Journal of Research in Artificial Intelligence for Systems and Applications (RAISA
2025, Vol. 01, No. 01, E-ISSN: 3090-3157

Original Research

Comparative Analysis of Machine Learning Algorithms with Sequential Feature Selection and Gridsearch Optimization (Hepatitis Case Study)

Tasya Oktaviana Dwi Cahyanti[1], Nelly Oktavia Adiwijaya[1,*], Gama Wisnu Fajarianto[1]

[1]Departement of Informatics, Faculty of Computer Science, University of Jember

Abstract

Hepatitis is a dangerous disease that can cause liver damage. It is often difficult to diagnose because the symptoms of different categories of hepatitis are hard to distinguish. Hepatitis that is not promptly and properly managed, especially in individuals with chronic liver conditions, can lead to complications. This study aims to improve model performance in hepatitis diagnosis using medical data from hepatitis A and B patients at Citra Husada Hospital in Jember. The results show that the method of gridsearch tuning with SMOTE and SFS is very effective in enhancing the performance of the random forest and extra tree algorithms in managing hepatitis symptom data. The best performance was achieved by the random forest algorithm with an 80:20 data ratio, reaching the highest accuracy of 94.87%, recall of 95.96%, precision of 93.33%, f1-score of 94.07%, specificity of 97.97%, and ROC AUC of 99.13%.

Keywords

Hepatitis, Random Forest, Extra Tree; SFS, GridSearch, SMOTE

1. Introduction

Hepatitis is a dangerous disease caused by liver injury, characterized by inflammation of the liver and damage to hepatocytes. Inflammation can lead to the death of liver cells and affect liver function. Hepatitis is caused by a viral infection, and these viruses are categorized into hepatitis A, hepatitis B, hepatitis C, hepatitis D, hepatitis E, and hepatitis G [1], [2], [3]. According to data from the Jember Health Office, cases of hepatitis A in Jember increased to 217 cases at the end of 2019. Meanwhile, hepatitis B cases, based on data from the Ministry of Health, are predominantly transmitted from mother to child in Indonesia. East Java was recorded as the province with the highest national cases of pregnant women testing positive for hepatitis B, with a total of 8,269 cases in 2022.

The clinical symptoms of different types of hepatitis cannot be distinguished from one another. The most common symptoms include fever, jaundice (yellowing of the eyes and skin), weakness, fatigue, nausea, vomiting, abdominal pain, arthralgia (joint pain), myalgia (muscle pain), diarrhea, anorexia (eating disorder or loss of appetite), and dark-colored urine. Not everyone infected will experience all of these symptoms [4], [5]. This can complicate diagnosis and requires further examination. Hepatitis that is not promptly and properly treated, especially in individuals with chronic liver disorders, can lead

[*]Corresponding author: Nelly Oktavia Adiwijaya

Email addresses:

tasyaoktaviana5885@gmail.com (Tasya Oktaviana Dwi Cahyanti), nelly.oa@unej.ac.id (Nelly Oktavia Adiwijaya), gamawisnuf@unej.ac.id (Gama Wisnu Fajarianto)

Received: 30 12 2024; **Accepted:** 03 01 2025; **Published:** 01 05 2025

to complications such as liver failure, cirrhosis, fulminant hepatitis, or liver cancer (hepatocellular carcinoma) [6]. However, diagnosing hepatitis involves the clinical interpretation of various symptoms, risk factors, laboratory tests, and vital signs, which is a challenging task. This task becomes even more complex if the available data is unclear. This can slow down the decision-making process for doctors, even those with extensive experience. Human diagnosis is not entirely accurate as humans are prone to errors [1].

