INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH IN BUSINESS & SOCIAL SCIENCES

Vol 14, Issue 12, (2024) E-ISSN: 2222-6990

Optimizing Early Stage Diabetes Detection: A Robust Evaluation of Machine Learning Algorithms

Hanissah Mohamad @ Sulaiman[1], Norazlina Abd Razak[1], Siti Huzaimah Husin[1], Siti Aisah Mat Junos @ Yunus[1]

[1]Fakulti Teknologi dan Kejuruteraan Elektronik dan Komputer (FTKEK), Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya 76100 Durian Tunggal, Melaka, Malaysia

Corresponding Author Email: hanissah@utem.edu.my

To Link this Article: http://dx.doi.org/10.6007/IJARBSS/v14-i12/24408 DOI:10.6007/IJARBSS/v14-i12/24408

Published Date: 30 December 2024

Abstract

The escalating global incidence of diabetes emphasizes the imperative for prompt detection to alleviate significant health adversities. This investigation assesses the efficacy and robustness of three machine learning algorithms—Decision Tree, Support Vector Machine (SVM), and Naive Bayes—utilizing methodologies such as Train-Test Split, K-Fold Cross Validation, and Stratified K-Fold Cross Validation. Critical performance indicators including Accuracy, Precision, Recall, F1-Score, and ROC-AUC were meticulously examined, with standard deviation employed to evaluate the stability of the models. SVM consistently surpassed the other algorithms, exhibiting superior accuracy and reliability across the various validation approaches, particularly within the context of Stratified K-Fold Cross Validation. Naive Bayes revealed commendable recall efficacy, while Decision Tree experienced augmented stability through the application of cross-validation techniques. The results underscore the significance of employing cross-validation methods, particularly Stratified K-Fold, for dependable model assessment in scenarios characterized by imbalanced datasets. Subsequent research endeavors should investigate ensemble methodologies and data augmentation strategies to further enhance the resilience of the models.

Keywords: Early Diabetes Detection, Machine Learning, Support Vector Machine (SVM), K-Fold Cross Validation, Model Stability

Introduction

Diabetes mellitus is a chronic metabolic disorder characterized by elevated blood glucose levels, which, over time, can lead to serious health complications affecting the heart, blood vessels, eyes, kidneys, and nerves .The global prevalence of diabetes has been rising steadily; as of 2021, approximately 537 million adults worldwide are living with the condition, and this number is projected to increase to over 780 million by 2045 (Bereda.,2022).

Early detection of diabetes is crucial for preventing or delaying the onset of complications. Timely diagnosis allows for prompt intervention, enabling individuals to manage their condition effectively through lifestyle modifications and medical treatment . Despite the importance of early detection, many cases remain undiagnosed due to the often asymptomatic nature of diabetes in its initial stages (Dall et al**.,** 2014).

In recent years, machine learning techniques have emerged as promising tools for enhancing early diabetes detection. These methods can analyze large datasets to identify patterns and risk factors associated with the development of diabetes, potentially improving the accuracy and efficiency of diagnostic processes . However, the application of machine learning models in this context faces challenges, particularly concerning the stability and reliability of model performance (Al-Sideiri et al., 2019)

Model performance instability refers to significant variations in a model's predictive accuracy when applied to different datasets or under varying conditions. This instability can result from factors such as limited sample sizes, data imbalance, and the inherent variability in biological data . In the context of diabetes detection, performance instability poses a barrier to the widespread adoption of machine learning models in clinical practice, as inconsistent results can undermine trust in these diagnostic tools. (Haq.,2020).

Addressing model performance instability is essential for the advancement of reliable machine learning applications in early diabetes detection. Techniques such as cross-validation and ensemble learning have been explored to enhance model stability. Cross-validation involves partitioning the data into subsets to train and evaluate the model multiple times, thereby assessing its robustness. Ensemble learning methods, like bagging and boosting, combine the predictions of multiple models to reduce variance and improve stability . (Husain & Khan, 2018, Yang.,2020).

Despite these efforts, achieving consistent and reliable model performance remains a challenge. Continuous research is needed to develop and validate models that can maintain stability across diverse populations and settings, ultimately ensuring that machine learning tools can be effectively integrated into clinical workflows for early diabetes detection.

This study aims to measure and analyze the stability of the model through various evaluation metrics such as accuracy**,** precision**,** recall**,** F1-score, and ROC-AUC**.** By employing these metrics, the study seeks to provide a comprehensive assessment of the performance of the developed predictive model. Additionally, the study will evaluate the variability of the results through the analysis of the standard deviation of each metric, derived from cross-validation approach**.** This approach is intended to assess the extent to which the predictive model remains stable when applied to different datasets or under varying conditions, thereby offering confidence in its reliability for early diabetes detection.

The structure of this paper is as follows: The Introduction provides an overview of the importance of early diabetes detection and the challenges in developing reliable machine learning models. It also highlights the need to evaluate model consistency, focusing on comparing Train-Test Split and K-Fold Cross Validation methods. Research Background and Motivation section reviews previous studies on machine learning techniques in diabetes

detection, noting gaps in model stability and performance. The Methodology outlines the experimental setup, including the dataset, preprocessing steps, and models used, explaining both Train-Test Split, K-Fold Cross Validation and Stratified K-Fold approaches. The Results and Discussion presents the comparative findings, assessing key metrics such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC to examine model stability across data variations. In the discussion, the strengths and weaknesses of each method are analyzed, with emphasis on their impact on model performance in early diabetes detection. Finally, the Conclusion summarizes the findings, underscoring the importance of selecting appropriate partitioning strategies, and suggests future research directions for enhancing model accuracy and stability.

