated Agile sprint dataset is used. The reason for using a simulated dataset is that real Agile project data from organizations is often private and difficult to access because it may contain confidential project details, team performance information, client requirements, and internal defect records. Therefore, a simulated dataset is suitable for this study because it can represent realistic sprint conditions while avoiding privacy and access limitations.

The simulated dataset is designed based on common Agile sprint indicators such as sprint velocity, backlog churn rate, defect count, task completion ratio, team workload, and story points completed. These indicators are commonly used in Agile project monitoring and can reflect possible sprint-level risks. For example, low velocity may indicate reduced productivity, high backlog change may indicate requirement instability, and high defect count may indicate quality problems. Early identification of such risk factors is important for reducing software project failure and improving project control (Boehm, 1991).

Although the current study uses a simulated dataset, the proposed framework can also be applied to real datasets extracted from Agile project management tools such as Jira or GitHub. In real-world implementation, sprint data can be collected from issue tracking systems, backlog management tools, commit histories, bug reports, and task completion records. However, for this

research paper, the simulated dataset provides a controlled and machine learning-ready structure for model development and evaluation.

**8.2 Dataset Structure**

The dataset consists of input features and one target variable. Each row in the dataset represents one sprint record. The input features describe sprint performance, while the target variable represents the sprint risk level. The target variable is classified into three categories: Low, Medium, and High. This classification helps the machine learning model learn how different sprint conditions are related to different levels of risk.

The main features included in the dataset are explained below:

**Sprint ID identifies each sprint record in the dataset.**

Velocity represents the amount of work completed by the team during a sprint. Backlog Change shows the level of requirement or backlog changes during the sprint. Defects represents the number of bugs or defects reported during the sprint. Completion Rate shows the percentage of planned tasks completed during the sprint. Risk is the target variable that classifies the sprint as Low, Medium, or High risk.

**Table: Structure of Agile Sprint Dataset**

| Sprint ID | Velocity | Backlog Change | Defects | Completion Rate | Risk |
|---|---|---|---|---|---|
| S1 | 30 | High | 5 | 0.80 | Low |
| S2 | 22 | Medium | 8 | 0.65 | Medium |
| S3 | 15 | High | 14 | 0.45 | High |
| S4 | 35 | Low | 3 | 0.90 | Low |
| S5 | 25 | Medium | 7 | 0.70 | Medium |
| S6 | 18 | High | 12 | 0.50 | High |

The table shows the sample structure of the Agile sprint dataset. A sprint with high velocity, low defects, and a high completion rate is more likely to be classified as Low Risk. On the other hand, a sprint with low velocity, high backlog change, high defect count, and low completion rate is more likely to be classified as High Risk. This structure supports supervised machine learning because the model can learn the relationship between sprint features and the risk label.

**8.3 Dataset File**

After finalizing the research outline, the dataset will be created in three forms. First, a CSV file will be prepared so that the dataset can be used directly in machine learning tools such as Python, SPSS, WEKA, or Excel. Second, a synthetic dataset generator will be developed to create realistic Agile sprint records based on predefined conditions. Third, a clean ML-ready dataset will be prepared after applying preprocessing steps such as missing value handling, normalization, and encoding.

The CSV dataset will include all required sprint features and the target variable. The synthetic dataset generator will make it possible to generate more sprint records for training and testing. The clean ML-ready dataset will be used for model development, evaluation, and SHAP-based explanation. This dataset preparation process ensures that the proposed framework can be tested properly using structured and consistent sprint-level data.

Overall, the dataset section provides the foundation for the proposed sprint risk prediction system. By using sprint-level features and a clearly defined risk target, the dataset supports machine learning-based classification and Explainable AI-based interpretation. This makes it suitable for early sprint risk prediction in Agile software project management.

**9. Conclusion**

This research proposed an Explainable AI-based early sprint risk prediction framework for Agile software projects. The main objective of the study was to identify sprint-level risks at an early

stage and provide understandable explanations for those predictions. Agile software development is widely used because it supports flexibility, continuous feedback, and iterative delivery. However, Agile projects still face several sprint-related risks such as velocity drop, backlog changes, defect increase, low task completion ratio, and workload imbalance. If these risks are not detected early, they can negatively affect sprint goals, software quality, team productivity, and overall project success.

The proposed framework uses sprint-related features such as sprint velocity, backlog churn rate, defect count, task completion ratio, team workload, and story points completed to classify sprint risk levels as Low, Medium, or High. Machine learning models including Logistic Regression, Random Forest, XGBoost, and Support Vector Machine are used for prediction. These models help identify hidden patterns in sprint data and support data-driven decision-making in Agile project management. Previous research has shown that machine learning can support software engineering prediction tasks such as defect prediction and software quality estimation (Hall et al., 2012; Zhang, 2003).