Therefore, to process diverse data, avoid clinical bias, and reduce evaluation time, this study proposes the application of machine learning. This study will process symptom data using machine learning for hepatitis A and hepatitis B. Hepatitis A is an inflammation of the liver caused by the hepatitis A virus (HAV) [7]. Hepatitis B is an infectious disease caused by liver inflammation due to the hepatitis B virus [8]. Machine learning is a branch of artificial intelligence that enables systems to learn from experience, allowing them to make predictions or decisions without being explicitly programmed [9]. Machine learning is divided into four categories: supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning [10]. To optimize model performance, this study uses sequential feature selection and gridsearch, with the implementation of SMOTE for data balancing. Sequential feature selection allows researchers to choose the most relevant and significant features in processing hepatitis symptom data [11]. Additionally, by using gridsearch, researchers can find the best parameter combinations for the Random Forest and Extra Tree algorithms [12]. The study will compare the Extra Tree and Random Forest algorithms by applying sequential feature selection before and after gridsearch optimization, with and without SMOTE. This comparison will help in selecting the most suitable model for processing hepatitis symptom data and ensuring optimal performance in machine learning.

This research aims to analyze and evaluate the performance comparison of random forest and extra tree machine learning optimization models using sequential feature selection and gridsearch with the application of SMOTE to determine the most suitable model for managing hepatitis disease symptom data to obtain high accuracy. The technique used is data processing using clinical data. This research also aims to help in managing hepatitis disease symptom data to detect and diagnose hepatitis disease more efficiently by applying machine learning models.

2. Research Method (14 Pt)

This research aims to compare and evaluate the performance of optimized random forest and extra trees models using sequential feature selection and gridsearch, along with applying SMOTE to address data imbalance. The objective is to find the most suitable model for analyzing hepatitis symptom data to achieve high accuracy. Furthermore, this research is expected to aid in the early detection and more efficient diagnosis of hepatitis using clinical data. A flowchart illustrating the proposed methodology is presented in Figure 1.

2.1. Data Description

The data used in this study were medical records of hepatitis A and hepatitis B patients obtained from Citra Husada Jember Hospital. The dataset consists of 130 features and 1 target column used to determine whether an individual is infected with hepatitis A or hepatitis B, or neither. There are 156 data points in total, with 65 data points from patients infected with hepatitis A, 51 data points from patients infected with hepatitis B, and 40 data points from uninfected patients.

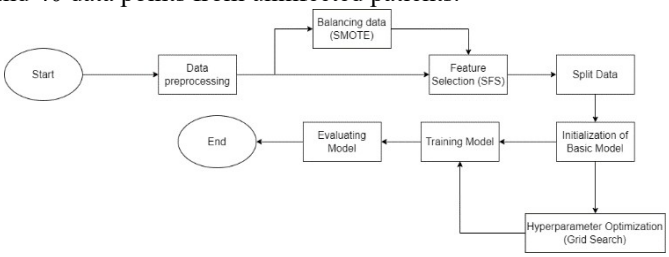

Figure 1. Flowchart illustrating the proposed methodology

2.2 Data Preprocessing

Data preprocessing is the initial phase in data processing. It involves transforming raw data, which is often inconsistent and in an unsuitable format, into usable and processable data. In this research, the data preprocessing steps include feature (column) removal, duplicate data checking, duplicate data removal, missing value checking, data analysis and visualization, missing value handling, data encoding, data transformation, outlier detection, dataset splitting, and data standardization.

2.3 Balancing Data (SMOTE)

SMOTE is a technique designed to address class imbalance in datasets. It generates synthetic samples for the minority class by interpolating feature values between an instance and its nearest neighbours. Data balancing is a process employed to address imbalanced datasets. Given the imbalance in the distribution of hepatitis A, hepatitis B, and non-hepatitis cases in our dataset, the SMOTE technique was applied to generate synthetic data for the minority class. This study compares the performance of models trained with and without SMOTE. The newly generated synthetic samples, as depicted in Equation 1 [15], were produced using the following formula (1).

$$z = x + \lambda.(y - x) \quad (1)$$

2.4. Feature Selection (SFS)

Sequential feature selection (SFS) is a method for selecting the most relevant features from a dataset. SFS builds models incrementally, starting with no features and adding one feature at a time that contributes the most to improving classification performance on the training data. This process is repeated until an optimal number of features is reached [11], [13], [16]. Feature selection is a process of selecting a subset of features from a larger set of features in a dataset. The primary objectives of feature selection are to reduce feature redundancy to minimize noise in the model, prevent overfitting, enhance model performance and interpretability, and identify the most influential features for model performance. In this study, we employed sequential feature selection.

