Selection of Software Development Life Cycle Models using Machine Learning Approach

Dires Bitew Aniley
School of Computing Department of Information Technology
Woldia University
Woldia, Ethiopia

Esubalew Alemneh Jalew
ICT4D Research Center, Bahir Dar Institute of Technology
Bahir Dar University
Bahir Dar, Ethiopia

Getasew Abeba Agegnehu
School of Computing Department of Information Technology
Woldia University
Woldia, Ethiopia

ABSTRACT
Inappropriate selection of Software Development Life Cycle (SDLC) models can lead to increased development time and costs, heightened overhead, elevated risk exposure, difficulty managing uncertainty, reduced quality, strained client relations, and insufficient project tracking and control. These challenges are exacerbated by the diverse expertise levels of software developers, ranging from novices to seasoned professionals. While experts may conduct in-depth analyses to select suitable SDLC models, beginners often lack the criteria for effective model selection. Previous research has shown limitations in the criteria used for SDLC model selection, usually relying on knowledge-based systems that lack flexibility and scalability. To address this problem, they propose an automatic SDLC model selection system using a machine-learning approach tailored to specific project requirements. They conducted comparative experimental analyses using machine learning and deep learning algorithms such as KNN, CNN, NB, ANN, Random Forest, and Decision Trees. Experimental results demonstrated that Decision Tree and Random Forest achieved 99.9% accuracy in the classification task, indicating their effectiveness in automating SDLC model selection for software projects.

Keywords
Software Development Life Cycle, Selection of Software Development Life Cycle, Project Characteristics, Machine Learning, Deep Learning

1 INTRODUCTION
In the current dynamic and competitive environment of software development, the selection of a Software Development Life Cycle (SDLC) model plays a crucial role in determining project success. SDLC models offer structured frameworks that steer the planning, execution, and completion of software projects, affecting elements such as development strategies, resource distribution, and project schedules. Nevertheless, given the abundance of SDLC models, which span from conventional waterfall to contemporary agile methodologies, choosing the most appropriate model for a specific software project can be challenging [1].

According to [2] and [3], SDLC model selection is significant in ensuring project alignment with organizational goals and stakeholder expectations. These studies highlight the diverse nature of software projects and the need for adaptable and flexible SDLC selection approaches to accommodate varying project requirements.

The necessity of SDLC model selection arises from the diverse nature of software projects, each characterized by unique requirements, constraints, and stakeholder expectations. While some projects may benefit from the structured and sequential approach of the waterfall model, others may thrive in the iterative and adaptive environment fostered by agile methodologies. The ability to identify the optimal SDLC model tailored to the specific needs of each project is therefore paramount for maximizing project efficiency, productivity, and ultimately, success.

Previous research efforts have explored various approaches to SDLC model selection, including qualitative assessments, expert opinions, and decision support systems. However, these approaches often rely on subjective judgments, lack of scalability, and may overlook valuable insights hidden within project data. Moreover, traditional methods may struggle to adapt to the evolving nature of software development practices and the increasing complexity of modern software projects.

Studies [4] and [5] highlighted the limitations of traditional SDLC model selection approaches and emphasize the need for data-driven methodologies to enhance decision-making processes in software development.

In response to these challenges, the integration of machine learning (ML) techniques offers a promising avenue for enhancing SDLC model selection processes [6]. ML algorithms have the ability to analyze large volumes of historical project data, identify patterns, and extract actionable insights to inform decision-making. By leveraging features such as project size, complexity, team composition, and project objectives, ML models can learn from past project experiences and select appropriate SDLC model for new projects.

The proposed model in this research aims to address the limitations of existing SDLC model selection approaches by leveraging ML algorithms to provide data-driven and objective recommendations. By training models on comprehensive datasets containing project attributes and corresponding SDLC model choices, they seek to develop predictive models capable of accurately predicting the optimal SDLC model for new projects.

This problem statement encapsulates the need to move beyond traditional, subjective approaches to SDLC model selection and develop more objective, scalable, and adaptive methodologies that influence ML to enhance decision-making processes in software development. They addressed this problem statement, and contributed a labeled dataset and new model to the advancement of SDLC model selection methodologies.

