SOFTWARE FAULT PREDICTION

As software systems continue to play a crucial role in various domains, ensuring their quality and reliability has become a paramount concern. Software defects, if left undetected, can lead to failures, malfunctions and dissatisfied users.

Presented by-
Ayush kumar
Shreyansh Swaroop
Anupam Adarsh

What is a Software

Software is a collection of instructions, information, or computer programmes needed to run a computer and carry out particular activities. Software instructs a computer on how to run. Most computers would be worthless without software. As an illustration, a web browser is a piece of software that enables people to access the internet. The software programme known as an operating system (OS)

Software Fault Prediction

One crucial step in the software development process is the accurate identification of software defects, especially with the rapid improvements in software technology and increasing complexity of software. Reducing software defects is very desired for a software development project. The primary objective of creating program patterns is to locate software defects.

Finding defects in a vast and sophisticated software system is the major challenge of software fault detection.

Problem Statement

What exactly ..

In the ever-evolving landscape of software development, the identification and mitigation of software faults play a crucial role in ensuring the reliability and performance of applications. Despite advancements in development methodologies and testing techniques, software faults continue to pose challenges, leading to system failures, downtime, and increased maintenance costs. Traditional approaches to fault prediction often rely on manual inspection and rule-based heuristics, which may not scale effectively in complex and dynamic software environments.

Aim

The aim of this project is to address the limitations of existing fault prediction methods by leveraging the power of machine learning algorithms. The challenge lies in developing a robust and accurate model that can automatically identify potential software faults before they manifest into critical issues.

LITERATURE SURVEY

PAPER NAME	AUTHOR	ALGORITHMS USED	PERFORMANCE METRICS	DATASET USED	DESCRIPTION
Fuzzy Rule Based Approach for Software Fault Prediction	Singh et al., 2016	C4.5, Random Forest, Naïve Bayes	Error Percentage	KC1, KC2, PC1, CM1, KC3	Firstly, an exploration into identifying if fuzzy rule-based systems could play a substantial role in enhancing such predictions would be a step forward towards accurate diagnosis and resolution.
A Framework for Software Defect Prediction and Metric Selection	Huda et al., 2017	SVM, MR, ANNIGMA, MR-ANNIGMA, MR-SVM	Accuracy, AUC	CM1, JM1, AR1, KC1, KC2, PC1, PC3, PC4	The scope of the current investigation was procedural metrics only. Therefore, the authors are encouraged to broaden their investigation to encompass OO metrics
A Novel Approach for Software Defect Prediction through Hybridizing Gradual Relational Association Rules with Artificial Neural Networks	Miholca et al., 2018	HyGRAR, ANN	Accuracy, Sensitivity, Specificity, Precision, AUC FPR, FNR, OE	Ar1, Ar3, Ar4, Ar5, Ar6 JEdit 4.0, JEdit 4.2	Future research will focus on leveraging deep learning architectures to perform automatic extraction of features as opposed to manufactured ones. HyGRAR could be enhanced in the future to produce even better outcomes.
Deep Neural Network Based Hybrid Approach for Software	Manjula et al., 2018	Genetic Algorithm, Deep Neural Network,	Precision, Sensitivity, Specificity, Recall, F-score, Accuracy	KC1, KC2, CM1, PC1, JM1	The approach may be further tested on a larger range of datasets in the future to confirm its efficacy in other software development settings

The Key challenge

By addressing these challenges, the project aims to contribute to the development of a reliable and efficient software fault prediction model that can enhance the overall quality and maintainability of software applications.

- **Data Complexity**

  · Software development involves a multitude of interconnected factors, including code complexity, developer activity, and external dependencies. Extracting relevant features from diverse and intricate datasets poses a significant challenge.

- **Imbalanced Data**

  · Software faults are often rare events in comparison to the overall codebase. Imbalanced datasets can affect the performance of machine learning models, leading to biased predictions. Developing strategies to handle imbalanced data is crucial for the success of the fault prediction model.

- **Dynamic Software Environments**

  · Software systems are subject to continuous changes and updates, making it challenging to build a fault prediction model that can adapt to dynamic environments. The model should be able to learn from evolving patterns and generalize well across different software versions.