2.5. Split Data

In the data splitting stage, the dataset was divided into two parts: training data and testing data. For k-fold cross-validation, the dataset was further divided into three parts: training data, testing data, and validation data. The training and testing data splits used were 90:10, 80:20, 70:30, and 60:40. Additionally, k-fold cross-validation was performed with k values of 3, 5, 7, and 10.

2.6. Initialization of Basic Model

This study, two algorithm models were used random forest and extra trees.

2.6.1 Random Forest

Random forest is a supervised learning algorithm that constructs an ensemble of decision trees to make more accurate predictions. In random forest, each decision tree produces a prediction, but the final prediction of the model is determined by a majority vote among all decision trees. The formula used in the prediction process using the random forest method can be seen in Equation 2 [13], [17].

$$I(x)\ \arg max_c(\sum_{n=1}^{N} I_{h_n(X)=c} I_{h_n(X)=c}) \quad (2)$$

2.6.2 Extra Trees

The extra trees classifier is an extension of the Random Forest algorithm with some variations. Similar to random forest, all base learners are decision trees. However, there are some differences in how the data is split. Extra trees classifier splits the data randomly without replacement, unlike random forest which uses the best split approach. [18], [19]. The formula for building a tree in extra trees [20]. The first one, calculating entropy, can be seen in equation 3.

$$Entropy\ (S1) = -\sum_{i=1}^{c1} p_i\ log_2\ p_i \quad (3)$$

Subsequently, the calculation of information gain can be seen in equation 4.

$$Gain\ (S1, A) = Entropy(S1) - \sum_{\$\epsilon\ Values\ (A)} \frac{|S1_u|}{|S1|}\ Entropy\ (S1_u) \quad (4)$$

2.7. Hyperparameter Optimization (Gridsearch)

Gridsearch is a technique for finding the optimal combination of hyperparameters using a specified range, upper bound, and number of steps. Gridsearch systematically explores all possible combinations within the defined grid, evaluating each to determine the best performing configuration [21]. At this stage, Gridsearch was employed to optimize the hyperparameters of both the random forest and extra trees models, aiming to enhance model performance. Cross-validation was utilized to evaluate the models' performance.

2.8. Training Model

At this stage, training was conducted on each machine learning model. These models were categorized into four groups: models with SMOTE and SFS, models with SFS only, and both groups before and after hyperparameter optimization using gridsearch.

2.9. Evaluating Model

The model evaluation process involves measuring the performance of a machine learning model by applying SMOTE and SFS feature selection, as well as a machine learning model with SFS feature selection without applying SMOTE before gridsearch optimization and a machine learning model that applies SMOTE and SFS feature selection, and a machine learning model with SFS feature selection without applying SMOTE after gridsearch optimization. Then, it is evaluated using a confusion matrix by calculating recall or sensitivity, specificity, precision, accuracy, F1-score, and ROC-AUC.

3.1.1 Recall or Sensitivity

Recall or sensitivity is a metric used to assess a model's ability to correctly identify positive cases. The formula for recall or sensitivity can be found in Equation 5 [22], [23], [24].

$$Recall = Sensitivity = \frac{(TP)}{(TP+FN)} \quad (5)$$

3.1.2 Specificity

Specificity is a metric used to assess a model's ability to correctly identify negative cases. The formula for specificity can be found in Equation 6 [22].

$$Specificity = \frac{(TN)}{(TN+FP)} \quad (6)$$

3.1.3 Precision

Precision is a metric used to assess a model's ability to correctly predict positive cases among all predicted positive cases. The formula for precision can be found in Equation 7 [22], [24].

$$Precision = \frac{(TP)}{(TP+FP)} \quad (7)$$

3.1.4 Accuracy

Accuracy is a metric used to evaluate how well a classifier predicts the true condition by considering both correct and incorrect predictions. The formula for accuracy can be found in Equation 8 [22], [24].