Finally, to achieve the identified specific objective, this thesis answered the following research questions:

- ✓ What are the key project characteristics and stakeholder requirements that influence the selection of an appropriate SDLC model?
- ✓ In what ways can historical project data be leveraged to identify patterns and trends that inform SDLC model selection decisions?
- ✓ To what extent can machine learning models accurately predict the optimal SDLC model for new projects, and what factors contribute to their predictive performance?

The remainder of this paper is organized as follows: Section II provides a detailed review of the related work, establishing the context and significance of their research. Section III describes the methodology and the theoretical framework underpinning their study, including the models, algorithms, or experimental setup utilized. In Section IV, they present the results of their experiments or analysis, supplemented with figures, and comprehensive discussions. Section V concludes the paper by summarizing the findings from their experiments. Section VI summarizes the key contributions of the study, discusses the implications of the findings, and suggests areas for future research.

2 RELATED WORKS

Kumar, K. (2013) addresses a crucial aspect of software engineering: the selection of an appropriate SDLC model. The authors make a significant contribution to the field of software engineering by providing a structured and objective approach to SDLC selection. Given the multitude of SDLC models available such as waterfall, incremental, iterative, RAD, and prototype models. Choosing the most suitable one for a specific project can significantly impact the project's success [7]. The authors propose a rule-based recommendation system to guide this selection process, aiming to streamline decision-making and enhance project outcomes.

However, a predefined set of rules that may not account for the dynamic and evolving nature of software development practices. This static approach can limit the system's adaptability to new project requirements and emerging SDLC models

Ozturk, V. (2016) this research paper delves into the critical decision-making process for selecting an appropriate SDLC model. The authors propose an innovative approach using fuzzy logic to handle the inherent uncertainties and subjective criteria involved in the selection [8]. Traditional methods often fall short in accounting for the nuanced preferences and varying project requirements. The fuzzy logic-based model excels by evaluating complex factors like project size, risk, and team expertise with greater flexibility and precision. However, the paper's validation through a narrow set of case studies restricts its demonstration of applicability across diverse industries and project types. Broader empirical evidence would enhance the robustness and generalizability of the proposed model

Overall, the integration of fuzzy logic into SDLC selection presents a significant advancement, offering a robust tool for project managers navigating the multifaceted landscape of software development. This approach promises to refine how teams choose SDLC models, potentially leading to more successful project executions.

The study by Adanna, A. A., & Nonyelum, O. F. (2020) provides a comprehensive analysis of the essential criteria for selecting the most suitable SDLC models, crucial for the success of software projects [9]. The authors outline a structured framework that highlights key factors such as project complexity, team size, client requirements, and risk management. Their methodical approach aids in matching those criteria with the characteristics of various SDLC models like Agile, Waterfall, Spiral, iterative, incremental, prototype, and RAD.

One of the paper's notable strengths is its practical guidance on evaluating these criteria through detailed tables and comparative analyses. This clarity helps project managers make informed decisions by aligning project-specific needs with the optimal SDLC approach. The inclusion of real-world examples enhances the relevance and applicability of the proposed criteria.

Dhami et al., 2021 applied advanced deep learning techniques to the domain of SDLC models prediction, an innovative approach that highlighted the potential of DL in automating complex decision-making processes within software engineering [10].

The study presented comprehensive experimental results, evaluating the performance of different ML and DL techniques using multiple metrics. These models are trained and evaluated using a substantial dataset comprising 6000 instances across different project types and complexities. Evaluation metrics such as accuracy, precision, recall, and F1 score are used to assess the models' performance in predicting four main SDLC models.

However, the number of SDLC models they used (waterfall, evolutionary, incremental, and hybrid) and level of datasets for their experiment may not represent the diversity in software development projects across different industries and scales. This limitation could affect the generalizability of the model predictions.

3 METHODLOGIES

In this research they use experimental methodologies which are broadly used in computing science to evaluate new solutions for problems. The evaluation has two phases. In an exploratory phase the researcher took measurements that will help identify what are the questions that should be asked about the system under evaluation [11]. Then an evaluation phase attempted to answer these questions. A well-designed experiment started with a list of the questions that the experiment was expected to answer.

