applied sciences

MDPI

Article

Enhancing Machine Learning Models Through PCA, SMOTE-ENN, and Stochastic Weighted Averaging

Youngjin Han and Inwhee Joe *

Department of Computer Science, Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul 04763, Republic of Korea; sni94@hanyang.ac.kr
* Correspondence: iwjoe@hanyang.ac.kr

Abstract: Predicting survival outcomes in critical accidents has been a focal point in machine learning research. This study addresses several limitations of existing methods, including insufficient management of data imbalance, lack of emphasis on hyperparameter tuning, and proneness to overfitting. Many existing models struggle to generalize effectively on imbalanced datasets or depend on default hyperparameter settings, resulting in biased predictions. By integrating Principal Component Analysis (PCA), hyperparameter optimization, and resampling methods, as well as combining Edited Nearest Neighbors (*ENN*) with the Synthetic Minority Oversampling Technique (SMOTE), the model significantly improves predictive accuracy and model generalization. An ensemble model combining seven machine learning algorithms—Logistic Regression, Support Vector Machine, KNN, Random Forest, XGBoost, LightGBM, and CatBoost—was applied to predict survival outcomes. Stochastic Weighted Averaging (SWA) was applied to mitigate overfitting and enhance generalization. The accuracy increased from 91.97% to 94.89% after SWA was applied in this specific scenario. The combination of PCA-based dimensionality reduction, hyperparameter tuning, and resampling techniques (*ENN* + SMOTE) ensured the model handled data imbalance and optimized predictive accuracy. The final model demonstrated excellent performance, with Area Under the Curve (AUC) and Average Precision (AP) values both reaching 0.98, indicating high accuracy and precision. These improvements were validated using the Titanic dataset in a binary classification problem of predicting passenger survival. The results emphasize that ensemble learning, enhanced by SWA, offers a powerful framework for handling imbalanced and complex datasets, providing significant advancements in predictive modeling accuracy. This study provides insights into how machine learning techniques can be effectively combined to solve classification challenges in real-world scenarios.

Keywords: survival prediction; hyperparameter optimization; ensemble methods; data imbalance; PCA; Bayesian optimization; voting; stacking; SWA

Citation: Han, Y.; Joe, I. Enhancing Machine Learning Models Through PCA, SMOTE-ENN, and Stochastic Weighted Averaging. *Appl. Sci.* **2024**, *14*, 9772. https://doi.org/10.3390/app14219772

Academic Editor: Douglas O'Shaughnessy

Received: 29 September 2024
Revised: 18 October 2024
Accepted: 22 October 2024
Published: 25 October 2024

1. Introduction

Predicting survival in major accidents has long been a focus of machine learning research, and numerous studies have attempted to improve the accuracy and reliability of predictions using the Titanic dataset as a benchmark [1]. Earlier studies have investigated different machine learning algorithms like logistic regression, KNN, SVM, and decision trees for survival prediction models [2]. Although these studies have successfully demonstrated the potential of these models for predicting survival outcomes, they have not fully addressed significant challenges. A key issue highlighted in existing research is the limited emphasis on hyperparameter tuning. Many previous studies have employed conventional methods like grid search or default settings, which may not maximize model performance [3]. Additionally, the problem of imbalanced datasets, where the number of survivors is significantly lower than the number of non-survivors, is often inadequately addressed, leading to biased predictions and reduced model effectiveness [4].

This study applies Principal Component Analysis (PCA) alongside regularization techniques to mitigate overfitting and combines the Synthetic Minority Oversampling Technique (SMOTE) with the Edited Nearest Neighbors (*ENN*) technique (*ENN* + SMOTE) to address data imbalance [5]. A comprehensive comparative analysis of three hyperparameter optimization techniques—grid search, random search, and Bayesian optimization—is performed, and an optimal ensemble approach combining Voting and Stacking methods, along with Stochastic Weighted Averaging (SWA) is utilized to achieve optimal performance [6]. Moreover, before applying the ensemble techniques, the weights were processed as a combined metric that maximizes the interval and accuracy of both Test Accuracy and Train Accuracy [7]. This approach ensures that the ensemble models are not only accurate but also well-generalized, balancing the trade-off between training and testing performance. By systematically comparing these methods, this study seeks to improve the predictive accuracy and generalizability of machine learning models in the context of survival prediction in serious accidents. This approach not only addresses the limitations observed in previous studies but also provides a more robust framework for predictive modeling in critical scenarios, offering significant contributions to the field of machine learning, particularly in optimizing predictive models for imbalanced and complex datasets [8].

This paper proposes a novel method for predicting the occurrence of serious accidents based on data-driven approaches. The contribution of this study is to find a hybrid prediction model with a machine learning approach that has high prediction accuracy, low computation time, and low memory usage.

This paper is organized as follows: Section 2 describes the materials and estimation methodology; Section 3 presents experimental results, comparisons, and discussions; and Section 4 summarizes the conclusions.

2. Materials and Methods

This section outlines the materials and methodologies employed in this study, including the datasets, the proposed methods for handling imbalanced data, the classifiers used, the feature subset selection process, and the performance evaluation metrics.

2.1. Datasets

The dataset used in this study is the well-known Titanic dataset, sourced from Kaggle (https://www.kaggle.com/competitions/titanic/data, accessed on 5 September 2024) [9]. The training set contains 891 records, while the test set includes 418 records. Key features include passenger class (Pclass), gender, age, number of siblings or spouses on board (SibSp), number of parents or children on board (Parch), ticket number, fare, cabin number, and port of embarkment (Embarked). The target variable is 'survival', which indicates whether the passenger survived the disaster (1) or did not survive (0). The class imbalance in the dataset, with 61.62% non-survivors and 38.38% survivors, was addressed using robust ensemble models and cost-sensitive learning to minimize bias toward the majority class. Additional experiments with artificially increased imbalance confirmed the stability of the model under these conditions. The distribution of passenger classes (Pclass) in the training set shows that 24.24% of passengers were in the first class, 20.65% in the second class, and 55.11% in the third class. Similarly, in the test set, 25.60% of passengers were in the first class, 22.25% in the second class, and 52.15% in the third class. Survival rates within each class in the training data were 62.96% survived in Pclass 1, 47.28% survived in Pclass 2, and 24.24% survived in Pclass 3. This consistency between the training and test sets validates the use of the computed metrics, as the test set closely mirrors the class distribution found in the training data, ensuring that the model's performance is relevant to real-world scenarios. An exploratory data analysis revealed that 'Fare' has the strongest positive correlation with survival (0.257), while 'Pclass' has a notable negative correlation (-0.338). Other features such as 'Parch' and 'Age' have weaker correlations with survival.

2.2. Methodology

Figure 1 outlines a generalizable methodology that can be applied to any classification problem, including survival prediction. While the methodology was demonstrated on the Titanic dataset, the individual steps—data preprocessing, feature engineering, resampling, dimensionality reduction, model selection, and optimization—can be adapted to other datasets with similar characteristics.

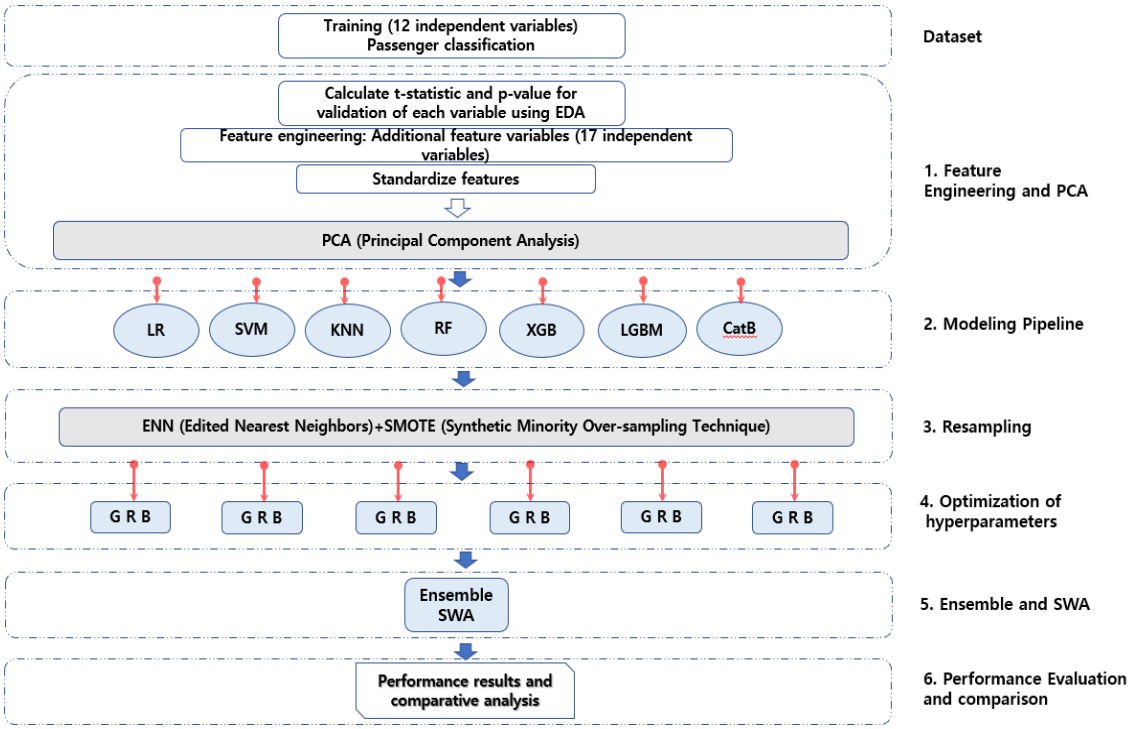

Figure 1. Structure of a survival prediction research system.