- **Integration with Development Workflow**

  · For effective fault prediction, the model needs to seamlessly integrate into the existing software development workflow. Ensuring practical usability and acceptance by developers is critical for the success of the project.

Objective

The primary objective of the project is to develop an effective and accurate model that can predict software faults and defects in a given codebase. The project aims to leverage advanced machine learning techniques to analyze data, identify patterns, and create a predictive model capable of anticipating potential software faults before testing phase in the development life cycle.

The key goals of the project include:

- **Model Development:**

  Design and implement a robust machine learning model that can analyze datasets, software metrics, and code complexity.

- **Feature Selection:**

  Identify and select the most influential features out of all features that contribute to the prediction of software faults.

- **Data Collection and Preprocessing**

  Gather relevant datasets and preprocess the datasets to ensure its suitability for training and testing the machine learning model. This involves cleaning, transforming, and normalizing the data to enhance the model's performance.

- **Model Training and Validation**

  Train the machine learning model using a labeled dataset and validate its performance using a separate set of data.

The key goals of the project include:

- **Integration with Development Workflow**

  Explore ways to integrate the fault prediction model seamlessly into the software development workflow, allowing developers to proactively address potential issues during the coding and testing phases.

- **Documentation and Reporting:**

  Provide comprehensive documentation of the model development process, including methodologies, algorithms used, and parameters chosen. Generate reports highlighting the model's effectiveness and its potential impact on software development practices.

- **Evaluation and Comparison**

  Evaluate the performance of the developed model against existing fault prediction methods and benchmarks. Compare the accuracy and efficiency of the machine learning-based approach with traditional methods.

DESCRIPTION OF DATASET

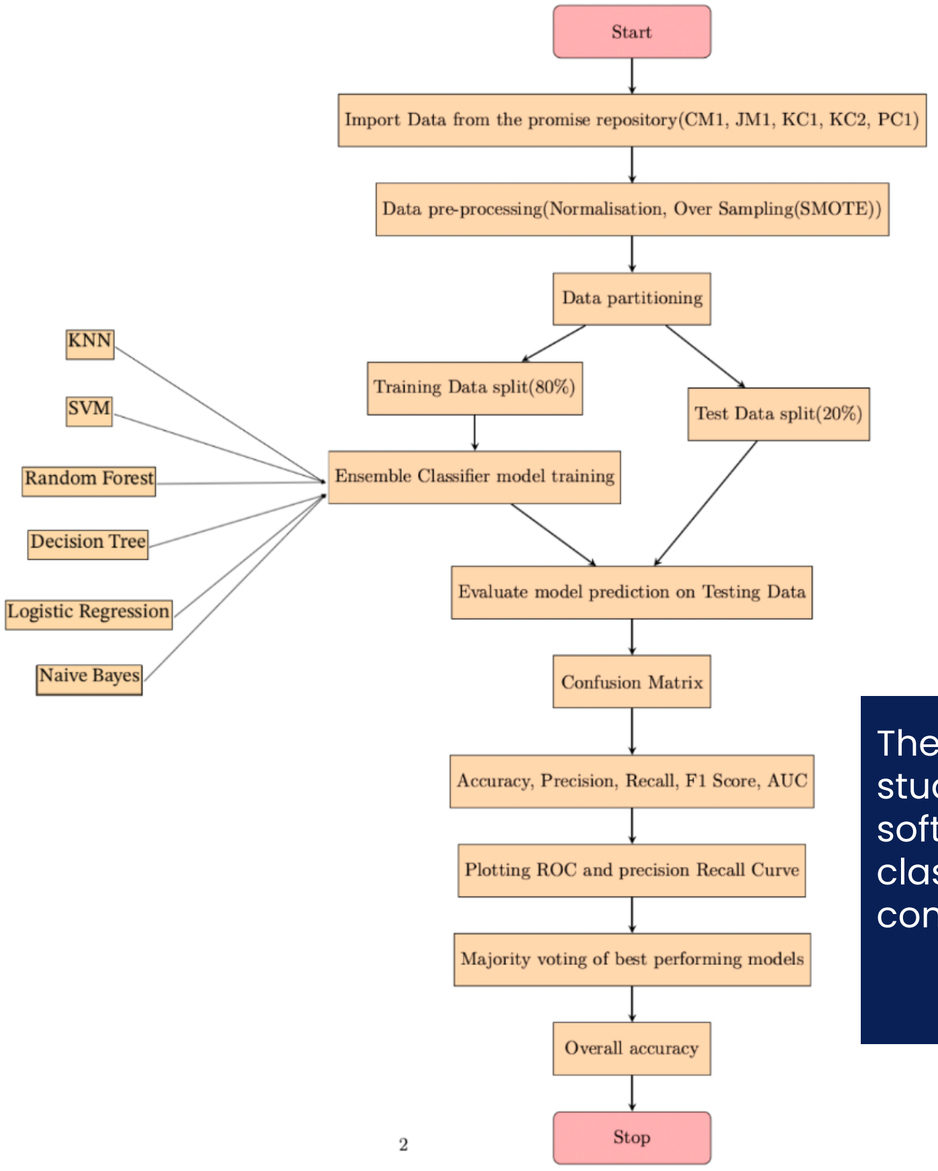
In this project, we have worked on five datasets from NASA's PROMISE Repository of Software Engineering