3.1 Dataset Preparation Metnodology

Preparing a dataset for use in software engineering research involves several key steps to ensure that the data is relevant, accurate, and suitable for the research objectives. Here is a methodology for dataset preparation in this research:

- Review literature relevant for Software development model selection.
- Identify criteria and model relationship from available literature.
- Identify significant criteria and models that the research mainly concerns.
- Identify the possible domain value of each criterion.
- Identify rules for model selection from collected literature facts.
- Prepare a dataset based on rules.

This research focused on ten (10) main criteria namely product delivery, clear requirement specification. Requirement change, user involvement, project size, project risk, project complexity, team experience, project budget, and reusability of components were

selected as a criterion for system development model selection. System development models for this research focused on are selected based the following criteria.

- formal software development model.
- The model must be in majority of literatures in system development model researched works.
- Recent Project developers used the model.
- The model must have its own unique criteria.

Based on the above criteria eight software development models namely waterfall, iterative, incremental, prototype, spiral, RAD, v-model, and agile were selected in this research.

They identified relevant criteria, models, potential values for each criterion, and the corresponding model value. For nine criteria, the model values were categorized as neutral, low, medium, or high, while one criterion had possible values of high, low, and neutral. The 'neutral' label was applied when the literature did not provide a value for an attribute, indicating no influence on model selection. The numerical representation for these values was assigned as follows: low (-1), medium (1), high (2), and neutral (0).

Using this numeric system, they generated a matrix based on the 10 criteria and their possible values. The matrix with four potential values per criterion results in 1,048,576 possible combinations. However, one criterion (Reusability of components) has only three possible values. As a result, the total number of valid combinations to construct their dataset is 786,432 (410-49). With this generated matrix, they labeled the dataset following the specific rules. The sample generated rules for dataset labeling shown in Section 1 below is represented the features of the rule product delivery (PD), clear requirement specification (CRS). Requirement change (RC), user involvement (UI), project size (PS), project risk management (RM), project complexity (CP), team experience (TE), project budget (PB), and reusability of components (RUC) order respectively. .

```
{
('0','2','-1','0','0','-1','0','0','0','0'):'Waterfall',
('0','0','-1','2','-1','-1','-1','0','0','0'):'Iterative',
('0','0','2','2','-1','-1','2','0','0','0'):'Agile',
('0','-1','2','2','-1','1','2','0','2','0'):'Prototype,
('0','0','0','2','1','1','-1','0','0','0'):'Incremental',
('0','0','2','0','2','-1','0','0','0','0'):'Spiral',
('0','0','0','0','2','-1','0','0','0','0'):'RAD',
('0','0','2','0','0','2','0','0','0','0'):'Spiral',
}
```

Section: 1 Sample rules used for dataset labeling

3.2 Model Development Methodology

Selecting an appropriate SDLC model using ML research involves several steps, including data collection, feature engineering, model selection, and evaluation. Here's a suggested model architecture for this task:

➢ *Dataset Preprocessing*

The dataset for SDLC model selection is a valuable resource for training ML models to predict the most suitable SDLC model based on various criteria. However, before feeding the dataset into ML algorithms, it's essential to preprocess the data to ensure its quality and compatibility with the chosen modeling techniques. The key steps involved in data preprocessing for this dataset are:

➢ Handling Missing Values: Conduct a thorough examination of the dataset to identify any missing values in the attributes. Decide on an appropriate strategy for handling missing values, such as imputation or removal of rows/columns with missing data. Encoding categorical variables, feature scaling, data splitting, split the dataset into training and testing sets to facilitate model training and evaluation, addressing class imbalance.

➢ Model Selection: Choose an appropriate ML algorithm or ensemble methods and deep learning (DL) algorithms for predicting the suitable SDLC model for a given software development project. Possible algorithms include:

- Classification algorithms: Decision trees (DT), Naïve Bayes (NB), K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), and Convolutional Neural Network (CNN).
- Ensemble methods: Random Forest (RF).

➢ Model Training:

- Split the dataset into training, validation, and testing sets.
- Train the selected ML model on the training data, using the validation set to tune hyperparameters and prevent overfitting.

➢ Evaluation Metrics and Model Evaluation:

- Define evaluation metrics to assess the performance of the model. Since this is a classification task (selecting an SDLC model), metrics such as accuracy, precision, recall, F1-score, and confusion matrix can be used to evaluate the model's performance.
- Evaluate the trained model on the test dataset to estimate its performance on unseen data. Analyze the model's predictions and compare them to the actual SDLC models used in the projects.