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FN+FP)} \quad (8)$$

3.1.5 F1-Score

The F1-score is the harmonic mean of precision and recall. The formula for the F1-score can be found in Equation 9 [22], [24], [25].

$$F1-score = \frac{2*(recall*precision)}{(recall+precision)} \quad (9)$$

3.1.6 ROC AUC

ROC-AUC is used to evaluate an algorithm's ability to generalize to new data. The formulas for ROC-AUC can be found in Equations 10 and 11.

$$TPR = \frac{(TP)}{(TP+FN)} \quad (10)$$

$$FPR = \frac{(FP)}{(TN+FP)} \quad (11)$$

3. Results and Analysis

3.1. Result Data Preprocessing

To facilitate data management and analysis, several libraries were employed in this study: pandas, matplotlib, and NumPy. Pandas was used for data analysis, manipulation, and cleaning. Matplotlib was utilized for data visualization. NumPy was employed for performing mathematical operations on arrays.

3.1.1 Feature Removal

Feature removal is conducted for unused features. The drop() method is used to remove features in the data frame that are not relevant to this research. The study focuses on symptom data for managing hepatitis disease data. Features that are removed include medical record number, admission date, discharge date, date of birth, primary ICD 10 diagnosis code, secondary ICD 10 diagnosis code, and laboratory results. Additionally, other removed features such as chief complaint, current medical history, and past medical history are excluded because these features have been separated based on the data they contain. After feature removal, the number of features becomes 68.

3.1.2 Duplicated Data Checking

Duplicate data checking is the process of identifying and removing identical rows or entries from the dataset. Duplicate data checking is performed using the duplicated() method, and the sum() method is used to count the total number of rows identified as duplicates. In this study, there are no duplicate data.

3.1.3 Missing Value Checking

Missing value checking is conducted to identify where data is missing or has null values. Missing value checking is performed using the isnull() method, and the sum() method is used to count the number of missing values per column in the dataset.

3.1.4 Data Analysis and Visualization

Before handling missing values, data analysis and visualization are performed to understand the characteristics of data distribution, whether the data is symmetric or asymmetric. This information is used to address the missing value issues in the data appropriately. From the analysis and visualization results, it is found that the average data is asymmetric.

3.1.5 Overcoming missing value

Handling or filling missing values in each column is done using the mode value of each column. Since the results of the column analysis and visualization show that the average data is asymmetric, using the mode method mode() to address missing values can be the best choice. If missing values are handled by deleting columns, it can lead to the loss of valuable information that might be present in the deleted columns (features), as well as the loss of important columns (features), which can decrease the model's performance. Therefore, addressing missing values using the mode value is the most effective way because it retains all original features in the dataset and can also improve model performance, allowing the model to learn from as much information as possible.

3.1.6 Data Encoding

Data encoding is performed using the one-hot encoding method on categorical columns, except for the diagnosis column. The diagnosis column is excluded because it is the target variable or the variable to be predicted in the analysis or model to be built. After performing one-hot encoding, the number of features increased to 95. This occurs because, in the one-hot encoding process, each categorical feature is transformed into a new independent column (feature) in binary form (0 or 1).

3.1.7 Data Transformation

Data transformation on the 'Diagnosis' column is used to convert categorical values such as 'Non Hepatitis,' 'Hepatitis A,' and 'Hepatitis B' into numerical representations ('Non Hepatitis': 0, 'Hepatitis A': 1, 'Hepatitis B': 2) that are easier for machine learning algorithms to understand. By performing this transformation, the data becomes more ready to be used in model development.

3.1.8 Outlier Checking

Outlier checking is used to identify values that are significantly different from the majority of the data in the dataset to maintain data quality and improve the performance of the machine learning model. Outliers can affect descriptive statistics and cause overfitting. This outlier checking uses the quantile() method because it is simple, easy to compute, ignores the influence of outliers in the calculations, is effective in identifying extreme values, is flexible in determining outlier thresholds, and is suitable for various types of diverse data. In this study, no handling of outlier data was performed because these outlier values represent legitimate or authentic values in the dataset. Removing or adjusting these values could eliminate important information relevant to the characteristics being studied.

3.1.9 Dataset Division

The dataset is divided into 'x' (features) which includes all features except the 'Diagnosis' feature, and 'y' (target) which contains only the 'Diagnosis' feature. With this division, the number of features becomes 94. This dataset splitting is crucial for preparing the data before the machine learning model training process. By separating the dataset into 'x' and 'y,' it allows for more effective data management and analysis in the context of model development and evaluation.