Dataset	No. of Features	No. of Instances	Faulty Instances	Non-Faulty Instances
KC1	22	498	49 (9.839%)	449 (90.161%)
JM1	22	10885	2106 (19.348%)	8779 (80.652%)
CM1	22	2109	326 (15.458%)	1783 (84.542%)
KC2	22	522	107 (20.498%)	415 (79.502%)
PC1	22	1109	77 (6.944%)	1032 (93.056%)

- NASA's CM1 dataset contains valuable information collected by their spacecraft instrument and is implemented in the programming language "C".
- Similarly, JM1 dataset refers to a real-time predictive ground system coded using "C".
- Furthermore, KC1 takes things up a notch by integrating storage management for efficient reception and processing of ground data through its incorporation of "C++" technology.
- The KC2 dataset is also pertinent with regards to implementing C++ functions that support complex science-data processing requirements.
- The PC1 dataset provides insightful flight-software related information derived from C-functions linked to an Earth orbiting satellite.

THE PROPOSED SOFTWARE FAULT PREDICTION MODEL

We propose a comprehensive solution for predicting software faults different base classifiers algorithms. Our approach involves applying multiple (6) different base classifiers, to build initial fault prediction models. These models are evaluated using accuracy as the primary performance metric, along with other performance metrics such as precision recall, F1 score, AUC and execution time. Based on the evaluation results, the best performing model is chosen.



The proposed solution in this research study attempts to address prediction of software faults using different base classifiers algorithms. The methodology consists of several steps:

The steps are:

SELECTION OF BASE CLASSIFIERS

Six different base classifiers namely KNN, SVM, Random Forest, Decision Tree, Logistic Regression, and Naïve Bayes were chosen to build initial fault prediction models. These classifiers were diverse in nature, enabling a comprehensive exploration of different classification techniques.

EVALUATION OF THE CLASSIFIERS

The base classifier models were assessed for their performance through multiple metrics, which includes accuracy, precision, recall, F1 score, area under the curve and execution time. Accuracy was considered the primary metric for evaluating model performance.

IDENTIFICATION OF THE BEST MODEL

Based on the evaluation results, K-Nearest Neighbour (KNN) was selected as the best performing model among the initial classifiers based on Accuracy.

PERFORMANCE METRICS

Our study evaluates multiple performance measures of software fault prediction models. Accuracy is our primary focus, but we also consider recall, precision, F1 score (F-Measure) and area under the ROC curve (AUC).

- **Accuracy:**

To gauge the correctness of our expectations, we often turn to accuracy as a benchmark. This calculation involves comparing the total number of successful forecasts to all predictions made, providing insight into their level of correctness.

- **Precision:**

For a reliable prediction, precision comes into play by striving for accuracy with positive cases. It is the computation of the proportion between correct positive predictions and all actual positives.

- **Recall**

Recall, however, places importance on the comprehensiveness of forecasts and gauges the proportion of actual positives that are correctly identified as such.

- **F1 score**

  The F1 score is a fair measure that accounts for both incorrect positive and negative results, uniting accuracy with completeness by calculating their average harmonic. The effectiveness of a software fault prediction model is evaluated by measuring its ability to differentiate between positive and negative samples over all classification thresholds through Area Under the Curve (AUC) values.