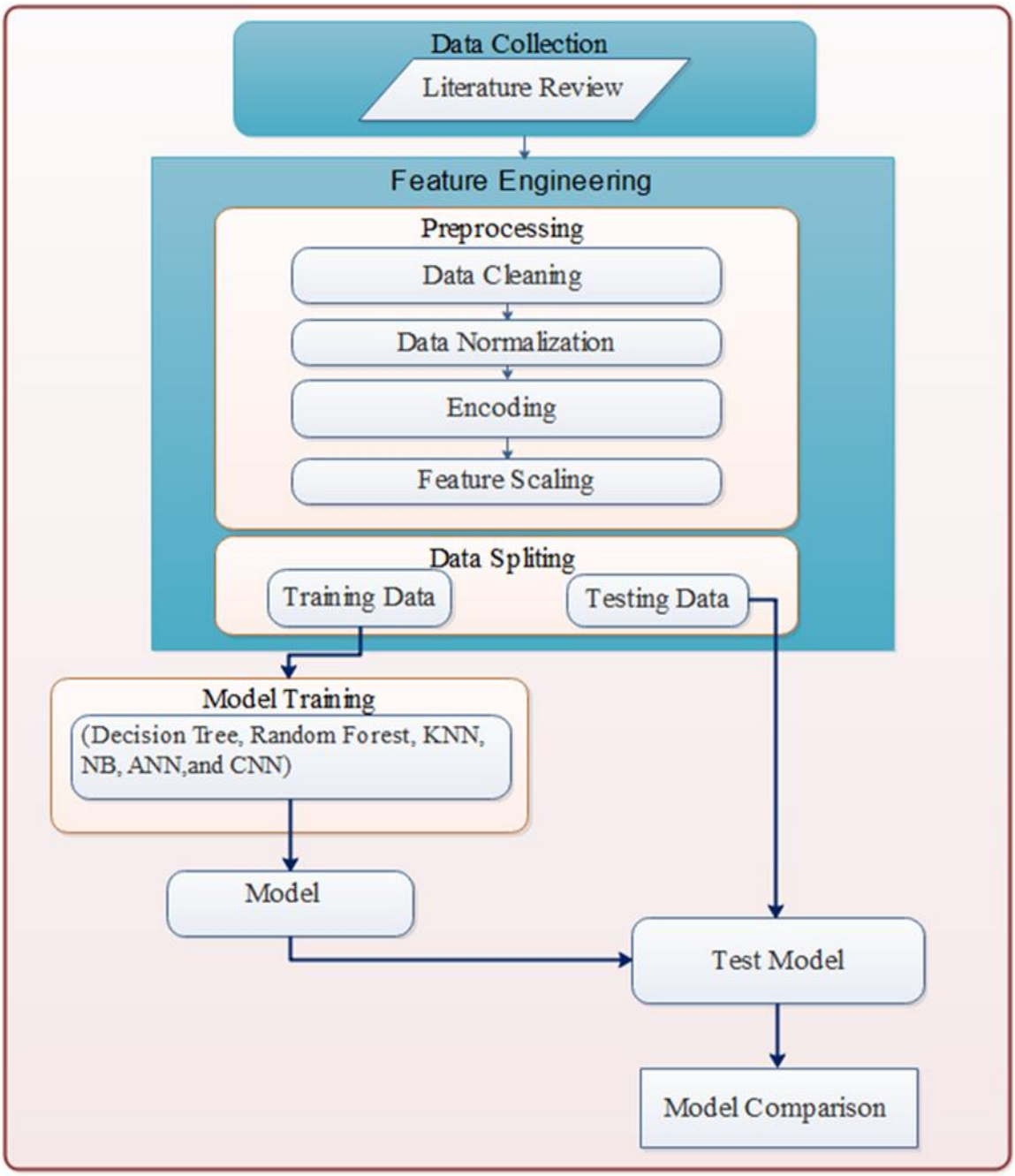

Figure 1 The Proposed Model Architecture

4 RESULT AND DESCUSSION

4.1 Dataset Setup

They developed a comprehensive dataset to facilitate the selection of SDLC models using ML techniques. This dataset was meticulously crafted through a series of steps outlined below:

4.1.1 Data Collection and Compilation

Their dataset compilation process commenced with an extensive review of existing literature, aimed at gathering factual information regarding the relationship between various criteria and formal SDLC models. This involved synthesizing insights from numerous scholarly papers, with a total of 12 papers directly contributing to their dataset preparation[9], [12], [13], [14], [15], [8], [16], [17], [7].