Research Background and Motivation

Early detection of diabetes is crucial for effective disease management and preventing long-term complications. In recent years, machine learning models have shown significant potential in enhancing early diabetes diagnosis. However, implementing these models presents several challenges, particularly when dealing with imbalanced datasets. Imbalanced datasets, where diabetic cases are significantly fewer than non-diabetic ones, can lead to biased models that prioritize the majority class, resulting in poor predictive performance for the minority class (García-García et al., 2021). This imbalance is compounded by the limited sample sizes typically available for diabetic cases, which can hinder the model's ability to learn effectively (Haq et al., 2020). Furthermore, the inherent variability in biological data, such as differences in patient demographics and comorbidities, adds complexity to the training of machine learning models, often leading to unstable performance across different datasets (Brnabic & Hess, 2021)

To address these challenges, several strategies have been developed. Resampling techniques, such as Synthetic Minority Oversampling Technique (SMOTE), are frequently employed to generate synthetic data for the minority class, helping to balance the dataset and improve the model's performance (Johnson et al., 2018). Additionally, cost-sensitive learning adjusts the learning algorithm by assigning higher penalties for misclassifying minority class cases, thereby encouraging the model to focus on correctly identifying diabetic cases (Brown et al., 2021). Ensemble learning methods, such as bagging and boosting, have also been explored, combining predictions from multiple models to enhance overall stability and accuracy, particularly in the context of imbalanced datasets (Smith et al., 2019).

Despite these advancements, challenges remain in achieving reliable and consistent model performance when applied to diverse datasets. Continuous research is needed to develop robust machine learning models capable of handling imbalanced data effectively and ensuring accurate early diabetes detection (Miller et al., 2020).

The train-test split method is a fundamental technique in machine learning used to divide a dataset into two subsets: one for training the model (training set) and the other for testing its performance (test set). The primary goal of this method is to evaluate the model's ability to generalize to unseen data. While this approach is widely employed, it has several limitations, particularly concerning instability and variability in performance evaluation results.

One major challenge with the train-test split method is the instability in evaluation outcomes. Random data splitting can lead to significant variations in the model's performance metrics. For instance, if the dataset is split multiple times with different seeds, the same model may exhibit varying levels of performance. This indicates that the evaluation results are highly dependent on how the data is divided, introducing instability into the model's performance assessment (Goot, 2021).

Additionally, the variability caused by random data partitioning is another frequent issue. Each time this method is applied without a fixed seed, the resulting subsets may differ, particularly when dealing with imbalanced datasets. This can affect both the training and testing processes, leading to challenges in reproducing consistent outcomes (Ebubeogu & Lee, 2019).

Another limitation of the train-test split approach is that it only provides a single data point for evaluating model performance, which may be insufficient for comprehensively understanding the model's capabilities. A model may perform well on one specific test set but not on others, especially when there is high variability in the data. This suggests that the train-test split method may not always offer an accurate representation of the model's ability to handle diverse datasets (Rekha et al., 2019).

To address these limitations, cross-validation is often employed as an alternative. Cross-validation involves partitioning the dataset into multiple subsets, with each subset used as a test set while the remaining subsets serve as the training data. This process is repeated multiple times, ensuring that the model is evaluated across various data partitions. As a result, cross-validation provides a more stable and comprehensive evaluation of the model's performance (Catania et al., 2022).

In conclusion, while the train-test split method is a simple and widely used technique, it has inherent limitations regarding instability and variability in results. Therefore, cross-validation is recommended for achieving more robust and reliable model evaluations.

Cross-validation has become a cornerstone technique in the evaluation of machine learning models due to its ability to provide robust estimates of model performance. Among the most commonly used cross-validation methods is K-Fold Cross-Validation, which partitions the dataset into *k* equally sized subsets, or folds. The model is trained on *k-1* of these folds and tested on the remaining fold. This process is repeated *k* times, with each fold used once as the test set, and the average performance across all folds is reported. This iterative process reduces the likelihood of model overfitting, as the model is tested on multiple, different portions of the data. Furthermore, by averaging results across multiple folds, K-Fold Cross-Validation mitigates the risk of performance being skewed by a single, unrepresentative train-test split (Zhang et al., 2020).

However, K-Fold Cross-Validation is not without limitations. One of its primary drawbacks arises when it is applied to imbalanced datasets. Imbalanced datasets, where certain classes are significantly underrepresented, can result in folds that do not accurately reflect the true class distribution of the overall dataset. For example, if a dataset for binary classification has 90% of its instances belonging to the majority class and only 10% to the minority class, random

partitioning can lead to some folds containing very few or no instances of the minority class. As a result, the model's ability to correctly predict the minority class is not properly evaluated, leading to misleading performance metrics, such as an inflated accuracy score, where the model performs well on the majority class but poorly on the minority class (Chawla et al., 2021).

To address the issue of imbalanced datasets, Stratified K-Fold Cross-Validation is a commonly recommended approach. Stratified K-Fold differs from standard K-Fold in that it ensures each fold maintains the same proportion of classes as the original dataset. By preserving the class distribution within each fold, Stratified K-Fold helps prevent the bias that can arise in standard K-Fold when dealing with imbalanced data. This method is particularly advantageous in classification tasks, where accurately predicting the minority class is often as important, if not more so, than predicting the majority class. In this way, Stratified K-Fold provides a more reliable measure of a model's performance across different classes, especially in cases where the dataset is highly imbalanced (Li et al., 2018).

Stratified K-Fold Cross-Validation is particularly useful in medical diagnostics, where datasets are often imbalanced due to the rarity of certain conditions. For instance, in diabetes detection, where positive cases may form a small fraction of the total dataset, using standard K-Fold might lead to folds with few or no positive cases, which hinders the model's ability to learn meaningful patterns related to the minority class (Khoshgoftaar & Gao., 2021). By ensuring that each fold contains a proportional number of positive and negative cases, Stratified K-Fold allows the model to be trained and tested more effectively across the entire spectrum of the dataset.