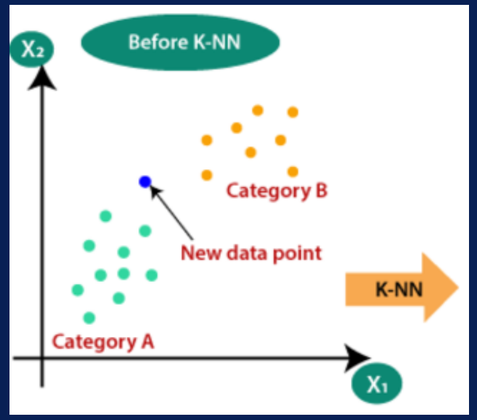
- **Area under Curve**

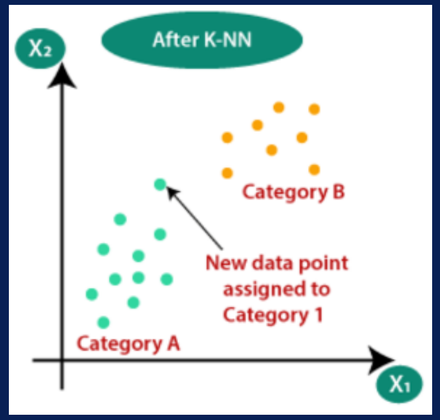
Larger AUC values indicate superior performance. To gain insights into their predictive abilities and handling of imbalanced datasets, we utilize these performance metrics in our assessment of these models.

IMPLEMENTATION

K-NEAREST NEIGHBOUR (KNN)

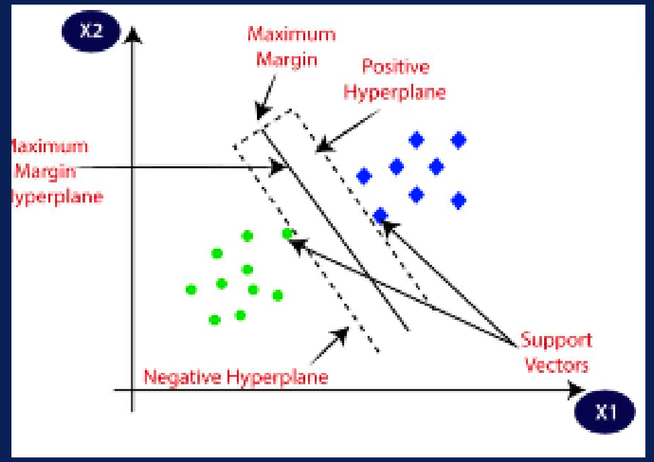
K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity.





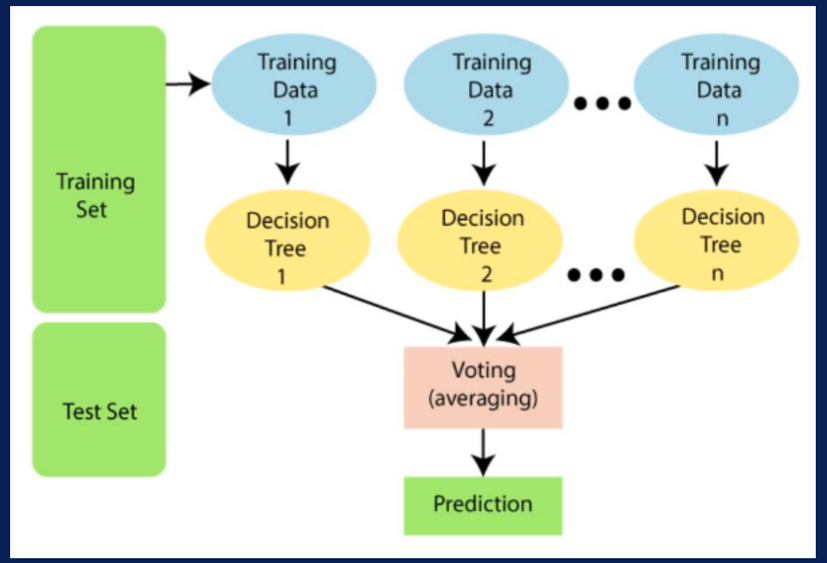
SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.