1) Rule Generation

From the amassed literature, they distilled 121 unique rules governing the relationship between SDLC models and criteria. These rules served as the foundation for labeling the dataset, enabling us to categorize instances based on their adherence to specific criteria-model relationships.

2) Labeling Process

Utilizing the generated rules, they labeled each instance in the dataset according to its conformity with the established criteria-model relationships. Instances that did not align with any rule were assigned the class "unknown." Subsequently, a total of 85,370 instances were successfully labeled with one of the eight formal SDLC models, while 701,062 instances remained categorized as "unknown".

3) Dataset Refinement

Recognizing the necessity of a dataset comprising only labeled instances, they removed rows assigned the "unknown" class, resulting in a final dataset of 85,370 instances. This dataset encapsulates the essential relationships between criteria and SDLC models, devoid of ambiguities associated with unlabeled instances.

The dataset is a structured, numeric dataset designed for multi-class classification tasks. It consists of 8 distinct classes, each represented by integer values ranging from 0 to 7. The dataset is composed of 10 features, each feature taking on values from the set {-1, 0, 1, 2}.

The features representing a different aspect of the data that influences the classification outcome. Each feature is assumed to be independent and not necessarily correlated with others. The values {-1, 0, 1, 2} could indicate different levels or states, where: -1 represents the feature value which is low/small in the literature, 0 represents a neutral which is empty value in the literature, 1 represents the feature value which is medium/intermediate in the literature, and 2 represent the feature value which is high in the literature.

The target variable (class) is an integer from 0 to 7, with each integer denoting a specific class. The representation of numbers in the class are: v-model (0), waterfall (1), iterative (2), incremental (3), spiral (4), agile (5), prototype (6), and RAD (7). Each class is exclusive, meaning that any given sample belongs to one and only one class. The classes are defined in such a way that they collectively cover scenarios in software project development.

This research focused on ten (10) main criteria namely product delivery (PD), clear requirement specification (CRS). Requirement change (RC), user involvement (UI), project size (PS), project risk management (RM), project complexity (CP), team experience (TE), project budget (PB), and reusability of components (RUC) were selected as a criterion for system development model selection.

Table 1 Sample Dataset

PD	CRS	RC	UI	PS	RM	CP	TE	PB	RUC	Class
2	2	2	2	2	1	2	-1	-1	2	5
0	1	2	2	0	0	2	0	1	2	5
-1	-1	-1	-1	-1	-1	1	2	-1	-1	1
0	1	2	2	0	0	2	0	2	-1	6
1	2	-1	-1	2	-1	2	-1	-1	-1	1
0	1	2	2	0	0	2	1	-1	-1	6
0	1	2	2	0	0	2	1	-1	0	6
0	1	2	2	0	0	2	1	-1	2	5
0	1	2	2	0	0	2	1	1	0	6
0	1	2	2	0	0	2	1	1	2	5
0	2	-1	-1	-1	2	2	1	2	0	3
0	2	-1	-1	0	-1	-1	-1	-1	0	1

A. Performance of Traditional Classifiers:

- ✓ K-Nearest Neighbors:
  - KNN achieved an accuracy of 99% on the test set.
  - KNN performed admirably, with a high accuracy score, indicating its effectiveness in classifying the SDLC model selection dataset. However, its performance might be sensitive to the choice of hyperparameters, such as the number of neighbors.
- ✓ Random Forest:
  - RF achieved an accuracy of 99.9% on the test set.
  - RF demonstrated outstanding performance, nearly achieving perfect accuracy. Its ensemble approach helped mitigate overfitting and handle complex relationships within the data. This suggests RF as a robust choice for SDLC model selection.
- ✓ Decision Tree:
  - DT achieved an accuracy 99.9% on the test set.
  - DT ability to perfectly classify the data might indicate overfitting, especially on a relatively small dataset. While DT offer interpretability, caution should be exercised due to their propensity for overfitting.
- ✓ Naïve Bayes:
  - NB achieved an accuracy of 67.9% on the test set.
  - NB showed significantly lower accuracy compared to other models. This could be due to its assumption of feature independence, which might not hold true for the SDLC model selection dataset. NB might not be suitable for capturing complex relationships in the data.