3.1.10 Data Standardization

Data standardization is the process of transforming numerical data to adjust its distribution so that it has a mean of zero and a variance of one. In data processing for machine learning, data standardization is performed to ensure that all features have a similar scale. In this study, the StandardScaler() method imported from the scikit-learn library is used for standardization.

3.2. Result Balancing Data

SMOTE was employed to address the class imbalance issue in the dataset, as the number of samples in each class was disproportionate: 65 for Hepatitis A, 51 for Hepatitis B, and 40 for non-hepatitis. SMOTE helped to augment the minority class by creating new synthetic samples. Data balancing was achieved by importing the imblearn.over_sampling library.

3.3. Result Feature Selection

In this study, feature selection is performed using Sequential Feature Selection (SFS) because, despite the removal of unused features during the initial data preprocessing stage, the number of features is still large, at 94 features. Therefore, SFS is used to select or identify the most important or relevant features. Feature selection is done by importing SequentialFeatureSelection and setting k-features=31 for data with SMOTE applied, and k-features=32 for data without SMOTE. The choice of k-features values is determined by applying feature importance and threshold-based feature search.

3.4. Result Split Data

The dataset was partitioned using both train-test split and k-fold cross-validation. For the train-test split, various ratios were explored, including 60:40, 70:30, 80:20, and 90:10. Additionally, k-fold cross-validation was conducted with k values of 3, 5, 7, and 10. Table 4.4 presents the results of these data partitioning methods for both the SMOTE-oversampled and original datasets. The results of data partitioning, both with and without the application of SMOTE, can be found in Table 1.

Table 1. Dataset split ratio

Split Data - *SMOTE*	Data *Training*	Data *Testing*	Data Validation	Split Data Tanpa *SMOTE*	Data *Training*	Data *Testing*	Data Validation
60:40	117	78	-	60:40	93	62	-
70:30	136	59	-	70:30	108	47	-
80:20	156	39	-	80:20	124	31	-
90:10	175	20	-	90:10	139	16	-
K=3	117	45	13	K=3	93	51	11
K=5	140	39	16	K=5	111	31	13
K=7	151	27	17	K=7	119	22	14
K=10	158	19	18	K=10	126	15	14

3.5. Result Initialization of Basic Model

The baseline models in this study were initialized using two algorithms: random forest and extra trees. Several approaches were explored in this initialization phase. These include random forest and extra trees models with SMOTE and SFS feature selection, and random forest and extra trees models with SFS feature selection but without SMOTE. Both random forest and extra trees algorithms were imported from the scikit-learn ensemble library.

3.6. Result Hyperparameter Optimization

Hyperparameter optimization using gridsearch was conducted on both random forest and extra trees algorithms. By experimenting with predefined combinations of hyperparameters, the optimal configuration for each model was determined to maximize performance. Gridsearch optimization was applied to models with and without SMOTE and SFS feature selection. The GridSearchCV function from the scikit-learn model_selection library was utilized for this purpose. The specific hyperparameters and their corresponding ranges are presented in Table 2.

Table 2. Hyperparameter optimization

Hyperparameter		
Algorithm	*Hyperparameter*	Rentang Nilai
Random Forest	*N_estimators*	100,150
	Max_features	*Sqrt, log2*
	Max_depth	10, 15
	Min_samples_split	2, 4
	Min_samples_leaf	1, 2
	Bootstrap	*True, False*
	Criterion	*Gini, Entropy*
Extra Tree	*N_estimator*	100, 150
	Max_depth	10, 15
	Min_samples_split	2, 5
	Bootstrap	*False*

3.7. Result Training Model

In the model training stage, all the models that have been created will be trained using the pre-determined training data. Each model will use the training data to identify and learn patterns and relationships within the dataset. The goal at this stage is to prepare and train the models so that they can make accurate decisions on new or test data.