The proposed 'Structure of Survival Prediction Research System' study consists of six major steps: (1) Feature Engineering and PCA, (2) Modeling Pipeline, (3) Resampling, (4) Optimization of hyperparameters, (5) Ensemble and SWA, and (6) Performance Evaluation and Comparison. These steps are summarized in Figure 1.

The first part of this study involves conducting data statistics for independent variables and preprocessing. The validity test of each variable was performed using exploratory data analysis (EDA). The data were divided into survival groups by removing redundant information or outliers, and additional independent variables were generated based on the distribution of features, such as age, fare, and embarkation location. This resulted in the creation of 17 variables.

Next, dimensionality reduction techniques using PCA were employed to select the optimal model. Third, imbalanced data were addressed using the best-performing of the six resampling methods. Fourth, a comprehensive comparison of each model using grid search, random search, and Bayesian optimization is essential for evaluating their performance and selecting the best hyperparameters. Fifth, the models were compared by combining Stacking and Voting ensemble techniques and Stochastic Weighted Averaging (SWA) to perform the evaluation, and finally, the performance evaluation was verified.

Feature engineering involves the extraction of key features relevant to the problem at hand. In the case of the Titanic dataset, domain knowledge was used to engineer features such as passenger titles, family size, and deck information. However, this step is adaptable across datasets. For other datasets, domain-specific features can be engineered based on available information. While specific features like 'Title' were used in the Titanic case, in other datasets, features that capture important domain-specific relationships would be

engineered. Thus, the methodology is flexible enough to be applied to other datasets, with the feature engineering step being tailored to the characteristics of the new dataset. This flexibility ensures generalizability across different applications.

2.2.1. Feature Engineering and Principal Component Analysis (PCA)

The first part of this study involves conducting data statistics for independent variables and preprocessing. The validity test of each variable was performed using exploratory data analysis (EDA) to ensure that relevant and significant features were considered. The dataset was divided into survival groups by removing redundant information and outliers. Based on the distribution of important features, such as age, fare, and embarkation location, additional independent variables were generated. This feature engineering process resulted in the creation of 17 variables, which provided a more comprehensive representation of the passengers' profiles and their likelihood of survival.

The 17 variables included standard features like Passenger Class (Pclass), Age, Gender (Sex), and Fare, as well as newly engineered variables derived from existing ones, such as the number of family members on board, the interaction between socioeconomic class and age, and imputed values for missing data points like age. Additionally, the port of embarkation (Embarked) and fare ranges were categorized to better capture the impact of ticket price and boarding location on survival rates.

Principal Component Analysis (PCA) was then applied to reduce the dimensionality of these variables, streamlining the model and reducing computational requirements. While PCA reduces the dimensionality of the feature space, improving computational efficiency, it also aids in mitigating overfitting by retaining only the most significant components. The ensemble of seven classifiers was selected for its ability to capture different aspects of the data, with models such as SVM excelling in high-dimensional spaces and Random Forests providing robustness against overfitting. The use of an ensemble classifier, particularly when combined with Stochastic Weighted Averaging (SWA), further enhances generalization by averaging predictions from diverse models, leading to improved performance compared with a single, finely tuned model. By converting the 17 original variables into a smaller set of independent components, PCA ensured that the most significant variance in the dataset was retained while minimizing the risk of overfitting.

Principal Component Analysis (PCA) was utilized in this study to improve model performance in the Titanic survival prediction model. While PCA reduces dimensionality, it may limit the interpretability of individual features, which could be important in some contexts. However, in survival prediction systems such as this, interpretability is often secondary to predictive accuracy and performance. In this context, interpretability is not a critical requirement, as the goal is to maximize the model's accuracy in predicting survival outcomes. PCA effectively removes irrelevant data, highlighting key patterns while reducing overfitting and improving the model's generalization capabilities [10]. PCA is also valuable for reducing computational complexity, especially when dealing with large datasets. Although the computational cost increases when using an ensemble of multiple machine learning classifiers, this cost is justified by the significant gains in accuracy and robustness that ensemble methods provide. The added complexity is offset by the improved generalization and predictive performance achieved through the combination of PCA and ensemble learning [11].

Principal Component Analysis (PCA): A dimensionality reduction technique that transforms a large set of features into a smaller set while retaining as much of the original information as possible. PCA identifies the directions (principal components) in which the data varies the most and projects the data onto these new axes. The first principal component captures the largest amount of variance in the data, followed by the second component, and so on. The transformation is achieved through an orthogonal linear

transformation, where the data matrix X is transformed into a new matrix Z, representing the principal components. The principal components are calculated as

$$Z = XW$$

Here, X is the original data matrix, W is the matrix of eigenvectors (principal components), and Z is the transformed data matrix (projected data). The eigenvectors correspond to the directions of maximum variance in the data, while the eigenvalues indicate the amount of variance explained by each principal component. The principal components are ordered such that

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

where λ_1 represents the largest variance, λ_2 the second largest, and so on. By selecting the first few principal components, PCA reduces the dimensionality of the dataset while retaining the most important variance.

This transformation simplifies the model, reduces computation time by lowering the number of features, and improves model performance by minimizing overfitting and enhancing generalization.

2.2.2. Modeling Pipeline

Seven classifiers were employed in this study to analyze their effectiveness in predicting survival. These classifiers were chosen for their strengths in handling various aspects of classification tasks [12].

Logistic Regression: A statistical model widely used for binary classification problems where the goal is to predict the probability that a given input instance belongs to one of two classes. This model is particularly effective when the relationship between features and binary outcomes is approximately linear.

Support Vector Machine (SVM): A powerful classification algorithm that finds a hyperplane that best separates classes in the feature space. SVM's primary objective is to create a maximum-margin hyperplane that separates data points from different classes, focusing on the nearest data points, termed support vectors. SVMs are particularly effective in high-dimensional spaces and are robust to overfitting, especially when the number of dimensions exceeds the number of samples [13].

K-Nearest Neighbors (KNNs): KNNs is a simple, yet powerful, classification algorithm that makes predictions based on the majority class of the k nearest data points in the feature space. KNNs assumes that similar data points are close to each other, and it assigns a class label to a new data point by finding the majority label among its closest neighbors. One of the key strengths of KNNs is that it is non-parametric, meaning it makes no assumptions about the underlying data distribution, making it useful for various types of data [14].

Random Forest: Random Forest is an ensemble technique that builds multiple decision trees to enhance predictive accuracy by averaging their predictions in regression or voting in classification tasks. Random Forest is highly effective in handling large, high-dimensional datasets and is resistant to overfitting due to its use of multiple trees. It also handles missing values and maintains accuracy [15].

XGBoost (Extreme Gradient Boosting): An advanced implementation of the gradient boosting framework, designed for efficiency, flexibility, and high performance. Gradient boosting builds models sequentially, where each new model attempts to correct the errors introduced by the previous one. XGBoost introduces innovations such as regularization techniques and efficient sparse data handling, significantly improving performance. Its main advantages include handling missing data, parallel processing, and regularization techniques that prevent overfitting [16].

LightGBM: A highly efficient gradient boosting framework optimized for both speed and performance. It uses a leaf-wise (best-first) tree growth algorithm to achieve deeper trees and better accuracy compared with level-wise (depth-wise) tree growth used in other

boosting algorithms. LightGBM is particularly effective in scenarios where the dataset is very large and has high dimensionality [17].

CatBoost: Another powerful gradient boosting algorithm specifically designed to efficiently handle categorical features. Based on the gradient boosting framework, CatBoost improves the way categorical variables are handled and reduces overfitting. It introduces a technique called 'order boosting' to reduce bias due to target leakage, ensuring that the learning process remains unbiased by splitting the data into separate training and validation sets at each iteration [18].

2.2.3. Resampling

In addressing the challenge of imbalanced datasets, this study employs several resampling techniques aimed at improving the classification models' performance. Six different resampling techniques (SMOTE, SMOTEENN, SMOTETomek, ADASYN, SMOTE + ADASYN, and *ENN* + SMOTE) were applied to evaluate their effectiveness in addressing class imbalance. After comparison, *ENN* + SMOTE was selected as the optimal resampling technique for the proposed methodology due to its superior performance. The other techniques were tested but not included in the final methodology. Additionally, experiments confirmed that the order of resampling, hyperparameter optimization, and ensemble learning significantly impacts performance, with the current order yielding the best results.

Specifically, the methods were applied after feature engineering and before model training to ensure that the dataset was balanced before feeding it into the models. These resampling techniques—SMOTE, SMOTEENN, SMOTETomek, ADASYN, SMOTE + ADASYN, and *ENN* + SMOTE—help mitigate class imbalance by generating synthetic minority class samples or cleaning noisy majority class samples. These methods are not specific to the Titanic dataset but are generally applicable to any dataset suffering from imbalanced class distributions, ensuring a more generalized methodology.

SMOTE (Synthetic Minority Oversampling Technique): SMOTE is frequently used to balance data by creating synthetic examples for minority class samples through interpolation. This approach helps balance the class distribution, enabling the classifier to learn from a more representative sample of the data [19].

SMOTEENN (SMOTE + Edited Nearest Neighbors): SMOTEENN combines the synthetic generation capability of SMOTE with the data cleaning functionality of the Edited Nearest Neighbors (*ENN*) technique. SMOTE generates synthetic instances, while *ENN* removes instances from the majority class that are incorrectly classified by their nearest neighbors, thus enhancing the classifier's performance by reducing noise and the risk of overfitting [20].

SMOTETomek: SMOTETomek combines SMOTE with Tomek links, a technique that identifies and removes instances from different classes that are each other's nearest neighbors. This process not only balances the class distribution but also cleans the overlapping instances between classes, improving decision boundaries for the classifier.