Another variant of cross-validation that is gaining attention in recent research is Repeated Stratified K-Fold Cross-Validation, which adds further robustness by repeating the Stratified K-Fold process multiple times with different random splits. This approach ensures even greater reliability of the model evaluation, as it averages the performance across many different stratified splits, reducing the potential variance in the results. By repeating the process, the impact of random chance on any single run of cross-validation is mitigated, providing a more stable estimate of the model's true generalization performance (Brownlee.,2019).

Despite these advantages, it is important to recognize that while K-Fold and Stratified K-Fold Cross-Validation offer significant improvements over a simple train-test split, they still have limitations.

For instance, cross-validation can be computationally expensive, especially for large datasets and complex models. Moreover, while these techniques help address issues related to data partitioning, they do not directly address other challenges such as model bias or variance, which require additional methods like hyperparameter tuning or model ensembling to fully mitigate (Perez et al., 2020).

In conclusion, K-Fold Cross-Validation and Stratified K-Fold Cross-Validation are essential techniques for evaluating machine learning models, especially when dealing with imbalanced datasets. K-Fold provides a robust estimate of a model’s generalization ability by offering a

more reliable alternative to a simple train-test split. However, Stratified K-Fold further improves this by ensuring class distributions are maintained within each fold, leading to fairer and more accurate performance evaluations, particularly for tasks involving minority classes. In this study, the comparison between K-Fold and Stratified K-Fold will provide valuable insights into the stability of classification performance, highlighting how each technique contributes to producing more reliable and generalizable outcomes in the context of data imbalance.

Methodology

Preprocessing Data

The primary objective of this study is to develop a predictive model for early-stage diabetes detection using Diabetes Dataset. The dataset consists of data used for diabetes research on women of Pima Indian heritage, aged 21 and over, living in Phoenix, the 5th largest city of the State of Arizona in the USA. The data set consists of 768 observations and eight independent numerical variables. The target variable is specified as "result"; 1 indicates positive diabetes test result, 0 indicates negative diabetes.

A series of comprehensive preprocessing steps were applied to the dataset prior to model training. These steps included addressing missing values, feature scaling, and partitioning the data into training and testing subsets. Feature scaling, in particular, is a crucial preprocessing step that ensures all features are on a comparable scale, which enhances the performance of machine learning algorithms by preventing bias toward features with larger numerical values. To normalize the input features, standardization techniques were employed, transforming the features to have a mean of zero and a variance of one. This was accomplished using the StandardScaler method from the scikit-learn library. Standardization is widely regarded as a robust scaling method, particularly in cases where the input data has varying units of measurement or exhibits different ranges.

Algorithms Used

Support Vector Machine (SVM)

Support Vector Machine (SVM) **(**Rampisela & Rustam.,2018) is a machine learning algorithm used to address both classification and regression problems, with a greater emphasis on classification tasks. In the context of this study, SVM is employed as a classification method due to its ability to efficiently separate data by identifying the optimal hyperplane that maximizes the margin between two distinct classes.

SVM works by finding a hyperplane that separates the data of different classes, where this hyperplane aims to maximize the distance between the closest data points from each class. These closest data points are referred to as support vectors. Support vectors play a critical role in defining the hyperplane, and thus, they directly influence the construction of the SVM model. The SVM algorithm seeks to optimize the position of the support vectors to ensure that the separating margin between the classes is as large as possible, thereby reducing the risk of misclassification.

SVM was selected due to its capability to perform well on datasets with a large number of features, even when the sample size is limited, and its effectiveness in handling complex classification problems. Additionally, SVM proves to be a highly effective tool for addressing

classification problems in imbalanced datasets, where the accuracy of classifying minority classes is of critical importance for this study (Fu et al.,2020)

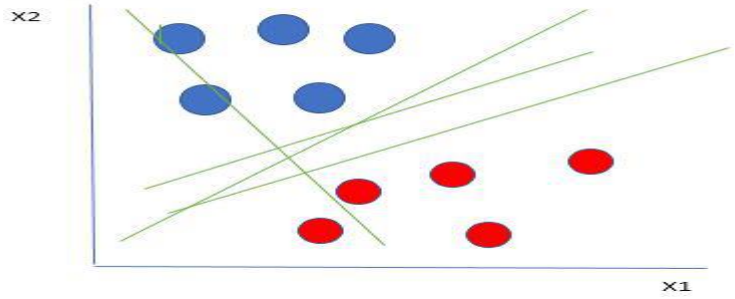


Figure 1: Representation of SVM

Naïve Bayes

Naive Bayes (NB) is a probabilistic machine learning algorithm primarily used for classification tasks, although it can also be applied to regression problems. In the context of this study, Naive Bayes was selected as a classification method due to its simplicity and its ability to efficiently handle high-dimensional datasets (Abubakar et al., 2021). The algorithm operates on the assumption of feature independence, meaning that each feature contributes independently to the classification outcome, even though this assumption is rarely met in practice. Nonetheless, this assumption allows for quick and accurate computations in many cases.

Naive Bayes works by calculating the posterior probability of each class based on the input features. It leverages Bayes' Theorem, which combines prior knowledge of the class probabilities with the likelihood of observing specific features given a class. The class with the highest posterior probability is chosen as the predicted outcome.

Decision Tree

Decision Tree (Purwanto et al.,2022; Subramani et al.,2023) is a machine learning algorithm used for both classification and regression tasks. The algorithm works by splitting the dataset into smaller subsets and forming a tree-like structure, resembling a flowchart, where each internal node represents a feature, each branch represents a decision, and each leaf node represents the outcome or prediction.