3.8. Result Evaluating Model

Machine learning models with SMOTE and Sequential Feature Selection (SFS) before and after GridSearch tuning show that GridSearch tuning significantly improves performance, with the highest performance achieved by the Random Forest model tuned with GridSearch at an 80:20 ratio, resulting in an accuracy of 94.87%, recall of 95.69%, precision of 93.33%, F1-score of 94.07%, specificity of 97.97%, and ROC AUC of 99.13%. This demonstrates that SMOTE and SFS help address data imbalance and select the most relevant features. Thus, the use of GridSearch tuning, SMOTE, and SFS can optimize model performance.

Handling missing values with the mode produces the best model performance compared to handling missing values by deletion. For a more comprehensive evaluation of the model, please refer to Table 3 for comparison of machine learning algorithms with SMOTE and SFS feature slection before and after tuning gridsearch with mode value, Table 4 to comparison of machine algorithms with SFS feature selection without SMOTE before and after tuning gridSearch with mode values, Table 5 to comparison of machine learnibg algorithms with SMOTE and SFS feature selection before and after gridsearch tuning with missing value removal, and Table 6 to comparison of machine learning algorithms with SFS feature selection without SMOTE before and after gridsearch tuning with missing value removal.

Table 3. Comparison of ML Algorithms with SMOTE and SFS Before and After Tuning Gridsearch with Mode Value

with SMOTE and SFS Before				
Model	*RF*	*ET*	*RF Tuning*	*ET Tuning*
Accuracy	87.17	84.61	94.87	87.17
Recall	89.02	85.00	95.69	87.22
Precision	86.80	83.09	93.33	85.49
F1-Score	86.12	83.24	94.07	85.55
Specificity	94.80	93.46	97.97	94.85
ROC AUC	98.34	98.05	99.13	98.01
After Tuning Gridsearch With Mode Value (K=7)				
Model	*RF*	*ET*	*RF Tuning*	*ET Tuning*
Accuracy	88.69	90.26	90.24	91.79
Recall	89.27	91.04	91.31	92.82
Precision	88.92	90.33	89.97	91.53
F1-Score	87.86	89.54	89.66	91.18
Specificity	94.50	96.06	95.29	96.06
ROC AUC	96.94	98.03	97.45	98.20

Table 4. Comparison of ML with SFS without SMOTE Before and After Tuning Gridsearch with Mode Values

with SFS without SMOTE Before				
Model	*RF*	*ET*	*RF Tuning*	*ET Tuning*
Accuracy	77.41	77.41	77.41	80.64
Recall	78.90	78.90	78.90	81.28
Precision	78.15	78.91	78.15	80.93
F1-Score	77.65	77.45	77.65	80.21
Specificity	90.10	90.74	90.10	91.76
ROC AUC	92.52	92.38	93.16	93.40
After Tuning Gridsearch with Mode Values (K=7)				
Model	*RF*	*ET*	*RF Tuning*	*ET Tuning*
Accuracy	85.77	83.17	86.42	86.42
Recall	86.13	83.46	86.81	86.18
Precision	86.21	84.09	86.91	87.08
F1-Score	85.05	82.68	85.86	85.66
Specificity	92.73	92.93	93.07	92.93
ROC AUC	94.37	94.45	94.74	94.50

Table 5. Comparison of ML Algorithms with SMOTE and SFS Before and After GridSearch Tuning with Missing Value Removal

Model	*RF*	*ET*	*RF Tuning*	*ET Tuning*
Accuracy	87.17	84.61	89.74	84.61
Recall	85.41	83.33	87.50	83.33
Precision	84.96	82.74	87.74	82.74
F1-Score	84.66	82.04	87.36	82.04
Specificity	94.49	93.62	95.41	93.62
ROC AUC	96.16	93.61	95.44	94.89

Table 6. Comparison of ML Algorithms with SFS without SMOTE Before and After GridSearch Tuning with Missing Value Removal

Model	*RF*	*ET*	*RF Tuning*	*ET Tuning*
Accuracy	77.41	77.41	77.41	80.64
Recall	77.57	75.33	75.79	78.17
Precision	78.57	76.66	75.76	80.65
F1-Score	76.53	75.87	75.38	78.88
Specificity	90.74	89.70	90.43	90.88
ROC AUC	95.12	92.31	95.01	95.13