ADASYN (Adaptive Synthetic Sampling): ADASYN is an adaptive method that generates synthetic instances for minority class samples that are more difficult to classify based on the distribution of the data. This method focuses on improving the classifier's ability to generalize on harder-to-learn instances, resulting in better performance, especially for datasets with complex decision boundaries.

SMOTE + ADASYN: This technique combines SMOTE and ADASYN, first using SMOTE to generate synthetic instances and then applying ADASYN to further focus on hard-to-learn minority class examples, leading to improved generalization and model accuracy.

ENN + SMOTE: This method combines the strengths of *ENN* and SMOTE, where *ENN* removes noisy or misclassified instances from the majority class, and SMOTE generates

synthetic instances for the minority class. This combination reduces noise while improving the class balance. The process of *ENN* is formalized as

$$ENN(D) = \{x_i \in D \mid y_i \neq majority(kNN(x_i))\}$$

where D is the dataset, x_i represents an instance, and y_i is its label. The $kNN(x_i)$ function finds the nearest neighbors of x_i and the misclassified majority class instances are removed, resulting in a cleaner dataset.

Next, SMOTE generates synthetic instances as

$$x_{new} = x_i + \lambda \times (x_{NN} - x_i)$$

where x_{new} is the synthetic data point, x_i is the minority class instance, x_{NN} is one of the nearest neighbors, and λ is a random number between 0 and 1. This combination of *ENN* cleaning and SMOTE oversampling results in a more balanced and less noisy dataset, improving classifier performance.

2.2.4. Optimization of Hyperparameters

In machine learning, hyperparameter optimization is critical for improving model performance by identifying the best set of hyperparameters that minimize model error. This study employs three distinct hyperparameter tuning strategies—grid search (G), random search (R), and Bayesian optimization (B), collectively referred to as GRB in Figure 1—to optimize model performance. Each method adjusts hyperparameters in different ways, and the one that achieves the highest accuracy is selected as the optimal approach. The objective is to identify the most suitable hyperparameter combination while maintaining computational efficiency.

Grid Search: Grid search is an exhaustive search method that systematically tests all possible combinations of hyperparameters within a predefined grid. Each combination is evaluated across multiple cross-validation folds, and the combination that minimizes the average loss is selected. This method ensures a global optimum is found but can be computationally expensive for large grids.

Random Search: Random search improves scalability by randomly sampling a fixed number of hyperparameter combinations from the grid. Instead of evaluating all combinations, it tests a subset of possibilities, which significantly reduces computation time. Although not exhaustive, random search can often find near-optimal solutions more efficiently, particularly when dealing with large grids.

Bayesian Optimization: Bayesian optimization, implemented via BayesSearchCV, is an advanced technique that constructs a probabilistic model of the objective function. This surrogate model is used to iteratively select the most promising hyperparameter combinations for evaluation. Bayesian optimization focuses on the most likely optimal regions of the hyperparameter space, updating the surrogate model with new information at each step. This method balances exploration and exploitation, leading to faster convergence toward the optimal hyperparameters.

By employing these techniques, this study seeks to maximize model performance while efficiently navigating the trade-off between accuracy and computational complexity.

2.2.5. Ensemble and SWA

The optimal implementation of hyperparameters for the seven models is carried out using an ensemble learning approach that combines Stacking and Voting classifiers. The models are wrapped within a pipeline that includes feature selection through SelectKBest, and the models themselves are retrained and simulated with varying random states. The final probabilities are then evaluated using the Stochastic Weighted Averaging (SWA) technique to enhance model performance.

Ensemble Approach (Stacking and Voting): The ensemble method leverages the strengths of individual models by combining them to improve predictive accuracy and

robustness. In stacking, multiple base models are trained on the same training data, and a meta-classifier is used to make the final prediction based on the outputs of these models. Voting combines the predictions of different models using a soft voting mechanism, where the final prediction is based on the average predicted probabilities from all models.

Stochastic Weighted Averaging (SWA): SWA enhances the generalization of the model by averaging the weights of several models trained with slight variations, such as different random states or random restarts. By retraining the models and averaging their predictions, SWA smooths out the fluctuations in model weight updates, leading to improved generalization and more stable predictions. Performance metrics are calculated using test data, and the results are compared before and after applying SWA to assess its impact on model generalization and accuracy.

2.2.6. Performance Evaluation and Comparison

The predictive modeling task in this study is a binary classification problem, and the performance of the models is evaluated using several key metrics. The models produce four possible outcomes. True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). These outcomes help assess the model's strengths and weaknesses, ensuring that the selected algorithm is well suited for the classification task. The effectiveness of the model in distinguishing survival outcomes is measured using the following evaluation metrics: accuracy, precision, recall (Equation (5)), and F1 score (Equations (1)–(4)). Additionally, the Receiver Operating Characteristic (ROC) curve and Precision–Recall (P-R) curve are used to provide further insights into model performance [21].

To interpret prediction error and optimize performance, the mean absolute error (MAE) is calculated as the average difference between the predicted and actual values (Equation (6)). A combinatorial approach is employed to create a weighted combination of multiple models or features, allowing the system to leverage the strengths of different models for superior predictive accuracy [22].

These evaluation metrics and techniques ensure that model optimization is achieved by fine-tuning the algorithm's parameters, thereby maximizing classification accuracy and improving generalization.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN} \tag{5}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} | \, y_i - \hat{y}_i \, | \tag{6}$$

3. Experimental Results and Discussions

3.1. Data Preprocessing and Feature Engineering

In this study, a comprehensive preprocessing and feature engineering process was implemented on the Titanic dataset. The model was tested on Python and r2022a platform on a computer with a 4-core 1.6 GHz Intel Core i5 8250U and 16 GB RAM. The Titanic dataset is provided with a predefined training set (891 records) and test set (418 records). These sets were concatenated temporarily during the preprocessing step to ensure consistent handling of missing values, feature engineering, and scaling across the entire dataset. The missing

values were handled using median or mode imputation, and new features, such as title, family size, interactive features, deck, and family survival patterns, were extracted and generated. After preprocessing, the dataset was split back into its original train and test sets. Categorical variables were encoded into numerical values, and feature scaling was applied using 'StandardScaler'. While Step 1 (Feature Engineering) may appear dataset-specific in this study, the concept of feature engineering is highly generalizable. For other datasets, domain expertise would be used to extract meaningful features. Feature engineering is not limited to the Titanic dataset but rather serves as an adaptable step in the machine learning process, applicable to any problem where domain-specific features are beneficial. This comprehensive feature engineering approach was designed to capture the complex relationships between variables and improve the overall performance of the model in predicting survival.

3.2. Feature Analysis

3.2.1. Feature Exploration and Visualization

For optimization, this study uses a custom function (Figure 1) to automate the statistical analysis and visualization process for key variables in the dataset. This function produces a single plot that overlays the feature distributions for the surviving group and the non-surviving group. The distributions are plotted using histograms and kernel density estimates (KDEs), allowing a visual comparison of the behavior of the features across the two groups.

The four plots illustrate how key characteristics—Age, SibSp (siblings/spouse aboard), Parch (parents/children aboard), and Fare—are distributed based on Titanic passengers' survival status (Figure 2): (a) survivors tend to be slightly younger than non-survivors, but the difference is not statistically significant (p~0.053), and both groups show a non-normal distribution; (b) while survivors generally have fewer siblings or spouses on board, the difference between survivors and non-survivors is not significant (p~0.292), with both distributions being non-normal; (c) survivors tend to have higher Parch values, indicating they traveled with more family members, and this difference is statistically significant (p~0.015), though both groups are non-normally distributed; and (d) survivors paid significantly higher fares, indicating that passengers in higher classes had better chances of survival, with the difference being highly significant (p = 0.000) and both groups showing non-normal distributions.

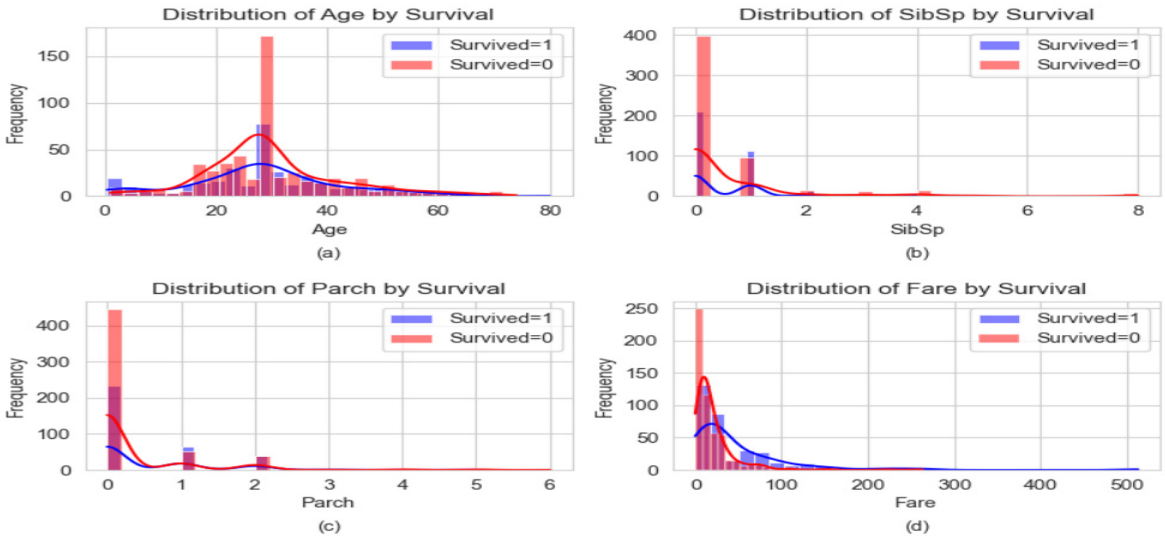

Figure 2. Survival standard deviation distribution: (**a**) Age distribution by survival, (**b**) SibSp distribution by survival, (**c**) Parch distribution by survival, (**d**) Fare distribution by survival.