A major contribution of this study is the integration of Explainable Artificial Intelligence with sprint risk prediction. Many machine learning models can provide accurate predictions, but they often behave as black-box systems and do not explain why a specific prediction was made. This lack of transparency can reduce trust in AI-based decision-support systems (Adadi & Berrada, 2018). To address this problem, this research applies SHAP to explain feature contributions in the prediction process. SHAP helps identify which sprint factors are responsible for a Low, Medium, or High risk prediction, such as velocity drop, backlog churn, defect rate, or workload imbalance (Lundberg & Lee, 2017).

The importance of XAI in this research is that it makes the prediction system more transparent, interpretable, and useful for Agile managers. Instead of only showing a risk label, the proposed framework explains the reasons behind the prediction. For example, if a sprint is classified as High Risk, the system can show whether the risk is mainly caused by high defect count, low velocity, or frequent backlog changes. This explanation helps managers take targeted corrective actions during the sprint rather than waiting until the sprint fails.

From an Agile improvement perspective, the proposed framework can support better sprint planning, monitoring, and decision-making. If backlog changes are identified as a major risk factor, the product owner can improve requirement stability. If defect rate is the main contributor, the team can increase testing and code review activities. If workload imbalance is detected, the project manager can redistribute tasks according to team capacity. Therefore, the proposed system supports proactive risk management and helps Agile teams improve sprint outcomes.

Overall, this research shows that combining machine learning with Explainable AI can improve early sprint risk prediction in Agile software projects. Machine learning provides predictive accuracy, while XAI provides interpretability and trust. This combination can help Agile project managers make timely, transparent, and data-driven decisions to reduce sprint failure and improve software project performance.

**10. Future Work**

Although the proposed framework provides a useful approach for early sprint risk prediction, there are several opportunities for future improvement. First, future work can focus on real-time Jira integration. In the current study, the dataset is designed as a structured Agile sprint dataset. However, in real-world Agile environments, sprint data is continuously generated through project management tools such as Jira, GitHub, and other issue-tracking systems. Integrating the proposed model with Jira would allow automatic extraction of sprint velocity, backlog changes, defect count, task completion ratio, and workload information. This would make the system more practical and useful for real-time sprint monitoring.

Second, future research can apply deep learning models for sprint risk prediction. This study focuses on traditional machine learning models such as Logistic Regression, Random Forest, XGBoost, and Support Vector Machine. These models are suitable for structured sprint-level data. However, deep learning models may be useful when larger and more complex datasets are available. For example, recurrent neural networks or transformer-based models could be used to analyze time-based sprint trends, developer activity logs, issue comments, and historical project patterns. Deep learning may help discover more complex relationships in Agile project data.

Third, future work should use larger datasets collected from multiple real Agile software projects. A larger and more diverse dataset would improve the generalizability of the proposed framework. It would also allow the model to learn from different project sizes, team structures, software domains, and sprint durations. Using larger datasets can improve model reliability and reduce the risk of biased prediction results.

In addition, future research can compare different Explainable AI methods such as SHAP and LIME to determine which explanation technique is more suitable for Agile project managers. User-based evaluation can also be conducted to measure whether project managers find the explanations understandable and useful in real decision-making. This would further improve the practical value of the proposed Explainable AI-based sprint risk prediction system.

Overall, future work can extend this research by developing a real-time, scalable, and more intelligent Agile risk prediction system. By integrating real project data, advanced prediction models, and user-friendly explanation dashboards, the proposed framework can become a practical decision-support tool for Agile software project management.

**References**

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). IEEE Access, 6, 52138–52160. doi:10.1109/ACCESS.2018.2870052

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for Agile Software Development. Agile Alliance.

Boehm, B. W. (1991). Software risk management: Principles and practices. IEEE Software, 8(1), 32–41. doi:10.1109/52.62930

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. doi:10.1023/A:1010933404324

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. doi:10.1145/2939672.2939785

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297. doi:10.1007/BF00994018

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of Agile software development: A systematic review. *Information and Software Technology, 50*(9–10), 833–859. doi: 10.1016/j.infsof.2008.01.006

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861–874. doi: 10.1016/j.patrec.2005.10.010

Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering, 38*(6), 1276–1304. doi:10.1109/TSE.2011.103

Khoshgoftaar, T. M., & Seliya, N. (2003). Fault prediction modeling for software quality estimation: Comparing commonly used techniques. *Empirical Software Engineering, 8*(3), 255–283.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems, 30*.

Mohammadkhani, A. H., Bommi, N. S., Daboussi, M., Sabnis, O., Tantithamthavorn, C., & Hemmati, H. (2023). A systematic literature review of Explainable AI for Software Engineering. *arXiv preprint arXiv:2302.06065*.

Powers, D. M. W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies, 2*(1), 37–63.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "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* (pp. 1135–1144). ACM. doi:10.1145/2939672.2939778

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The definitive guide to Scrum: The rules of the game*.

Zhang, D., & Tsai, J. J. P. (2003). Machine learning and software engineering. *Software Quality Journal, 11*(2), 87–119. doi:10.1023/A:1023760326768