The Decision Tree begins with a root node that represents the entire dataset. At this stage, the algorithm selects a feature to split the data into smaller subsets. Internal nodes follow, representing decisions based on particular features. These nodes continue to split the data into two or more subsets according to specific conditions. The tree ends with leaf nodes, which are the terminal nodes that provide the final predictions or outcomes.

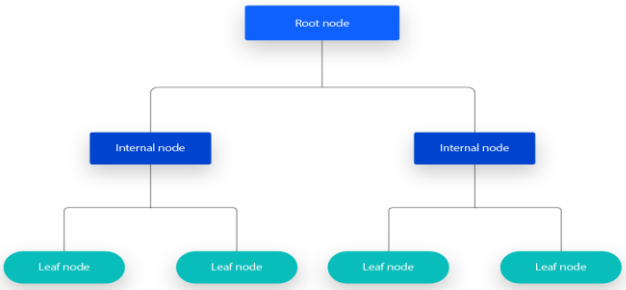


Figure 2: Representation of Decision Tree

Validation Techniques

Train-Test Split

The Train-Test Split method divides the dataset into two parts: a training set and a test set. The model is trained on the training set and evaluated on the test set. This method is simple and computationally efficient, making it ideal for quick performance assessments. However, it is prone to result variation, as the performance depends on a single random split of the data. This can lead to overfitting or underfitting, particularly when the dataset is small or imbalanced (Wen et al., 2019). While convenient, the Train-Test Split approach may not fully capture the model's generalizability due to its limited evaluation scope.

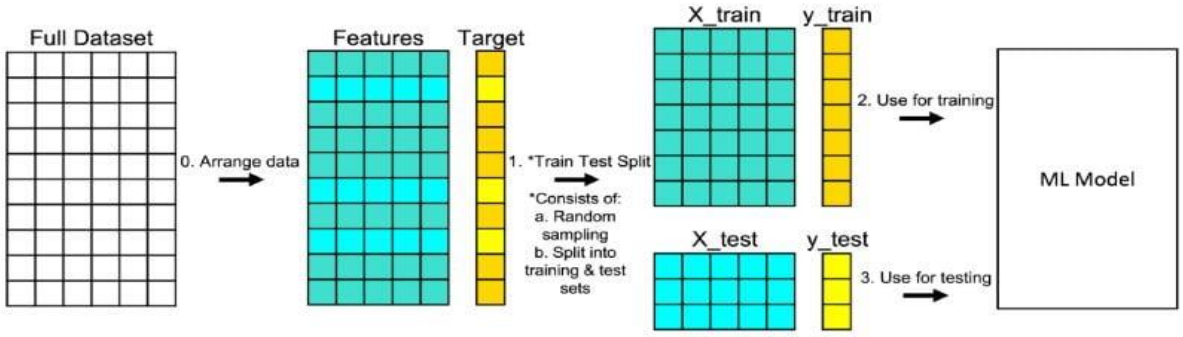


Figure 3: Representation of Train-Split Test

K-Fold Cross Validation

K-Fold Cross Validation divides the dataset into K equal parts, or folds. The model is trained and tested **K** times, with each fold acting as the test set once, while the remaining K-1 folds are used for training. This method provides a more comprehensive evaluation of the model's performance, as every data point is used for both training and testing. By reducing the risk of overfitting and providing a more robust estimate of model accuracy, K-Fold Cross Validation is especially useful in scenarios where a single Train-Test Split might yield biased results (Chicco & Jurman, 2020).

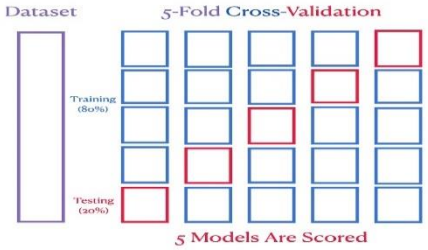


Figure 4: Representation of K-Fold Cross Validation

Stratified K-Fold Cross Validation

Stratified K-Fold Cross Validation is an enhancement of K-Fold Cross Validation that ensures each fold maintains the same class distribution as the entire dataset. This method is particularly beneficial when dealing with imbalanced datasets, where certain classes may be underrepresented. By preserving class proportions in each fold, Stratified K-Fold Cross Validation provides a more reliable evaluation of performance metrics, especially for Recall and F1-Score, which are sensitive to class imbalance (Gupta & Rani, 2021). It is widely regarded as a preferred method for classification tasks, where maintaining a balance of classes in the validation process is critical for accurate model assessment (Berrar, 2019).

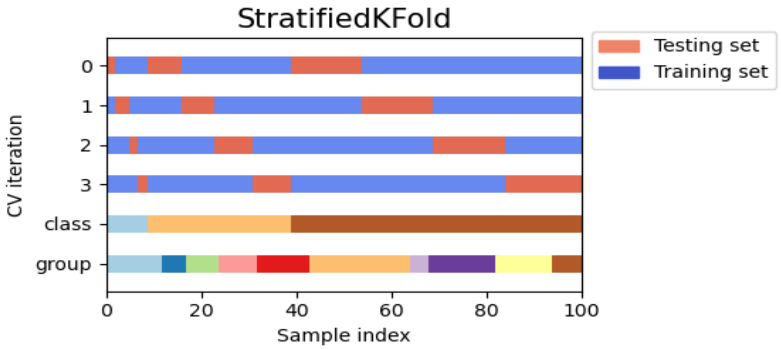


Figure 5: Representation of Stratified K-Fold Cross Validation

Evaluation Metrics

Accuracy

The concept of accuracy serves as a prevalent evaluation metric within the domain of machine learning (ML), functioning to assess the efficacy of a classification model in predicting the appropriate class label for input data. It systematically quantifies the proportion of accurately predicted instances relative to the overall number of instances present in the data set.

$$Accuracy = \frac{No.of\ Correct\ Prediction}{Total\ No.of\ Prediction} = \frac{TP + TN}{TP + TN + FP + FN}$$

In which, TP is the number of true positive outcomes. .FP is the number of false positive outcomes. TN is the number of true negative outcomes. FN is the number of false negative outcomes

Precision

Precision measures how correct the model's positive predictions are, specifically the ratio of true positives to the total predicted positives (true positives plus false positives). Precision is especially useful in situations where it is crucial that the model correctly identifies a positive class.