B. *Performance of Neural Network Models:*

✓ Artificial Neural Network:

- ANN achieved an accuracy of 96% on the test set.
- ANN showed competitive performance, falling slightly short of the best-performing traditional classifiers. ANN's ability to capture non-linear relationships makes it a viable option for SDLC model selection tasks, especially with larger datasets.

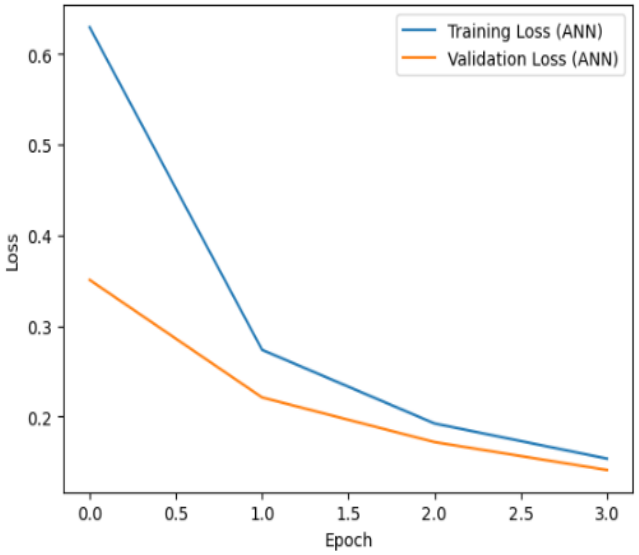

Figure 2 Training Accuracy of ANN

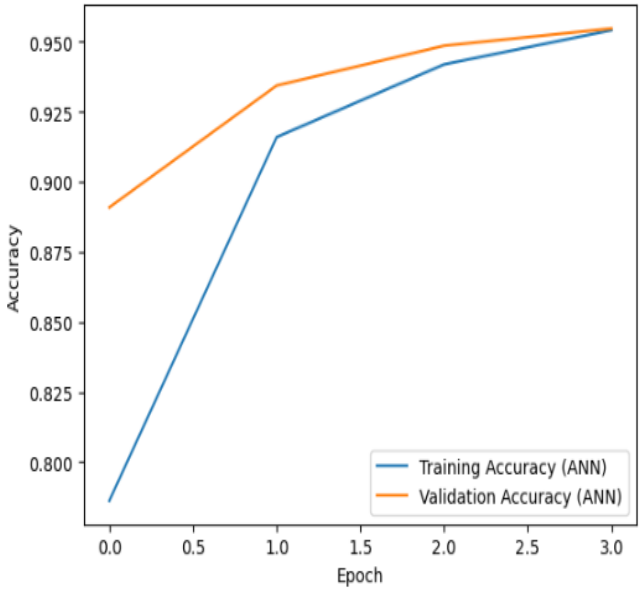

Figure 3 Training Loss of ANN

✓ Convolutional Neural Network:

- CNN achieved an accuracy of 97% on the test set.
- CNN demonstrated excellent performance, comparable to Random Forest and Decision Tree classifiers. Its sequential data processing capabilities might have been advantageous for extracting patterns in the SDLC model selection dataset.