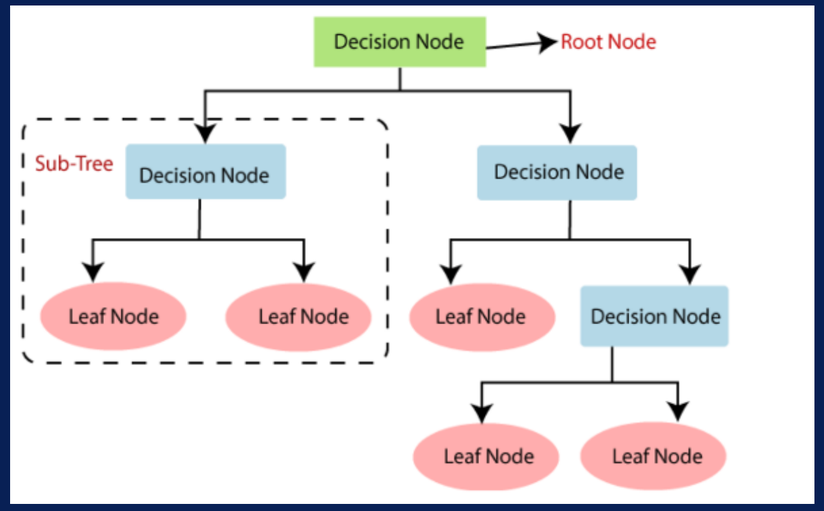
RANDOM FOREST

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model



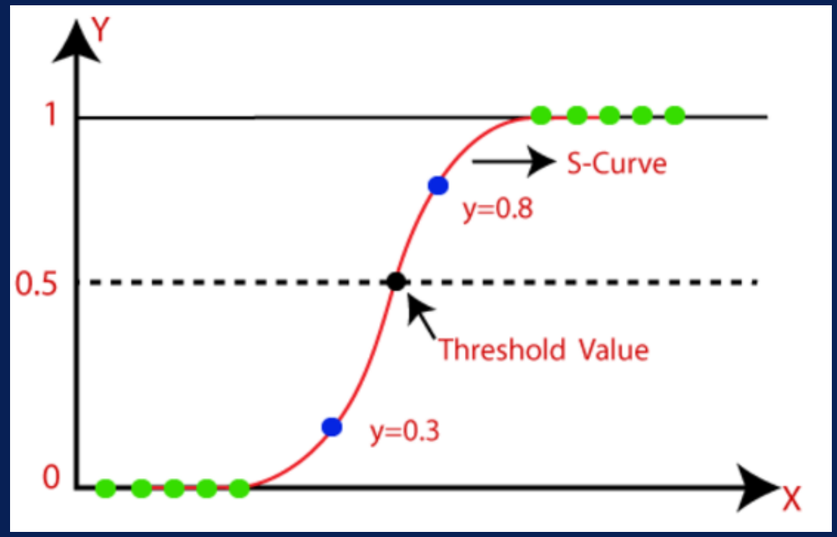
DECISION TREE

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.



LOGISTIC REGRESSION

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. It can be either Yes or No, 0 or 1, true or False, etc.



NAIVE BAYES

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset.

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

EXPERIMENTAL RESULTS

For each of the five datasets, software fault prediction algorithms were executed. We evaluated the performance of multiple classifiers, including KNN, SVM, Random Forest, Decision Tree, Logistic Regression, and Naïve Bayes. Performance metrics, like accuracy, precision, recall, F1 score, AUC and execution time were used to assess the effectiveness of these classifiers. The results, as shown in the tables, highlight the best performing classifier for each dataset, indicated in bold.

ACCURACY RESULTS

- We can observe that K-Nearest Neighbour (KNN) consistently performs well across different datasets, with accuracy scores ranging from 88.0% to 92.3423%. It is competitive compared to other algorithms, achieving comparable or higher accuracy. KNN's performance is consistent and robust across datasets. Optimization opportunities exist for fine-tuning hyperparameters and exploring distance metrics. KNN's simplicity and interpretability make it a reliable choice for classification tasks