In summary, Fare and Parch show significant differences between survivors and non-survivors, while Age and SibSp do not. All features exhibit non-normal distributions, which is typical for these types of data. This automated feature exploration ensures a consistent analysis of multiple variables, providing insight into key survival predictors.

3.2.2. Correlation Analysis of Features

Figure 3 generates a correlation heatmap to evaluate and compare feature correlations across multiple steps of data processing to ensure consistency across the dataset. This heatmap shows the normalized and processed correlation matrix of the dataset, excluding the target variable (Survived). The 17 features displayed in the heatmap represent the processed and engineered variables utilized for model training after the completion of all preprocessing steps. These features are the result of various preprocessing steps, such as categorical variable encoding, feature interactions, and missing value imputation. The features included in this heatmap are the features selected to train the model after applying the necessary transformations. The 17 variables used are defined in Table 1, which details the rationale for including each feature and its expected influence on the survival prediction.

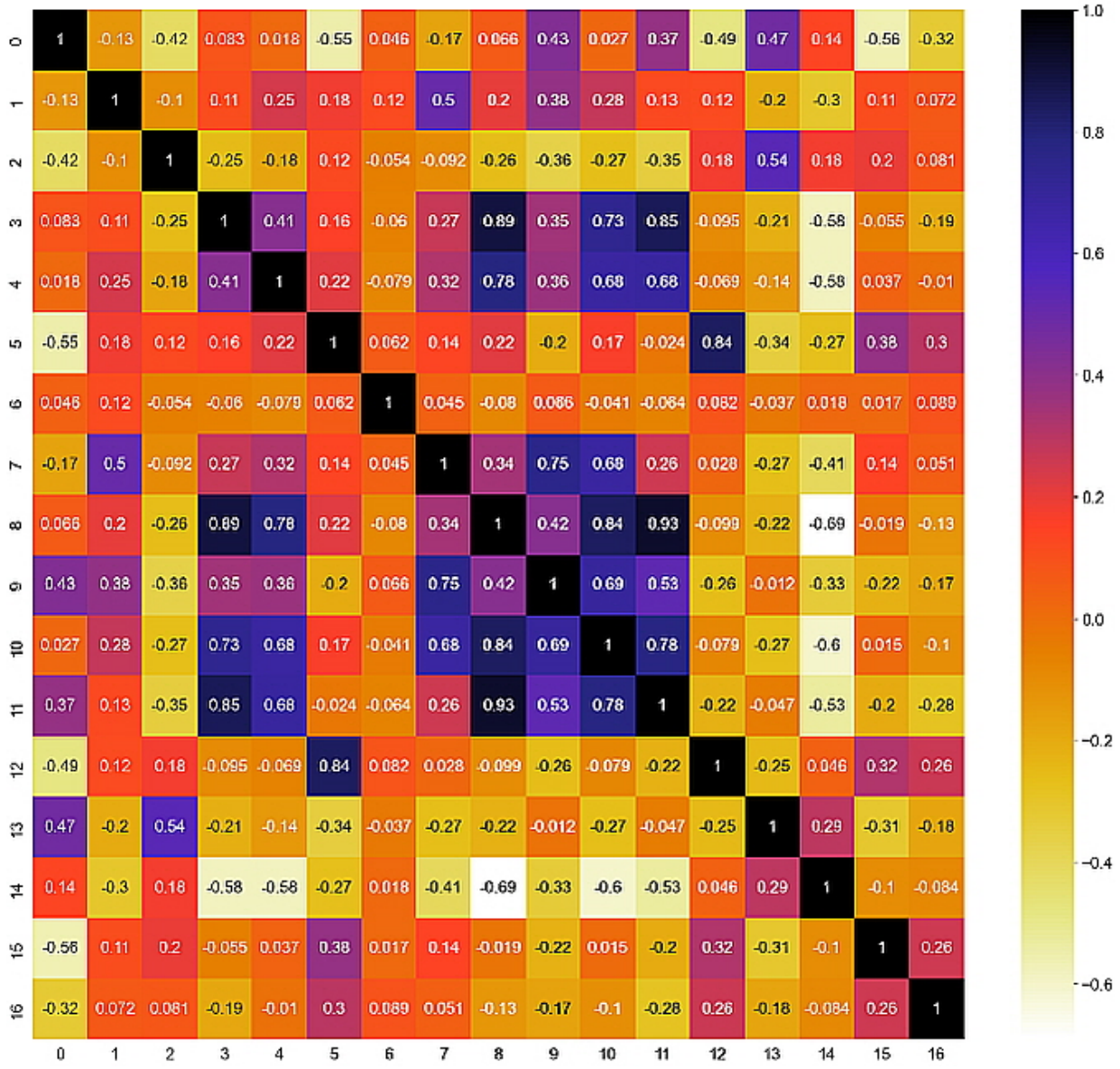

Figure 3. Comparison of functional correlations in data.

Table 1. Summary of the 17 Features Used for Survival Prediction.

#	Feature Name	Description
0	Age	Age of the passenger. Younger passengers had a higher survival likelihood.
1	Fare	The fare paid for the ticket. Higher fares often indicated better survival chances.
2	Title	Social status derived from the passenger's name (e.g., Mr., Mrs., Miss).
3	FamilySize	Size of the passenger's family onboard, including siblings, spouses, parents, and children.
4	Fare_Per_Person	The fare divided by family size, reflecting relative cost per person.
5	Pclass_1	Binary variable for first-class passengers, who generally had better survival chances.
6	Pclass_2	Binary variable for second-class passengers.
7	Pclass_3	Binary variable for third-class passengers, who generally had lower survival rates.
8	Sex_Male	Binary variable indicating male passengers. Gender played a key role in survival.
9	Sex_Female	Binary variable indicating female passengers.
10	Embarked_C	Binary variable for passengers who embarked from Cherbourg.
11	Embarked_S	Binary variable for passengers who embarked from Southampton.
12	Embarked_Q	Binary variable for passengers who embarked from Queenstown.
13	IsAlone	Binary feature indicating whether the passenger was traveling alone.
14	Deck	Deck level of the passenger's cabin. Higher decks may have offered better survival chances.
15	FamilySurvival	Reflects the survival likelihood of family members onboard.
16	Age Class	Interaction between age and class, capturing the combined effect on survival.

3.3. Overfitting

3.3.1. Model Evaluation and Overfitting Detection

The evaluated models include Logistic Regression, SVM, KNN, Random Forest, XGBoost, LightGBM, and CatBoost. The main performance metrics calculated and displayed are the area under the Receiver Operating Characteristic (ROC) curve and the Average Precision (AP) of the Precision–Recall (P-R) curve. Among the models, CatBoost exhibited the highest AUC (0.9411) and Average Precision (0.9269), making it the most effective model in terms of distinguishing between survivors and non-survivors with high precision. Random Forest also showed strong performance, with an AUC of 0.9289, though its lower Average Precision (0.8877) indicates it may be less accurate in predicting positive outcomes compared with CatBoost. Logistic Regression (AUC 0.9257, AP 0.9144) and LightGBM (AUC 0.9205, AP 0.9019) delivered balanced and reliable results, while SVM and KNNs had the lowest AUCs (0.9045 and 0.8900, respectively), suggesting that these models may not be as effective, particularly when considering precision in positive predictions (Figure 4).

In addition to AUC and Precision–Recall metrics, the models' performance was further evaluated using additional metrics such as accuracy, precision, recall, and F1 score, as outlined in Table 2. CatBoost consistently delivers strong performance, achieving an accuracy of 0.87, a precision of 0.88, and an F1 score of 0.84, reinforcing its superiority in handling this classification task. LightGBM also stands out with an accuracy of 0.86, while SVM and Random Forest demonstrate slightly lower accuracy but remain competitive.

Interestingly, KNNs lags slightly behind in terms of both accuracy and F1 score, indicating that it may not be as well suited for this dataset.

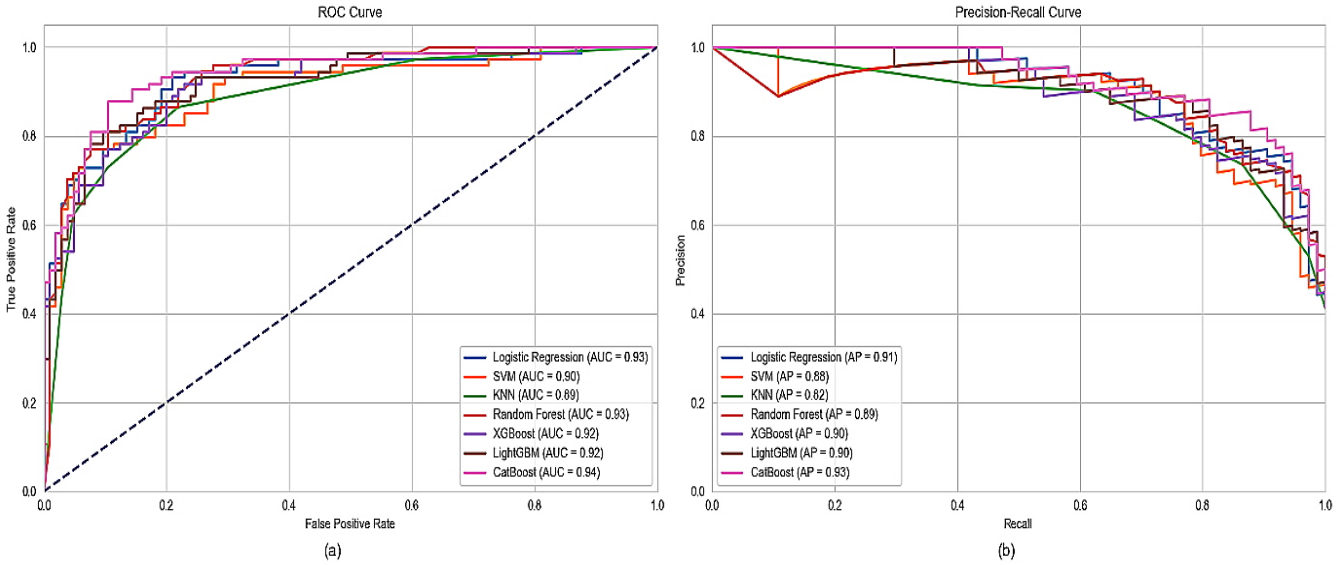

Figure 4. Overfitting check by all models: (**a**) ROC curve, (**b**) P-R curve.