4. Conclusion

The following are the conclusions of the research result:

1. The performance results of both algorithms, extra trees and random forest, with SMOTE and SFS before gridsearch optimization showed good performance, but the extra trees algorithm with SMOTE and SFS performed slightly better, with the best performance at k=7. After gridsearch optimization, the performance of both random forest and extra trees algorithms improved. Overall, the optimized random forest model with SMOTE and SFS achieved the best performance at an 80:20 ratio. This indicates that the gridsearch tuning method with SMOTE and SFS is very effective in improving model performance in handling hepatitis disease symptom data.
2. The performance results of both machine learning models with SFS without SMOTE before and after gridsearch tuning show that gridsearch tuning can improve performance results at various data ratios, although in some cases there is a decrease or stagnation in performance after gridsearch tuning. The best results before and after gridsearch optimization were found in both models at a 90:10 ratio. Overall, without SMOTE, the performance is quite good, but not as good as with SMOTE.

References

[1] A. Singh, J. C. Mehta, D. Anand, P. Nath, B. Pandey, and A. Khamparia, "An intelligent hybrid approach for hepatitis disease diagnosis: Combining enhanced k-means clustering and improved ensemble learning," in Expert Systems, Blackwell Publishing Ltd, Jan. 2021. doi: 10.1111/exsy.12526.

[2] M. T. M. Shata, H. F. Hetta, Y. Sharma, and K. E. Sherman, "Viral hepatitis in pregnancy," Oct. 01, 2022, John Wiley and Sons Inc. doi: 10.1111/jvh.13725.

[3] M. M. Majzoobi, S. Namdar, R. Najafi-Vosough, A. A. Hajilooi, and H. Mahjub, "Prediction of Hepatitis disease using ensemble learning methods," J Prev Med Hyg, vol. 63, no. 3, pp. E424–E428, 2022, doi: 10.15167/2421-4248/jpmh2022.63.3.2515.

[4] D. Castaneda, A. J. Gonzalez, M. Alomari, K. Tandon, and X. B. Zervos, "From hepatitis A to E: A critical review of viral hepatitis," Apr. 28, 2021, Baishideng Publishing Group Co. doi: 10.3748/wjg.v27.i16.1691.

[5] M. Wang and Z. Feng, "Mechanisms of hepatocellular injury in hepatitis A," 2021, MDPI AG. doi:

10.3390/v13050861.

[6] O. Gholizadeh et al., "Hepatitis A: Viral Structure, Classification, Life Cycle, Clinical Symptoms, Diagnosis Error, and Vaccination," 2023, Hindawi Limited. doi: 10.1155/2023/4263309.

[7] S. Hammond, "Hepatitis A," Workplace Health Saf, vol. 69, no. 3, p. 142, Mar. 2021, doi: 10.1177/2165079920988687.

[8] A. Din, Y. Li, and Q. Liu, "Viral dynamics and control of hepatitis B virus (HBV) using an epidemic model," Alexandria Engineering Journal, vol. 59, no. 2, pp. 667–679, Mar. 2020, doi: 10.1016/j.aej.2020.01.034.

[9] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. K. A. A. Khan, "Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review," in Procedia Computer Science, Elsevier B.V., 2020, pp. 1251–1260. doi: 10.1016/j.procs.2020.04.133.

[10] M. E. Febrian, F. X. Ferdinan, G. P. Sendani, K. M. Suryanigrum, and R. Yunanda, "Diabetes prediction using supervised machine learning," in Procedia Computer Science, Elsevier B.V., 2022, pp. 21–30. doi: 10.1016/j.procs.2022.12.107.

[11] A. Alotaibi et al., "Explainable Ensemble-Based Machine Learning Models for Detecting the Presence of Cirrhosis in Hepatitis C Patients," Computation, vol. 11, no. 6, Jun. 2023, doi: 10.3390/computation11060104.