Algorithm	CM1	JM1	KC1	KC2	PC1
KNN	88.0	78.135	82.7014	77.1428	94.3423
SVM	88.0	80.6614	84.5971	80.00	91.4414
Random Forest	86.0	81.7638	83.1753	78.0952	91.4414
Decision Tree	86.0	74.9196	82.2274	73.3333	89.1891
Logistic Regression	88.0	80.8911	85.3080	83.8095	90.9909
Naïve Bayes	88.0	80.6155	81.0426	85.7142	87.3873

RECALL RESULTS

- We can observe that KNN achieved perfect recall (1.00) for the CM1 dataset, indicating its strong ability to identify true positives. Additionally, SVM also demonstrated high recall scores (1.00), implying their effectiveness in classifying positive instances for multiple datasets. Moreover, Random Forest showed consistently high recall scores across various datasets, indicating their robustness in capturing positive instances. KNN's exceptional performance suggests its potential as a reliable algorithm for recall-oriented tasks.

Algorithm	CM1	JM1	KC1	KC2	PC1
KNN	0.88	0.8267	0.8656	0.8181	0.9266
SVM	0.8800	0.8093	0.8628	0.7979	0.9144
Random Forest	0.8775	0.8397	0.8756	0.8131	0.9220
Decision Tree	0.9111	0.8466	0.8849	0.7954	0.9241
Logistic Regression	0.8800	0.8135	0.8656	0.8461	0.9216
Naïve Bayes	0.8958	0.8255	0.8791	0.8571	0.9310

Algorithm	CM1	JM1	KC1	KC2	PC1
KNN	0.88	0.8267	0.8656	0.8181	0.9266
SVM	0.8800	0.8093	0.8628	0.7979	0.9144
Random Forest	0.8775	0.8397	0.8756	0.8131	0.9220
Decision Tree	0.9111	0.8466	0.8849	0.7954	0.9241
Logistic Regression	0.8800	0.8135	0.8656	0.8461	0.9216
Naïve Bayes	0.8958	0.8255	0.8791	0.8571	0.9310

F1 SCORE RESULTS

- We can observe that several algorithms consistently achieved high F1 scores across multiple datasets, including KNN, SVM, Random Forest, and Logistic Regression. This suggests their effectiveness in handling diverse data patterns. Notably, KNN consistently attained the highest F1 scores among all algorithms across all datasets, indicating its robustness and suitability for the given classification tasks.

AUC RESULTS

- We can observe that Firstly, the Random Forest exhibit high AUC scores, suggesting their effectiveness. Additionally, Naïve Bayes demonstrate competitive performance.

Algorithm	CM1	JM1	KC1	KC2	PC1
KNN	0.6250	0.6492	0.6955	0.7015	0.6156
SVM	0.2320	0.6402	0.4772	0.7662	0.4163
Random Forest	0.7296	0.7535	0.8199	0.7240	0.8841
Decision Tree	0.6325	0.5855	0.5996	0.5452	0.5805
Logistic Regression	0.4517	0.4537	0.5539	0.6462	0.5447
Naïve Bayes	0.7793	0.6636	0.7710	0.7537	0.7039

EXECUTION TIME RESULTS

Algorithm	CM1	JM1	KC1	KC2	PC1
KNN	0.0331	0.2645	0.0619	0.0327	0.0570
SVM	0.0631	5.7844	0.5365	0.0694	0.0841
Random Forest	0.1826	2.1781	0.3108	0.1711	0.2332
Decision Tree	0.0525	0.3586	0.0848	0.2057	0.2078
Logistic Regression	0.0756	0.4223	0.1456	0.0749	0.0617
Naïve Bayes	0.1702	0.4159	0.0627	0.2240	0.1133

FUTURE WORK

Based on the evaluation results, K-Nearest Neighbor (KNN) was selected as the best performing model among the all initial classifiers based on Accuracy. So, the selected algorithm is K-Nearest Neighbor (KNN).