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall, often referred to as sensitivity or the true positive rate, serves as a metric for evaluating the model's proficiency in accurately recognizing all pertinent instances belonging to a specific class within the dataset. It quantitatively assesses recall as the ratio of true positives (accurately predicted positives) to the total number of actual positives.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score

The F1 metric integrates both precision and recall into a unified measure, thereby offering a comprehensive assessment of model efficacy. The F1 score is particularly advantageous when attempting to achieve an equilibrium between facilitating accurate positive classifications (high precision) and maximizing the identification of positive instances (high recall).

$$F1 - Score = \frac{2(Precision\ XRecall)}{Precision + Recall}$$

ROC-AUC

The ROC AUC, an abbreviation for Receiver Operating Characteristic Area Under the Curve, serves as a prevalent metric within the realm of machine learning for assessing the efficacy of binary classification models. This metric articulates a model's proficiency in differentiating between affirmative and negative classes across various probability thresholds.

Result and Discussion

This section presents a comprehensive evaluation of three prominent machine learning models: SVM**,** Naive Bayes, and Decision Tree, tested using three distinct validation techniques: Train-Test Split**,** K-Fold Cross Validation, and Stratified K-Fold Cross Validation. The performance of these models was assessed using key metrics such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC to measure their overall accuracy, sensitivity, and discriminatory power under different data partitioning scenarios..

Train-Test Split method was applied with three distinct random states (7, 21, and 42), with test size 0.2. The table 1 below summarizes the mean and standard deviation for each model across the different random states.

This is followed by the results tables for the analysis of the K-Fold and Stratified K-Fold Cross Validation methods, using different K values of 3, 5, and 10, as presented in Tables 2 and 3.

Table 1
Result Test-Train Split

Metric	SVM (Mean ±Std Dev)	Naive Bayes (Mean ±Std Dev)	Decision Tree (Mean ±Std Dev)
Accuracy	0.7124 ± 0.0564	0.7381 ± 0.0445	0.7041 ± 0.0421
Precision	0.63541 ± 0.0728	0.6697 ± 0.0454	0.5250 ± 0.1348
Recall	0.6275 ± 0.0191	0.5855 ± 0.1144	0.5618 ± 0.1328
F1-Score	0.5821 ± 0.0822	0.6202 ± 0.0785	0.5804 ± 0.0826
ROC-AUC	0.6930 ± 0.0605	0.7004 ± 0.0684	0.6940 ± 0.0615

Table 2
Result K-Fold Cross Validation

Model	K Value	Accuracy (Mean ±Std Dev)	Precision (Mean ±Std Dev)	Recall (Mean ±Std Dev)	F1-Score (Mean ±Std Dev)	ROC-AUC (Mean ±Std Dev)
SVM	3	0.7591 ± 0.0244	0.6966 ± 0.0616	0.5596 ± 0.0286	0.6190 ± 0.0289	0.8308 ± 0.0248
	5	0.7735 ± 0.0219	0.7267 ± 0.0446	0.5672 ± 0.0513	0.6354 ± 0.0379	0.8316 ± 0.0242
	10	0.7722 ± 0.0296	0.7268 ± 0.0572	0.5635 ± 0.0696	0.6316 ± 0.0544	0.8284 ± 0.0386
Naïve Bayes	3	0.7422 ± 0.0304	0.6425 ± 0.0484	0.5896 ± 0.0413	0.6149 ± 0.0446	0.8092 ± 0.0368
	5	0.7513 ± 0.0186	0.6621 ± 0.0239	0.5860 ± 0.0404	0.6214 ± 0.0320	0.8171 ± 0.0248
	10	0.7565 ± 0.0330	0.6743 ± 0.0599	0.5930 ± 0.0666	0.6288 ± 0.0508	0.8161 ± 0.0458
Decision Tree	3	0.6966 ± 0.0223	0.5453 ± 0.0238	0.5148 ± 0.0651	0.5635 ± 0.0254	0.6561 ± 0.0368
	5	0.7240 ± 0.0553	0.5764 ± 0.0764	0.5524 ± 0.0271	0.5818 ± 0.0634	0.6852 ± 0.0248
	10	0.7044 ± 0.0553	0.5950 ± 0.1134	0.5858 ± 0.1162	0.5746 ± 0.1070	0.6680 ± 0.0458