[12] M. Ogunsanya, J. Isichei, and S. Desai, "Manufacturing Letters Grid Search Hyperparameter Tuning in Additive Manufacturing Processes-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0) Peer-review under responsibility of the Scientific Committee of the NAMRI/SME," 2023, [Online]. Available: www.sciencedirect.com

[13] H. Mamdouh Farghaly, M. Y. Shams, and T. Abd El-Hafeez, "Hepatitis C Virus prediction based on machine learning framework: a real-world case study in Egypt," Knowl Inf Syst, vol. 65, no. 6, pp. 2595–2617, Jun. 2023, doi: 10.1007/s10115-023-01851-4.

[14] S. Keputusan Direktur Jenderal Pendidikan Tinggi, dan Teknologi Nomor, N. Sharfina, and N. Ghaniaviyanto Ramadhan, "Terakreditasi SINTA Peringkat 3 Analisis SMOTE Pada Klasifikasi Hepatitis C Berbasis Random Forest dan Naïve Bayes," 2026.

[15] A. Al Ahad, B. Das, M. R. Khan, N. Saha, A. Zahid, and M. Ahmad, "Multiclass liver disease prediction with adaptive data preprocessing and ensemble modeling," Results in Engineering, vol. 22, Jun. 2024, doi: 10.1016/j.rineng.2024.102059.

[16] A. M. Ali et al., "Explainable Machine Learning Approach for Hepatitis C Diagnosis Using SFS Feature Selection," Machines, vol. 11, no. 3, Mar. 2023, doi: 10.3390/machines11030391.

[17] S. Kumari, D. Kumar, and M. Mittal, "An ensemble approach for classification and prediction of diabetes mellitus using soft voting classifier," International Journal of Cognitive Computing in Engineering, vol. 2, pp. 40–46, Jun. 2021, doi: 10.1016/j.ijcce.2021.01.001.

[18] A. Q. Md, S. Kulkarni, C. J. Joshua, T. Vaichole, S. Mohan, and C. Iwendi, "Enhanced Preprocessing Approach Using Ensemble Machine Learning Algorithms for Detecting Liver Disease," Biomedicines, vol. 11, no. 2, Feb. 2023, doi: 10.3390/biomedicines11020581.

[19] S. Yousefi, S. Yin, and M. G. Alfarizi, "Intelligent Fault Diagnosis of Manufacturing Processes Using Extra Tree Classification Algorithm and Feature Selection Strategies," IEEE Open Journal of the Industrial Electronics Society, vol. 4, pp. 618–628, 2023, doi: 10.1109/OJIES.2023.3334429.

[20] B. S. Ahamed, M. S. Arya, and A. O. Nancy V, "Prediction of Type-2 Diabetes Mellitus Disease Using Machine Learning Classifiers and Techniques," May 10, 2022, Frontiers Media S.A. doi: 10.3389/fcomp.2022.835242.

[21] D. A. Anggoro and S. S. Mukti, "Performance Comparison of Grid Search and Random Search Methods for Hyperparameter Tuning in Extreme Gradient Boosting Algorithm to Predict Chronic Kidney Failure," International Journal of Intelligent Engineering and Systems, vol. 14, no. 6, pp. 198–207, Dec. 2021, doi: 10.22266/ijies2021.1231.19.

[22] D. Valero-Carreras, J. Alcaraz, and M. Landete, "Comparing two SVM models through different metrics based on the confusion matrix," Comput Oper Res, vol. 152, Apr. 2023, doi: 10.1016/j.cor.2022.106131.

[23] M. A. Islam, M. Z. H. Majumder, and M. A. Hussein, "Chronic kidney disease prediction based on machine learning algorithms," J Pathol Inform, vol. 14, Jan. 2023, doi: 10.1016/j.jpi.2023.100189.

[24] A. Alizargar, Y. L. Chang, and T. H. Tan, "Performance Comparison of Machine Learning Approaches on Hepatitis C Prediction Employing Data Mining Techniques," Bioengineering, vol. 10, no. 4, Apr. 2023, doi: 10.3390/bioengineering10040481.

[25] Y. Fan, X. Lu, and G. Sun, "IHCP: interpretable hepatitis C prediction system based on black-box machine learning models," BMC Bioinformatics, vol. 24, no. 1, Dec. 2023, doi: 10.1186/s12859-023-05456-0.