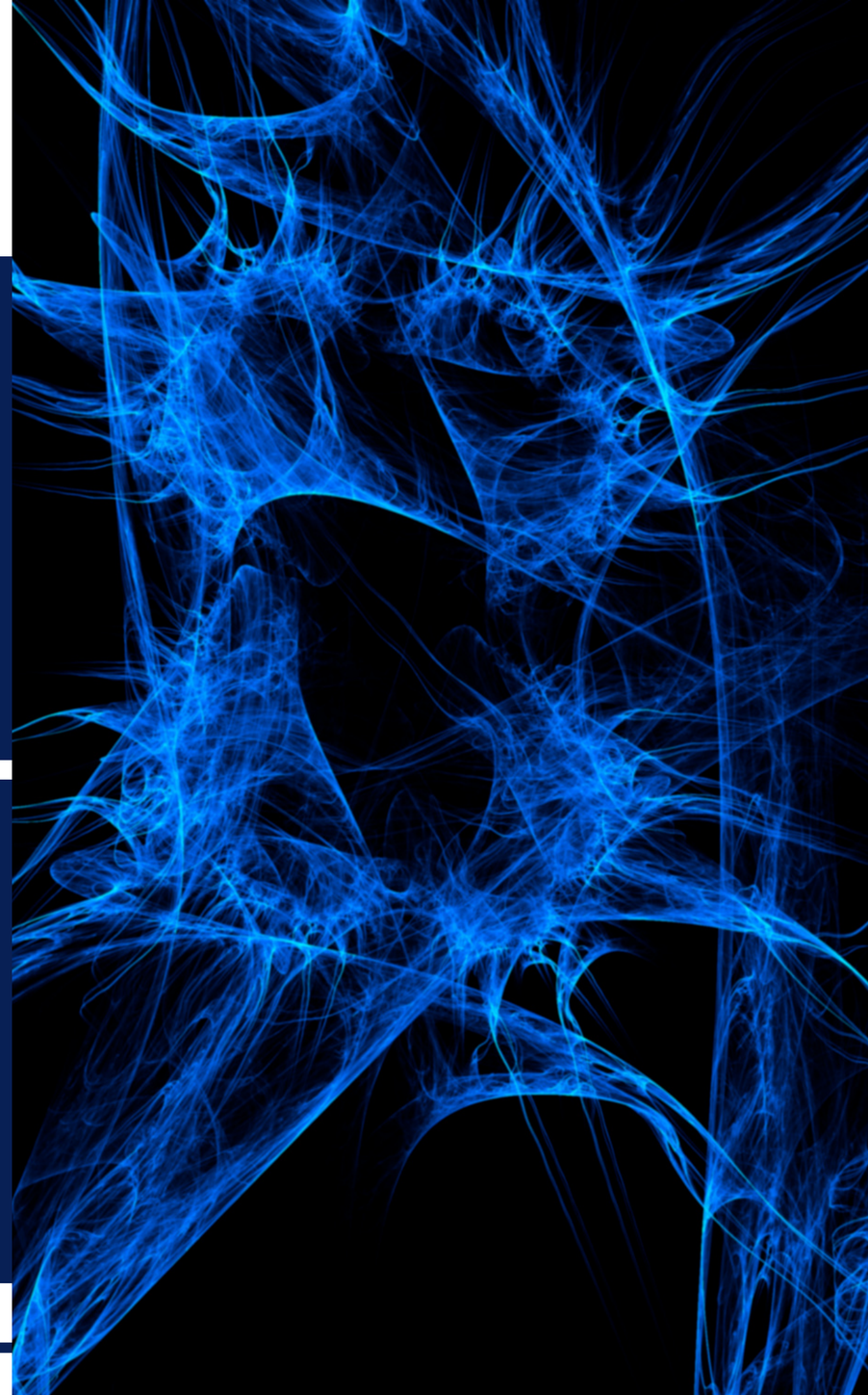
To enhance the accuracy of the selected algorithm i.e. KNN, we can employ a bio-inspired optimization algorithm. This algorithm leverages the principles of nature-inspired computation to refine the model and achieve better predictive accuracy. By iteratively optimizing the model's parameters, the algorithm seeks to improve its ability to accurately classify software defects.

Conclusion

In conclusion, we propose a comprehensive solution for predicting software faults different base classifiers algorithms. Our approach involves six different base classifiers namely KNN, SVM, Random Forest, Decision Tree, Logistic Regression, and Naïve Bayes were chosen to build initial fault prediction models.

These models are evaluated using accuracy as the primary performance metric, along with other performance metrics such as precision, recall, F1 score, AUC and execution time.

Based on the evaluation results, K-Nearest Neighbor (KNN) was selected as the best performing model among the initial classifiers based on Accuracy.



THANK YOU

PRESENTATION