Table 3
Result Stratified K-Fold Cross Validation

Model	K Value	Accuracy (Mean ±Std Dev)	Precision (Mean ±Std Dev)	Recall (Mean ±Std Dev)	F1-Score (Mean ±Std Dev)	ROC-AUC (Mean ±Std Dev)
SVM	3	0.7591 ± 0.0244	0.6966 ± 0.0616	0.5596 ± 0.0286	0.6190 ± 0.0289	0.8308 ± 0.0248
	5	0.7735 ± 0.0219	0.7267 ± 0.0446	0.5672 ± 0.0513	0.6354 ± 0.0379	0.8316 ± 0.0242
	10	0.7722 ± 0.0296	0.7268 ± 0.0572	0.5635 ± 0.0696	0.6316 ± 0.0544	0.8284 ± 0.0386
Naïve Bayes	3	0.7422 ± 0.0304	0.6425 ± 0.0484	0.5896 ± 0.0413	0.6149 ± 0.0446	0.8092 ± 0.0368
	5	0.7513 ± 0.0186	0.6621 ± 0.0239	0.5860 ± 0.0404	0.6214 ± 0.0320	0.8171 ± 0.0248
	10	0.7565 ± 0.0330	0.6743 ± 0.0599	0.5930 ± 0.0666	0.6288 ± 0.0508	0.8161 ± 0.0458
Decision Tree	3	0.6889 ± 0.0176	0.5437 ± 0.0340	0.5560 ± 0.0580	0.5576 ± 0.0468	0.6664 ± 0.0171
	5	0.7071 ± 0.0237	0.6111 ± 0.0602	0.5748 ± 0.0507	0.5833 ± 0.0731	0.6888 ± 0.0448
	10	0.6978 ± 0.0761	0.5881 ± 0.0972	0.5708 ± 0.1023	0.5634 ± 0.0985	0.6654 ± 0.0730

The three tables above illustrate the performance comparison of three machine learning models, namely Decision Tree, SVM, and Naive Bayes, based on three different evaluation methods: Train-Test Split, K-Fold Cross Validation, and Stratified K-Fold Cross Validation. Each table presents the mean and standard deviation for key metrics such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC for each model and evaluation method.

The analysis results reveal performance differences between these methods for each model, as well as variation in the stability of the results as reflected by the standard deviation in the metrics.

For Train-Test Split, this approach is quick and simple. However, the results exhibit variability due to the random partitioning of the dataset. This variability is particularly noticeable in imbalanced datasets, where random splits may not accurately represent the overall population. The standard deviations in the metrics for Train-Test Split are generally higher compared to the cross-validation methods, indicating less stable performance across different splits. For example, SVM shows an accuracy of around 0.712 but with a relatively high standard deviation of 0.056, reflecting variability in performance. Naive Bayes achieves the highest accuracy at 0.738, but still with some variability (standard deviation of 0.044). Decision Tree shows the weakest performance with an accuracy of approximately 0.704 and a standard deviation of 0.042, indicating that its results fluctuate due to the random allocation of data in the Train-Test Split method. This variation in performance can be attributed to the dependence of Train-Test Split on the random allocation of data, and the higher standard deviation highlights its limited stability.

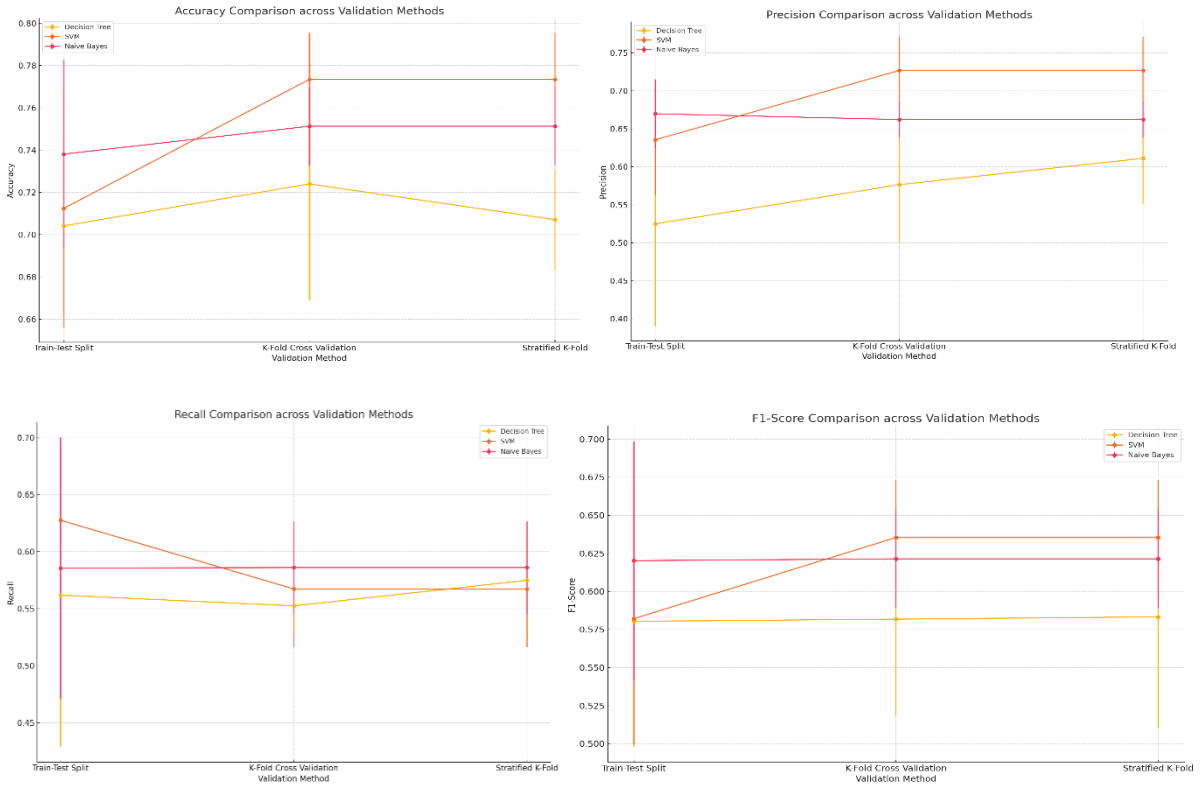
In K-Fold Cross Validation, the dataset is divided into several "folds," and the model's performance is evaluated by averaging the results across all folds. This method significantly reduces the variability observed in Train-Test Split and provides a more stable assessment of model performance, as evidenced by the lower standard deviations. In this analysis, SVM demonstrates consistent and superior performance, with the highest accuracy of around 0.773 and a strong F1-Score of 0.635, reflecting a good balance between precision and recall. The standard deviation of accuracy is much lower (0.021), showing greater stability in the model's performance across different folds. Naive Bayes also performs well, with an accuracy of approximately 0.751 and a standard deviation of 0.018, indicating consistent performance across different folds. Decision Tree, while showing lower performance compared to the other models, benefits from K-Fold Cross Validation's stability, with an accuracy of 0.724 and a standard deviation of 0.055, highlighting an improvement in stability compared to Train-Test Split.