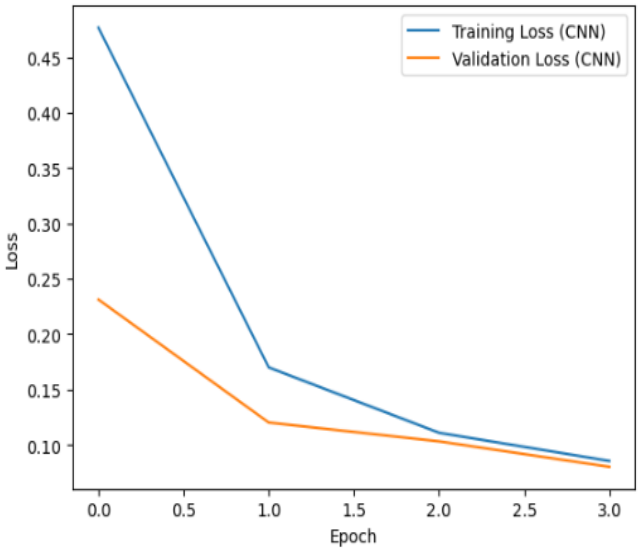

Figure 4 Training Accuracy of CNN

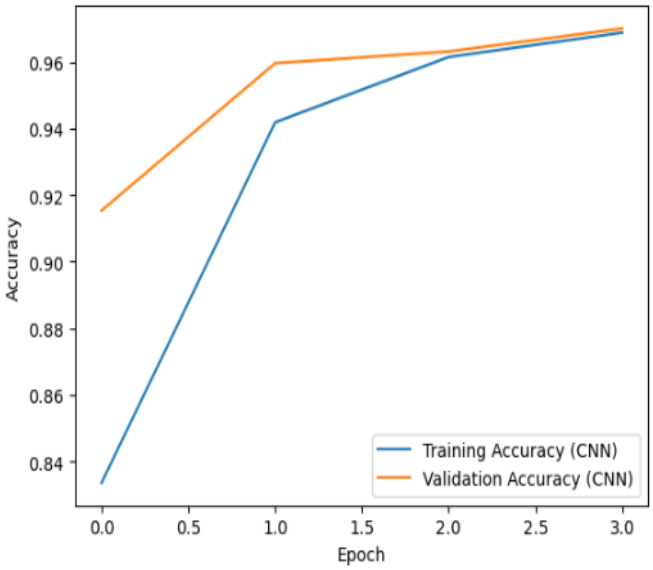

Figure 5 Training Loss of CNN

Comparison and Implications: Among traditional classifiers, Random Forest and Decision Tree classifiers outperformed others, showcasing their effectiveness in handling complex classification tasks. While Naive Bayes showed the lowest accuracy, it still serves as a baseline model. Its simplicity and speed make it suitable for quick prototyping and benchmarking against more sophisticated models. Neural network models, particularly CNN, exhibited competitive performance, demonstrating their potential in capturing intricate patterns within the SDLC model selection dataset. The choice between traditional classifiers and neural network models depends on factors such as dataset size, interpretability requirements, and computational resources available.

Table 2 Experiment Result of each Algorithm

	Model	Accuracy
1	KNN	99%
2	Random Forest	99.9%
3	Decision Tree	99.9%
4	Naive Bayes	67.9%
5	ANN	96%
6	CNN	97%

Based on the results obtained from their experiment, the Decision Tree and Random Forest models emerge as the top performers in terms of accuracy, achieving near-perfect scores of 99.9%. These models demonstrate robust performance in capturing the underlying patterns and relationships within the dataset. Therefore, for this particular dataset, employing Decision Tree or Random Forest algorithms would be advisable to achieve the highest accuracy in classification tasks. Random Forest and Decision Trees can manage both numerical and categorical data due to their ability to handle mixed data types. This flexibility allows them to work effectively with diverse datasets. However, it is crucial to account for the dataset's specific characteristics, such as its size, complexity, and the need for interpretability, before choosing the final algorithm. Nonetheless, the superior performance of Decision Tree and Random Forest models underscores their efficacy in handling the classification task at hand.

5 CONCLUSION

In conclusion, their research has systematically evaluated various ML algorithms for SDLC model selection using a comprehensive dataset. Through rigorous experimentation, they have identified Decision Tree and Random Forest as the top-performing models, achieving exceptional accuracy rates of 99.9%. These models demonstrate robustness in capturing underlying patterns within the dataset, showcasing their effectiveness in classification tasks related to SDLC model selection. Additionally, KNN, CNN, and ANN models exhibited commendable accuracy, further validating their suitability for handling structured data with complex relationships. Despite the comparatively lower accuracy of Naive Bayes, their findings underscore the importance of considering algorithmic strengths and dataset characteristics when selecting the most suitable model.

6 CONTRIBUTION

This research makes a significant contribution to the field of SDLC model selection by providing empirical evidence of the performance of various ML and DL algorithms. By systematically evaluating and comparing these algorithms using real-world data, they offer valuable insights into their effectiveness for SDLC model selection tasks. Their findings serve as a guide for practitioners and researchers in selecting the most appropriate model based algorithms on their specific requirements and dataset characteristics. Additionally, this research contributes to advancing the understanding of the applicability of ML techniques in software engineering domains.