Table 2. Model Performance Metrics.

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.84	0.84	0.77	0.80
SVM	0.86	0.89	0.76	0.82
KNN	0.83	0.83	0.73	0.78
Random Forest	0.84	0.84	0.77	0.80
XGBoost	0.83	0.81	0.77	0.79
LightGBM	0.86	0.85	0.81	0.83
CatBoost	0.87	0.88	0.80	0.84

The overfitting assessment, visualized in Figure 5, highlights significant overfitting in models such as Random Forest and XGBoost, where training accuracies (98.74% and 98.03%) were much higher than test accuracies (84.36% and 83.24%, respectively). This indicates that these models tend to memorize the training data and struggle to generalize to new data. Logistic Regression and SVM, with closely aligned training and test accuracies (Logistic: 85.25% training, 84.36% test; SVM: 86.38% training, 86.03% test), exhibited minimal overfitting. CatBoost also showed a relatively small gap between training (92.56%) and test accuracy (87.15%), although there was still a slight tendency to overfit. LightGBM demonstrated a larger gap (97.05% training, 86.03% test), but it remained within a reasonable range.

These results underscore the tradeoff between model complexity and generalization. Models like Random Forest and XGBoost, which are more complex, tend to overfit unless properly regularized, while simpler models like Logistic Regression and SVM generalize better. This emphasizes the importance of using validation techniques such as cross-validation, ensembling, or data augmentation to improve model robustness and mitigate overfitting in real-world applications where the model may fail when exposed to new data.

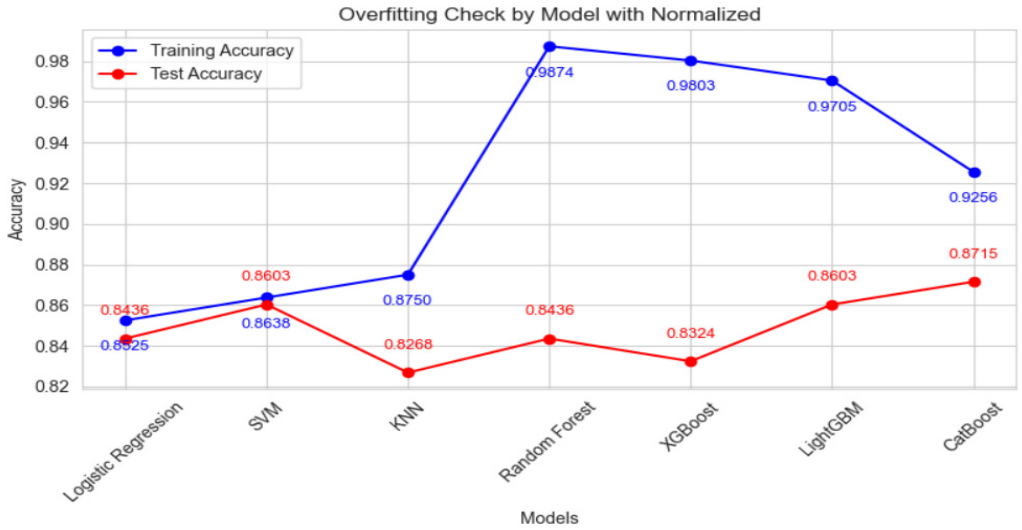

Figure 5. Overfitting check by all models with normalization.

3.3.2. Principal Component Analysis (PCA) and Overfitting Mitigation

Figure 6 demonstrates that applying Principal Component Analysis (PCA) significantly reduces overfitting in all models compared with their performance without PCA. Previously, models such as Random Forest exhibited substantial overfitting, with a large discrepancy between training and test accuracies. After applying PCA, overfitting was mitigated, resulting in more balanced performance across training and test datasets. For example, Random Forest, which previously had a training accuracy of 98.74% and a test accuracy of 84.36%, now shows more balanced results with a training accuracy of 83.43% and a test accuracy of 83.24%. This indicates that PCA effectively reduces model complexity, helping to prevent overfitting by removing noise from the training data, resulting in improved generalization to unseen data.

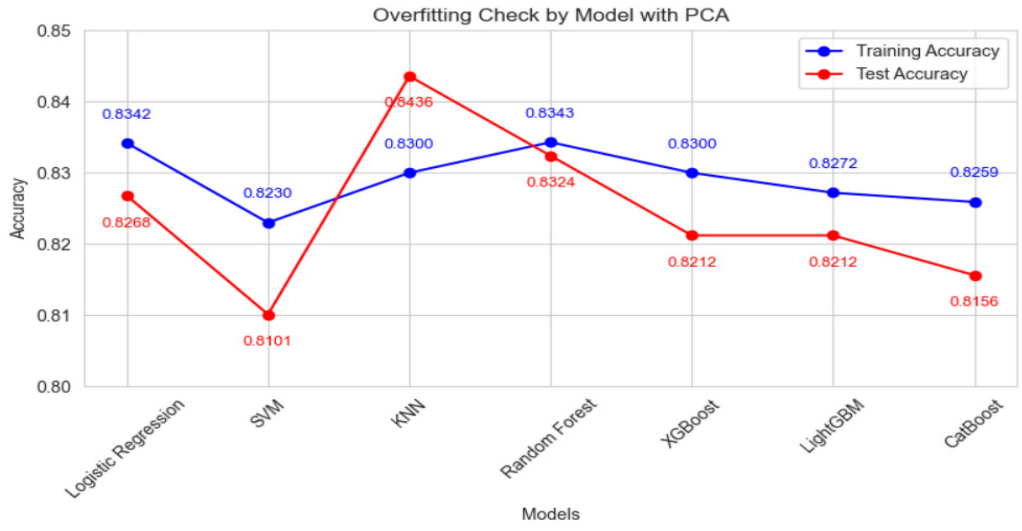

Figure 6. Overfitting check by all models with PCA.

Figure 7 compares the models with and without PCA, focusing on the difference between training and test accuracies, a direct indicator of overfitting. A large difference suggests that the model is overfitting, as it performs well on the training data but poorly on the test data.

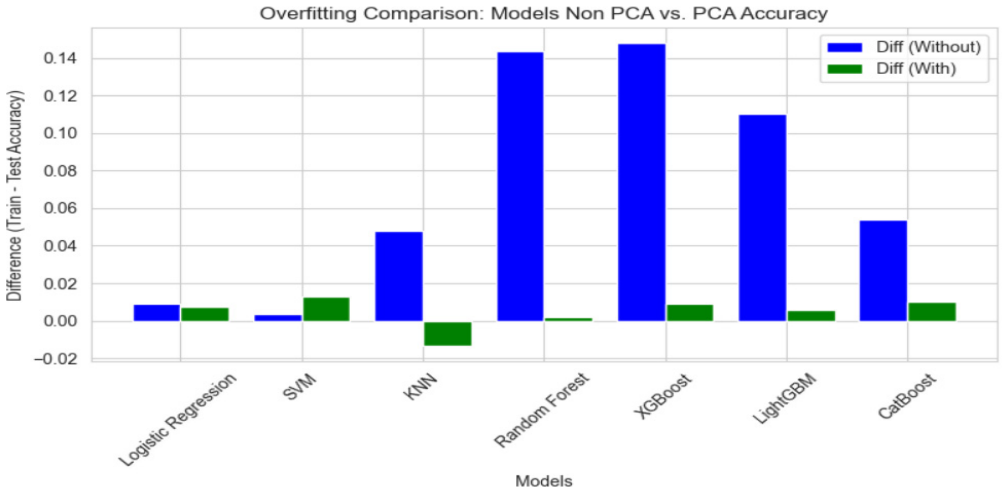

Figure 7. Overfitting comparison: Non-PCA vs. PCA accuracy.

Before applying PCA, models like Random Forest and XGBoost exhibited significant overfitting. For Random Forest, the gap between training and test accuracies was 14.38% (98.74% vs. 84.36%), and for XGBoost, the gap was 14.79% (98.03% vs. 83.24%). After applying PCA, these gaps were dramatically reduced to 0.19% (83.43% vs. 83.24%) for Random Forest and 0.88% (83.00% vs. 82.12%) for XGBoost, demonstrating that PCA was very effective in controlling overfitting in these models.

For other models, the initial overfitting was less severe but still present. For example, Logistic Regression showed a small gap between training and test accuracies before PCA (0.85% difference, 85.25% vs. 84.36%), and after PCA, the gap remained minimal (0.74%, 83.42% vs. 82.68%). Similarly, LightGBM saw its training–test accuracy gap reduce from 10.17% (97.05% vs. 86.03%) to 0.60% (82.72% vs. 82.12%) after applying PCA. CatBoost also benefited from PCA, with the overfitting gap decreasing from 5.41% (92.56% vs. 87.15%) to 1.02% (82.59% vs. 81.56%).