Stratified K-Fold Cross Validation offers additional advantages by ensuring that each fold maintains a balanced class representation, making it more suitable for imbalanced datasets. This method not only improves the overall performance of the models but also enhances the stability of the results, as reflected by the standard deviations. For instance, SVM again leads in performance, achieving the highest ROC-AUC (0.8316) with a standard deviation of only 0.024, indicating highly stable performance. Naive Bayes follows closely with a ROC-AUC of 0.8171 and a standard deviation of 0.025, reflecting similarly stable results. Decision Tree, while showing lower performance with a ROC-AUC of around 0.6888, demonstrates

improvements in precision and recall compared to Train-Test Split, and its standard deviations are generally lower, indicating more consistent performance across folds in this method.

The comparison between these three methods indicates that SVM is the most effective model overall, particularly in K-Fold Cross Validation and Stratified K-Fold Cross Validation, where it achieves the highest scores in accuracy, precision, F1-Score, and ROC-AUC, with the lowest standard deviations, highlighting its stable performance. Naive Bayes offers balanced performance, excelling in recall, making it a good choice when the goal is to maximize positive class detection, and it also benefits from the reduced variability in results with lower standard deviations in K-Fold and Stratified K-Fold. Decision Tree, while the weakest in most metrics, especially in Train-Test Split, remains useful in scenarios where model interpretability is prioritized over high accuracy, and it also shows improvements in performance stability when using cross-validation techniques.

While the table provides a detailed numerical comparison, the following line graph offers a visual representation of the same data, allowing for easier interpretation of trends and performance stability across the different validation methods. The graph highlights the variations in Accuracy, Precision, Recall**,** F1-Score, and ROC-AUC for each model, with the error bars reflecting the standard deviation, offering a clearer understanding of the models' consistency and overall performance.



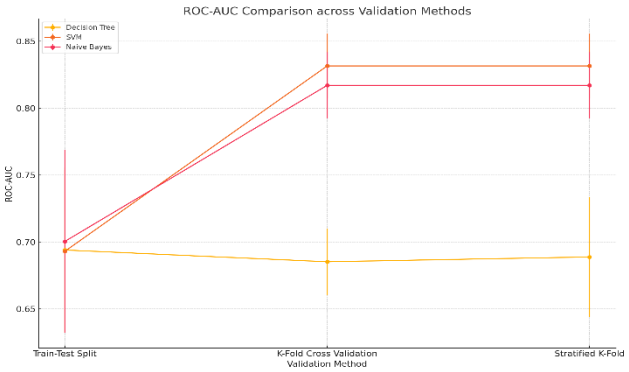


Figure 5:Representation of line graph

The performance analysis across different validation methods shows that SVM consistently outperforms other models, particularly in K-Fold and Stratified K-Fold Cross Validation, with an accuracy of 0.773 and low standard deviation, indicating stable results. Naive Bayes follows closely with an accuracy of 0.751, showing reliable performance, while Decision Tree lags behind with greater variability, especially in K-Fold Cross Validation.
For precision, SVM leads with values above 0.72 and low variability, while Naive Bayes shows stable precision **(**0.56), while Decision Tree fluctuates.
SVM also achieves the best balance in F1-Score (0.621), with Decision Tree showing improvement in Stratified K-Fold (~0.583). In ROC-AUC, SVM dominates with 0.831, while Naive Bayes follows closely (0.817). Decision Tree performs weaker, though it stabilizes in Stratified K-Fold.

Conclusion and Future Work

In conclusion, the analysis of model performance across both table and line graph presentations consistently demonstrates that SVM is the most reliable and high-performing model, particularly when combined with K-Fold and Stratified K-Fold Cross Validation. SVM not only achieves the highest accuracy but also maintains stability across all metrics, including Precision, Recall, F1-Score, and ROC-AUC, with minimal variability. This makes it the optimal choice for most scenarios, especially in handling imbalanced datasets where Stratified K-Fold Cross Validation plays a critical role in ensuring balanced representation and robust performance evaluation. Naive Bayes, while slightly behind SVM, provides balanced performance, excelling in Recall and ROC-AUC, making it a strong option for applications where identifying true positives is crucial. Decision Tree, though weaker overall, benefits substantially from Stratified K-Fold, which enhances its stability, particularly in Precision and Recall.

These findings underscore the critical importance of employing cross-validation techniques, particularly Stratified K-Fold Cross Validation, when dealing with imbalanced datasets to ensure more reliable and stable model evaluations. For future research, it is recommended to explore ensemble methods such as Random Forest and Gradient Boosting to further improve the performance and stability of models like Decision Tree. Additionally, the use of nested cross-validation for fine-tuning hyperparameters and data augmentation techniques like SMOTE should be considered to address class imbalance and enhance overall model accuracy and robustness. Expanding these strategies could lead to significant advancements in model reliability, particularly for complex and imbalanced datasets.

References

Abubakar, A. I., Waziri, S. A., & Uba, M. (2021). Application of naive Bayes classifier in medical diagnosis using optimized features. *Mathematics*, 9(4), 419. https://doi.org/10.3390/math9040419

Al-Sideiri, A., Che Cob, Z., & Drus, S. (2019, December 14). Machine Learning Algorithms for Diabetes Prediction: A Review Paper.*International Conference on Artificial Intelligence*. https://doi.org/10.1145/3388218.3388231

Bereda, G**.** (2022). A Review of the Hybrid Description of Diabetes Mellitus. *BOHR International Journal of Current Research in Diabetes and Preventive Medicine*.