For future work they recommend incorporating the Agile framework SDLC models like Scrum, Kanban, Extreme Programming (XP), Rational Unified Process (RUP) etc.

7 REFERENCES

[1] A. Mandal, “International Journal of Computer Sciences and Engineering Open Access Identifying the Reasons for Software Project Failure and Some of their Proposed Remedial through BRIDGE Process Models,” no. January 2015, 2016.

[2] A. Scalability and M. Ahmed, “Systematic Literature Review of Agile Scalability for Large Scale Projects,” vol. 6, no. 9, pp. 63–75, 2015.

[3] A. Aitken and V. Ilango, “A comparative analysis of traditional software engineering and agile software development,” *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 4751–4760, 2013, doi: 10.1109/HICSS.2013.31.

[4] M. Jørgensen and M. Shepperd, “A systematic review of software development cost estimation studies,” *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 33–53, 2007, doi: 10.1109/TSE.2007.256943.

[5] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, “An effective approach for software project effort and duration estimation with machine learning algorithms,” *J. Syst. Softw.*, vol. 137, pp. 184–196, 2018, doi: https://doi.org/10.1016/j.jss.2017.11.066.

[6] S. Shafiq, A. Mashkoor, C. Mayr-dorn, and A. Egyed, “Machine Learning for Software Engineering : A Systematic Mapping Machine Learning for Software Engineering : A Systematic Mapping ⋆,” no. June, 2020.

[7] K. Kumar, “A Rule-based Recommendation System for Selection of Software Development Life Cycle Models,” vol. 38, no. 4, pp. 1–6, 2013, doi: 10.1145/2492248.2492269.

[8] V. Ozturk, “Selection of appropriate software development life cycle using fuzzy logic,” no. January, 2016, doi: 10.3233/IFS-120686.

[9] A. A. Adanna and O. F. Nonyelum, “Criteria for choosing the right software development life cycle method for the success of software project,” *J. Innov. Comput.*, vol. 1, no. 1, pp. 16–26, 2020, [Online]. Available: https://www.iraseat.com/wp-content/Data/JIC/V001_I01_A04_JIC-20-007.pdf

[10] J. Dhami, N. Dave, O. Bagwe, A. Joshi, and P. Tawde, “Deep Learning Approach to Predict Software Development Life Cycle Model,” *2021 7th IEEE Int. Conf. Adv. Comput. Commun. Control. ICAC3 2021*, 2021, doi: 10.1109/ICAC353642.2021.9697271.

[11] R. Elio, J. Hoover, I. Nikolaidis, M. Salavatipour, L. Stewart, and K. Wong, “About Computing Science Research Methodology”.

[12] M. I. H. -, “Software Development Life Cycle (SDLC) Methodologies for Information Systems Project Management,” *Int. J. Multidiscip. Res.*, vol. 5, no. 5, 2023, doi:

10.36948/ijfmr.2023.v05i05.6223.

[13] L. Chandi, C. Silva, and T. Gualotu, "Model for Selecting Software Development Methodology," vol. 2, no. Icits, 2018, doi: 10.1007/978-3-319-73450-7.

[14] V. Rastogi, "Software Development Life Cycle Models-Comparison , Consequences," vol. 6, no. 1, pp. 168–172, 2015.

[15] D. Singh, A. Thakur, and A. Chaudhary, "A Comparative Study between Waterfall and Incremental Software Development Life Cycle Model," *Int. J. Emerg. Trends Sci. Technol.*, vol. 02, no. 04, pp. 2202–2208, 2015, [Online]. Available: www.ijetst.in

[16] P. M. Khan and M. M. S. S. Beg, "Extended decision support matrix for selection of sdlc-models on traditional and agile software development projects," *Int. Conf. Adv. Comput. Commun. Technol. ACCT*, vol. 3, no. 1, pp. 8–15, 2013, doi: 10.1109/ACCT.2013.12.

[17] R. R. Raval, "Comparative Study of Various Process Model in Software Development," no. November 2013, 2015, doi: 10.5120/14263-2363.