In conclusion, PCA proved to be an effective strategy in mitigating overfitting for all the models evaluated, particularly for complex models like Random Forest and XGBoost, where the initial overfitting was more pronounced. Across all models, PCA helped bring training and test accuracies closer together, thereby improving generalization and ensuring better performance on unseen data.

3.4. Resampling Methods

To address the challenge of imbalanced data in the Titanic dataset, six resampling methods were applied and evaluated. The purpose of this process was to balance the number of survivors and non-survivors, which is critical for improving the model's ability to generalize and accurately predict survival outcomes. The six methods used were SMOTE (Synthetic Minority Oversampling Technique), which generates synthetic samples from the minority class; SMOTETomek, a combination of SMOTE and Tomek links to remove noisy data points; ADASYN, which generates synthetic data focused on the minority class instances that are harder to classify; SMOTE+ADASYN, a hybrid of SMOTE and ADASYN techniques; TomekLinks, an undersampling method that removes overlapping majority class samples; and *ENN* + SMOTE, which combines Edited Nearest Neighbors (*ENN*) to clean noisy majority samples with SMOTE for minority oversampling.

Among these methods, as shown in Figure 8, *ENN* + SMOTE demonstrated the best performance, with a cross-validation accuracy of 0.89210, outperforming the other methods. This method's strength lies in its ability to not only generate synthetic minority samples but also to clean the majority class by removing borderline or misclassified samples through *ENN*, which enhances model generalization and reduces the risk of overfitting. SMOTETomek and TomekLinks also performed well, with accuracies of 0.84102 and 0.83650, respectively, while SMOTE, SMOTE + ADASYN, and ADASYN yielded lower performance.

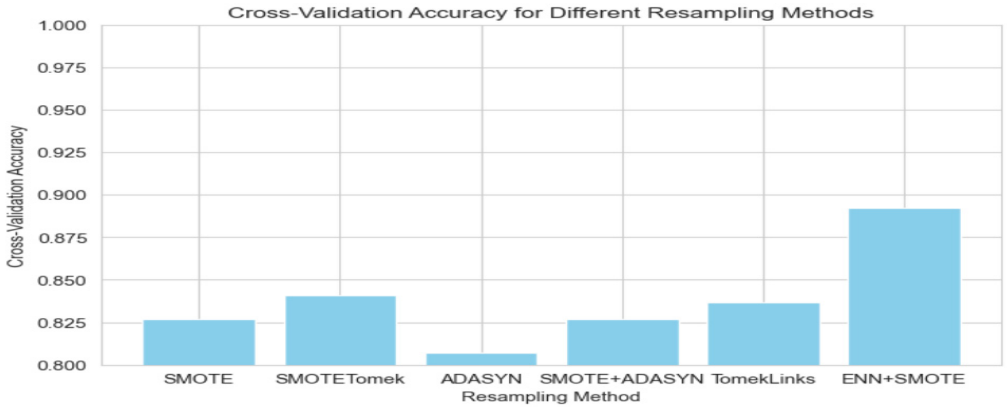

Figure 8. Cross-validation accuracy for resampling methods.

By selecting *ENN* + SMOTE, the model benefits from a cleaner, more balanced dataset that better represents both classes, leading to improved predictive performance and more reliable survival predictions on unseen data.

3.5. Hyperparameter Optimization and Model Visualization

3.5.1. Hyperparameter Optimization

This study uses three methods for hyperparameter tuning, grid search, random search, and Bayesian optimization, across seven machine learning models: Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNNs), Random Forest, XGBoost, LightGBM, and CatBoost. Hyperparameter tuning was consistently applied to all seven models. Figures 9 and 10 specifically illustrate the tuning process for logistic regression. The same tuning approach was applied to the other models, SVM, KNN, Random Forest, XGBoost, LightGBM, and CatBoost, even though the figures focus on logistic regression.

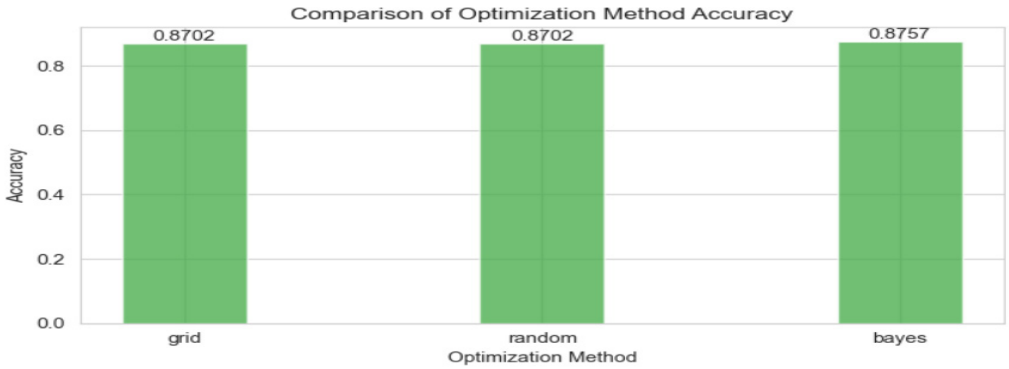

Figure 9. Comparison of optimization method accuracy.

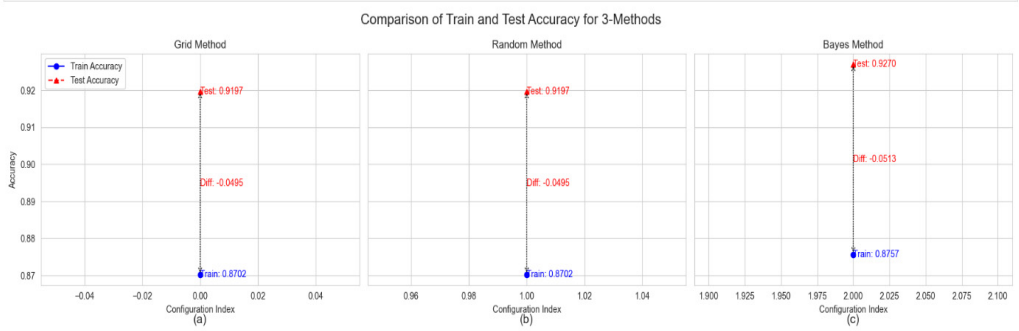

Figure 10. Visual comparison of training and test accuracy. (**a**) Grid method, (**b**) Random method, (**c**) Bayesian optimization method.

Figure 9 presents a comparison of the accuracy achieved by each optimization method. First, grid search exhaustively explored all possible combinations of predefined hyperparameters, yielding an accuracy of 0.8702 and a mean absolute error (MAE) of 0.0803, albeit at a significant computational cost. Second, random search efficiently sampled random combinations, resulting in the same accuracy and MAE as grid search while requiring fewer computational resources. Finally, Bayesian optimization employed a probabilistic model to focus on the most promising regions of the hyperparameter space, achieving the highest accuracy of 0.8757 and a lower MAE of 0.0730, thus proving to be the most effective method in terms of both performance and computational efficiency. The best hyperparameters identified by Bayesian optimization included a moderate regularization strength (C = 0.0101) and a maximum iteration count of 121. Therefore, Bayesian optimization was determined to be the most effective approach for optimizing model performance while balancing computational cost.

Figure 10 further illustrates the differences between training and test accuracy for each method, with arrows highlighting potential overfitting. Grid search and random search both showed a difference of -0.0495 between training and test accuracy, while Bayesian optimization exhibited a slightly larger gap of -0.0513. Despite this small difference, Bayesian optimization still demonstrated the lowest overall risk of overfitting, indicating better generalization to unseen data compared with the other methods. Therefore, Bayesian optimization was identified as the most effective method, balancing model complexity, generalization, and computational efficiency. Based on one's code and output, the result from Bayesian optimization will be returned as the best estimator.

In this study, hyperparameter tuning was applied consistently across seven machine learning models: Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest, XGBoost, LightGBM, and CatBoost. Each model underwent the same optimization process using grid search, random search, and Bayesian optimization to ensure comparability in performance evaluation. The goal was to identify the optimal set of hyperparameters for each model while balancing accuracy, mean absolute error (MAE), and computational efficiency. This uniform approach ensures that the models are evaluated on an equal footing, allowing for a fair comparison of their predictive performance and generalization ability.

3.5.2. Model Visualization

Figure 11 presents a comparison of model performance across three different configurations, with accuracy visualized by the three 'Model Comps' and mean absolute error (MAE) represented by two 'MAE Comps'.

Model Comp 1 corresponds to the accuracy achieved after applying regularization techniques, without applying Principal Component Analysis (PCA) to any models. The highest accuracy in this configuration was observed for the Random Forest model (0.9874), suggesting potential overfitting due to its high accuracy on the training data.

In Model Comp 2, where three hyperparameter optimization techniques (grid search, random search, and Bayesian search) were applied, the accuracy was more balanced across models. CatBoost achieved the highest accuracy at 0.9104, indicating that these optimization methods improved performance while mitigating overfitting.

Model Comp 3 shows the accuracy obtained using the ensemble voting classifier, which combines the best models from each optimization method. This configuration achieved the highest accuracy, particularly for CatBoost and LightGBM, both of which reached 0.9416, demonstrating strong generalization capabilities.

The MAE Comp 1 represents the error in the regularization scenario (Model Comp 1), where the Random Forest model, despite its high accuracy, had a relatively high MAE, suggesting poorer generalization. In contrast, the MAE Comp 2 shows the error when using the ensemble voting classifier (Model Comp 3). The lower MAE values, especially for CatBoost and LightGBM (with the lowest MAE at 0.0584), indicate that these models not only achieved higher accuracy but also significantly reduced prediction error.