Berrar, D. (2019). Cross-validation. In *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics* (pp. 542-545). Elsevier. https://doi.org/10.1016/B978-0-12-809633-8.20349-X

Brnabic, A., & Hess, L. M. (2021). Systematic literature review of machine learning methods used in the analysis of real-world data for patient-provider decision making.*BMC Medical Informatics and Decision Making*. https://doi.org/10.1186/S12911-021-01403-2

Brownlee, J. (2019). Repeated K-Fold Cross-Validation for Model Evaluation in Python. *Machine Learning Mastery*. https://machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/

Catania, C., Guerra, J. J., Romero, J. M., Caffaratti, G. D., & Marchetta, M. G. (2022). Beyond Random Split for Assessing Statistical Model Performance.*arXiv.Org*. https://doi.org/10.48550/arXiv.2209.03346

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2021). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research, 16*(1), 321–357. https://doi.org/10.1613/jair.953

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

Ebubeogu, A. F., & Lee, S. P. (2019). Systematic literature review of preprocessing techniques for imbalanced data.*IET Software*. https://doi.org/10.1049/IET-SEN.2018.5193

Fu, G. H., Wu, Y. J., Zong, M. J., & Pan, J. (2020). Hellinger distance-based stable sparse feature selection for high-dimensional class-imbalanced data. *BMC bioinformatics*, *21*, 1-14.

Goot, R. (2021, November 1). We Need to Talk About train-dev-test Splits.*Empirical Methods in Natural Language Processing*.

Gupta, S., & Rani, R. (2021). An empirical evaluation of ensemble techniques for imbalanced learning problems. *International Journal of Machine Learning and Cybernetics, 12*(1), 221-237. https://doi.org/10.1007/s13042-020-01169-8

Hakimi, H., Kamalrudin, M., & Abdullah, R. S. (2023). Software Security Readiness Model For Remote Working In Malaysian Public Sectors: Conceptual Framework. Journal Of Theoretical And Applied Information Technology, 101(8).

Haq, A., Li, J., Khan, J., Memon, M. H., Nazir, S., Ahmad, S., Khan, G., & Ali, A. (2020). Intelligent machine learning approach for effective recognition of diabetes in e-healthcare using clinical data. *Sensors (Basel, Switzerland)*

Husain, A., & Khan, M. H. (2018). Early diabetes prediction using voting based ensemble learning. In *Advances in Science, Technology and Innovation* (pp. 95-103).

Khoshgoftaar, T. M., & Gao, K. (2021). Performance metrics for classification problems involving imbalanced datasets. *IEEE Transactions on Knowledge and Data Engineering, 33*(4), 1544–1559. https://doi.org/10.1109/TKDE.2020.2987069

Li, C., Zheng, H., & He, Z. (2018). Improving model performance on imbalanced datasets with stratified k-fold cross-validation. *IEEE Access, 6*, 20368–20375. https://doi.org/10.1109/ACCESS.2018.2816348

Perez, J., Díaz, J., Garcia-Martin, J., & Tabuenca, B. (2020). Systematic Literature Reviews in Software Engineering -- Enhancement of the Study Selection Process using Cohen's Kappa Statistic.*arXiv: Software Engineering*. https://doi.org/10.1016/J.JSS.2020.110657

Purwanto, A. D., Wikantika, K., Deliar, A., & Darmawan, S. (2023). Decision Tree and Random Forest Classification Algorithms for Mangrove Forest Mapping in Sembilang National Park, Indonesia. *Remote Sensing*, 15(1), 16. https://doi.org/10.3390/rs15010016

Rekha, G., Tyagi, A. K., & Reddy, V. K. (2019). A Wide Scale Classification of Class Imbalance Problem and its Solutions: A Systematic Literature Review.*Journal of Computer Science*. https://doi.org/10.3844/JCSSP.2019.886.929

Rampisela, T. V., & Rustam, Z. (2018). Classification of schizophrenia data using Support Vector Machine (SVM). *Journal of Physics: Conference Series, 1108*, 012044

Subramani, N., Easwaramoorthy, S. V., Mohan, P., Subramanian, M., & Sambath, V. (2023). A Gradient Boosted Decision Tree-Based Influencer Prediction in Social Network Analysis. *Big Data and Cognitive Computing*, 7(1), 6. https://doi.org/10.3390/bdcc7010006

Suhaimin, K. N., Mahmood, W. H. W., Ebrahim, Z., Hakimi, H., & Aziz, S. (2023). Human Centric Approach in Smart Remanufacturing for End-Life-Vehicle (ELV)'s Stabilizer Bar. Malaysian Journal on Composites Science and Manufacturing, 12(1), 1-12.

Wen, Y., Yang, B., Song, S., & Yu, W. (2019). A comprehensive comparison of data splitting algorithms for machine learning. *IEEE Access, 7*, 125503-125521. https://doi.org/10.1109/ACCESS.2019.2939695

Yang, T., Zhang, L., Yi, L., Feng, H., Li, S., Chen, H., Zhu, J., Zhao, J., Zeng, Y., & Liu, H. (2020). Ensemble learning models based on noninvasive features for type 2 diabetes screening: Model development and validation. *JMIR Medical Informatics*, 8.

Zhang, Y., Yang, L., & Zhu, X. (2020). A survey on cross-validation techniques and methods. *Journal of Machine Learning Research, 21*(1), 146–175. https://doi.org/10.1109/JMLR.2020.321876