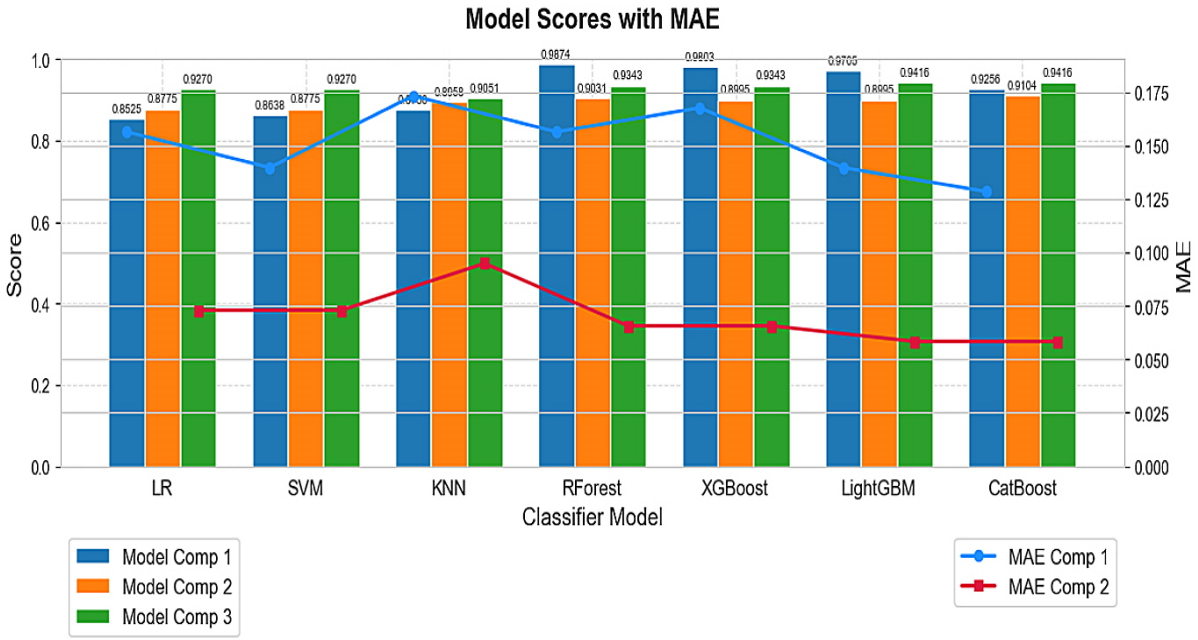

Figure 11. Ensemble of all model scores with MAE.

In summary, the MAE comparison reveals that while regularization can improve accuracy (Model Comp 1), it may not be sufficient to reduce prediction error. The ensemble voting approach in Model Comp 3, however, not only achieved superior accuracy but also minimized average error, reflecting improved generalization and robustness across models.

3.5.3. Ensemble by All Model ROC and P-R Curve

Figure 12 evaluates the performance of seven machine learning models using the ensemble voting classifier. The performance of the models is measured using two main metrics: Area Under the Receiver Operating Characteristic curve (AUC) and Average Precision (AP).

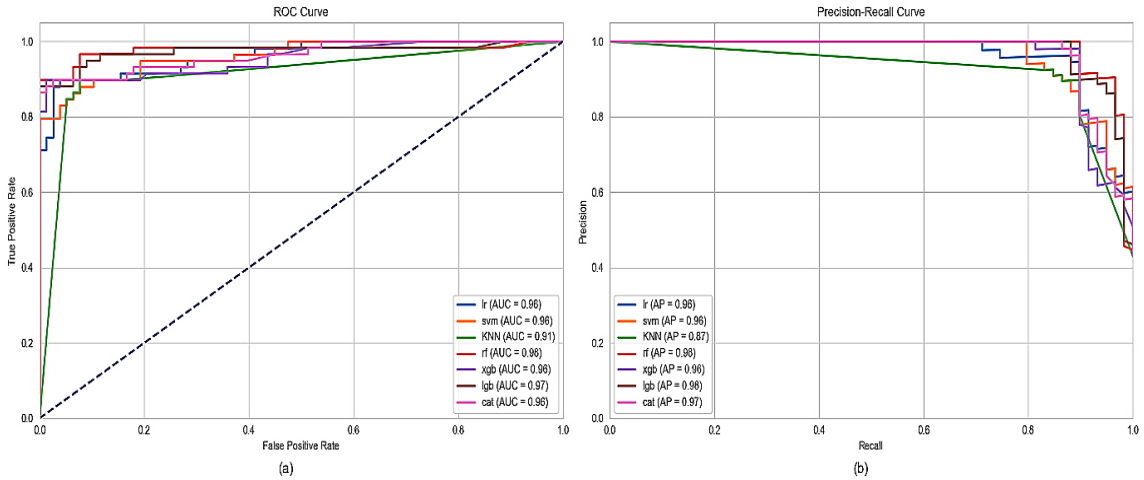

Figure 12. Ensemble by all models (**a**) ROC curve, (**b**) P-R curve.

These metrics provide valuable insights into each model's ability to separate classes and deliver precise classification. The linear model Logistic Regression performed well, achieving an AUC of 0.9618 and an Average Precision (AP) of 0.9619, indicating high precision in distinguishing between positive and negative classes. Similarly, the SVM model showed strong performance, comparable to Logistic Regression, with an AUC of 0.9620 and an AP of 0.9618. However, K-Nearest Neighbors (KNNs) underperformed

relative to the other models, with an AUC of 0.9148 and an AP of 0.8732, suggesting limitations in handling this dataset effectively in terms of precision and class separation.

The Random Forest model achieved the highest overall performance, with an AUC of 0.9767 and an AP of 0.9812, demonstrating its ability to manage the dataset's complexity and provide accurate classification. The XGBoost model also showed strong performance but with a slightly lower AUC of 0.9553 and AP of 0.9608, while still proving to be robust. LightGBM performed similarly well, with an AUC of 0.9739 and AP of 0.9775, positioning it among the top performers.

Finally, CatBoost, which is optimized for categorical data, demonstrated strong class separation and accuracy, with an AUC of 0.9635 and AP of 0.9670.

Overall, ensemble methods, particularly Random Forest, LightGBM, and XGBoost, proved to be highly effective in classifying the Titanic dataset. These models captured the data's underlying patterns more effectively than individual models like Logistic Regression and SVM, thanks to the strengths of ensemble learning. This finding suggests that combining models using ensemble techniques can lead to highly accurate and precise predictions. Moving forward, Stacking and Voting will be applied as ensemble methods, along with SWA, to further enhance prediction precision.

3.6. Stochastic Weighted Averaging (SWA) and Final Ensemble Model

In this study, a final ensemble model was implemented, combining Stacking and Voting classifiers, followed by the application of Stochastic Weighted Averaging (SWA) to improve model generalization and performance. The ensemble was constructed using seven base models: Logistic Regression (LR), SVM, K-Nearest Neighbors (KNN), Random Forest (RF), XGBoost, LightGBM, and CatBoost. Each model was integrated into a pipeline that included feature selection via SelectKBest. The results of the ensemble before and after SWA implementation are shown in Figure 13.

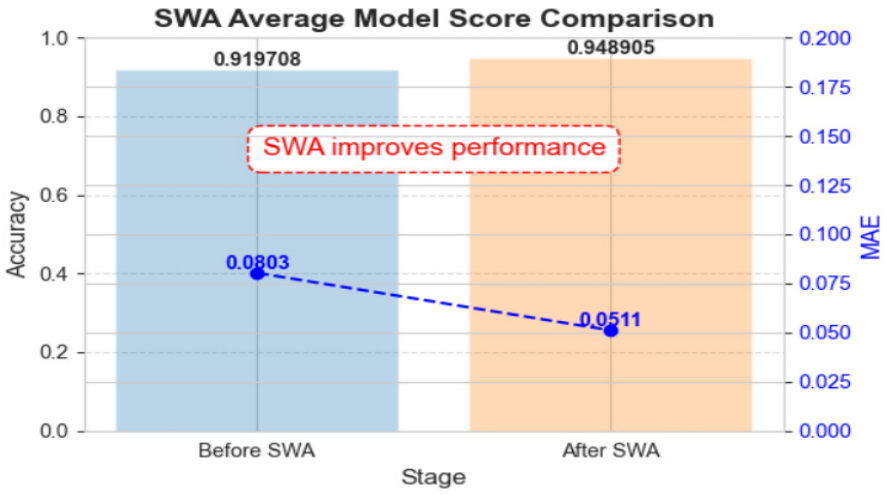

Figure 13. Final ensemble SWA.

3.6.1. Model Training and Performance Before SWA

The ensemble model was initially evaluated without SWA. The Stacking classifier, using XGBoost as the meta-model, and the Voting classifier, which employed a soft voting mechanism, were applied to the test data. The accuracy and mean absolute error (MAE) were calculated to assess the model's performance. The model achieved an accuracy of 0.9197 and an MAE of 0.0803, indicating strong baseline performance with reasonable accuracy and minimal error.

3.6.2. Application of SWA and Performance After SWA

The final stage involved applying SWA, wherein the ensemble model was retrained multiple times with slight variations in initial conditions (random states). The retrained

models were combined using SWA, which averages model weights over multiple iterations, thereby improving generalization and reducing overfitting. After SWA, the model's performance significantly improved, achieving an accuracy of 0.9489, with the MAE further reduced to 0.0511. This increase in accuracy and reduction in error demonstrates SWA's effectiveness in enhancing the model's ability to generalize to unseen data, providing a more robust predictive solution.

3.6.3. Conclusion of the Ensemble and SWA Results

The application of SWA resulted in a substantial improvement in the model's predictive power, with accuracy increasing from 0.9197 to 0.9489—a gain of approximately 3%. These findings confirm that SWA effectively enhances the generalization performance of ensemble models, making it a valuable approach for machine learning tasks. The results demonstrate that combining Stacking and Voting ensemble methods with SWA leads to more accurate and reliable predictions, especially for complex datasets like the Titanic dataset. The final model's robustness, reflected in its reduced MAE and higher accuracy after SWA, suggests that this approach is ideal for achieving both precision and stability in predictive tasks.

3.6.4. SWA Performance Indicators and ROC and P-R Curve

Table 3 presents the evaluation metrics for the classification task, comparing the model's performance before and after applying Stochastic Weighted Averaging (SWA). As shown, the model's accuracy increased significantly from 0.92 to 0.95 after SWA. Furthermore, the precision, recall, and F1 score metrics reflect enhanced predictive capability, with precision reaching a perfect score of 1.00 post SWA. These results indicate that SWA not only improved the model's accuracy but also boosted its overall reliability and effectiveness.

Table 3. Performance Metrics Before and After Applying SWA.

Model	Accuracy	Precision	Recall	F1 Score
SWA (before)	0.92	0.90	0.92	0.91
SWA (after)	0.95	1.00	0.88	0.94

Evaluation of the final optimized ensemble model resulted in the following indices: AUC (Area Under the Curve) of 0.98 and AP (Average Precision) of 0.98, as shown in Figure 14. These indices confirm that the ensemble model provides not only high accuracy but also excellent precision and recall, making it highly effective for classification tasks.

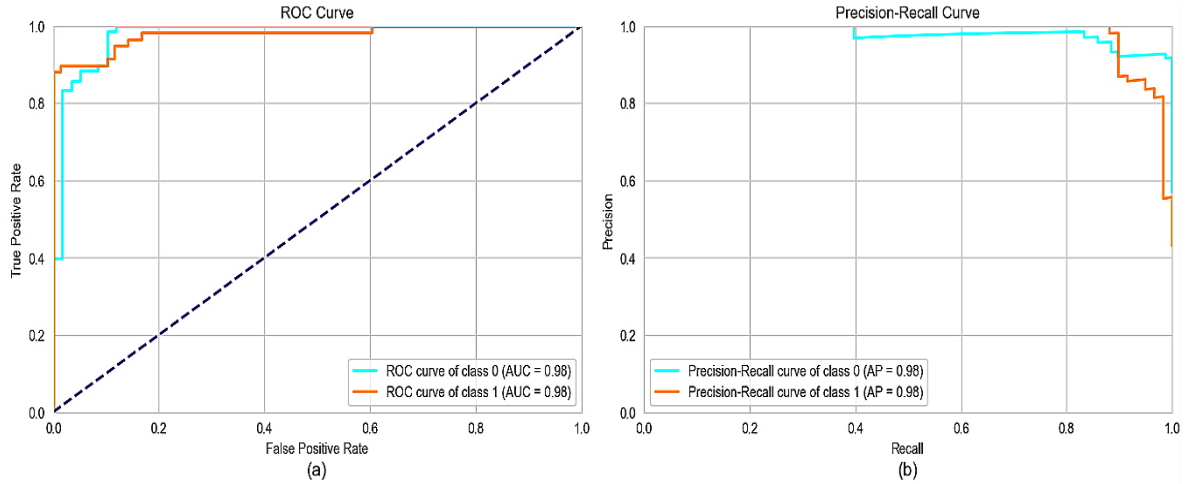

Figure 14. SWA–ensemble model (**a**) ROC curve, (**b**) P-R curve.

4. Discussion

This study presents a comprehensive ensemble-based approach to enhancing predictive performance in classification tasks, with a focus on the Titanic dataset. By integrating seven diverse machine learning models—Logistic Regression, Support Vector Machine (SVM), Random Forest, XGBoost, LightGBM, and CatBoost—into a unified ensemble, the model's robustness and accuracy were significantly improved. The combination of Stacking and Voting strategies leverages the strengths of each individual model, allowing the ensemble to capture complex data patterns more effectively than any single model could.

Furthermore, the application of Stochastic Weight Averaging (SWA) was pivotal in improving the generalization of the ensemble model. SWA averaged the model weights over multiple training cycles, reducing the likelihood of overfitting and enabling the model to generalize better to unseen data. This step was crucial, as models without SWA showed a tendency to perform well on training data but were less robust when applied to new data. After applying SWA, the model exhibited a substantial improvement in test accuracy, precision, and other key performance metrics.

The key findings of this study highlight the ensemble model's effectiveness, with notable improvements in accuracy, AUC, and Average Precision after the application of SWA. Specifically, the ensemble model's accuracy increased from 91.97% to 94.89%, while the AUC and Average Precision both reached 0.98. These improvements indicate that the combination of Stacking, Voting, and SWA provides a highly effective framework for building accurate and reliable classification models.

To provide a comprehensive evaluation of our model's performance, we compared our results with those of several recent studies using the Titanic dataset. Gupta and Saurabh (2023) [23] implemented an ensemble approach combining Random Forest, SVM, and Gradient Boosting, achieving an accuracy of 81.3%. In another study, Wu (2024) [24] compared AdaBoost, XGBoost, and Random Forest with Random Forest, yielding a highest accuracy of 83.205%. Manjusha et al. (2023) [25] also performed a comparative analysis using various machine learning models, reporting an accuracy of 81.10% with a Gradient Boost Trees model. Finally, Zhang (2024) [26] achieved an accuracy of 87.5% with XGBoost and Random Forest.

In comparison, our model, which incorporates dimensionality reduction (PCA), imbalanced data handling (SMOTE-ENN), and ensemble learning enhanced by Stochastic Weighted Averaging (SWA), achieved an accuracy of 94.89% and an AUC of 0.98. These results highlight the effectiveness of our hybrid approach in handling data imbalance and optimizing prediction accuracy, outperforming other models in terms of both accuracy and generalization.

This paper makes comparisons with works cited in references [23–26], which are studies that used machine learning methodologies on survival prediction tasks. However, while our case study is comparable to these works, the methodology we propose is more general and not limited to survival prediction alone. The same steps can be adapted for other classification tasks, particularly those involving class imbalance, missing data, or complex feature interactions.

5. Conclusions

In conclusion, this study demonstrates the significant benefits of employing ensemble methods, particularly the combination of Stacking and Voting strategies followed by Stochastic Weight Averaging (SWA) in predictive modeling tasks. The integration of seven machine learning models into an ensemble allowed for a robust and highly accurate classification system capable of handling complex datasets with a high degree of precision. The application of SWA further improved the model's ability to generalize, minimizing the generalization error and boosting overall performance metrics.

The results of this study underscore the effectiveness of ensemble learning, particularly when enhanced by SWA, in achieving high accuracy and reliability in classification tasks. With an accuracy improvement from 91.97% to 94.89% and consistently high AUC and

Average Precision values of 0.98, the final model demonstrates superior predictive performance. This approach shows great potential for applications in other complex predictive modeling tasks, where maximizing accuracy and reducing overfitting are critical objectives.

Future research could explore the extension of this methodology to other datasets and domains, as well as the impact of different model configurations and hyperparameter tuning strategies on the overall performance of the ensemble model. Moreover, further investigation into alternative methods for reducing overfitting and improving generalization, such as data augmentation or additional regularization techniques, could complement the results observed in this study.

Author Contributions: Conceptualization, Y.H. and I.J.; methodology, Y.H.; software, Y.H.; validation, Y.H. and I.J.; formal analysis, Y.H.; investigation, Y.H.; resources, Y.H.; data curation, Y.H.; writing—original draft preparation, Y.H.; writing—review and editing, Y.H. and I.J.; visualization, Y.H.; supervision, I.J.; project administration, I.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy restrictions.

Acknowledgments: The authors would like to thank the editor and the anonymous reviewers for their valuable comments and suggestions, which have significantly contributed to improving the quality of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mishra, V.P.; Singh, B.; Shukla, V.K.; Dasgupta, A. Predicting the likelihood of survival of Titanic's passengers by machine learning. In Proceedings of the 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 17–18 March 2021; pp. 52–57.
2. Singh, K.; Nagpal, R.; Sehgal, R. Exploratory data analysis and machine learning on Titanic disaster dataset. In Proceedings of the 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 29–31 January 2020; pp. 320–326.
3. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
4. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian optimization of machine learning algorithms. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Lake Tahoe, NV, USA, 3–6 December 2012; pp. 2951–2959.
5. Fatima, S.; Hussain, A.; Amir, S.B.; Aslam, S.M.H. XGBoost and Random Forest algorithms: An in-depth analysis. *Pak. J. Sci. Res.* **2023**, *3*, 26–31. [CrossRef]
6. Dasgupta, A.; Mishra, V.P.; Shukla, V.K.; Singh, B. Analyzing Titanic disaster using machine learning algorithms. In Proceedings of the International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 5–6 May 2021; pp. 406–411.
7. Brownlee, J. *Evaluating Machine Learning Algorithms*; Machine Learning Mastery: Vermont, Australia, 2020.
8. Ai, Y. Predicting Titanic survivors by using machine learning. *Highlights Sci. Eng. Technol. CSIC* **2022**, *34*, 360–367. [CrossRef]
9. Kaggle Titanic Dataset. Available online: https://www.kaggle.com/competitions/titanic/data (accessed on 5 September 2024).
10. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
11. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
12. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. TensorFlow: A System for Large-Scale Machine Learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
13. Chollet, F. Keras: The Python Deep Learning Library. Available online: https://keras.io (accessed on 5 September 2024).
14. Zhang, Z. Introduction to K-Nearest Neighbors. *Mach. Learn. Appl.* **2017**, *12*, 58–64.
15. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
16. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]

17. Lipton, Z.C. The Mythos of Model Interpretability. In Proceedings of the 2016 ICML Workshop on Human Interpretability in Machine Learning, New York, NY, USA, 19 June 2016.
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26–30 June 2016; pp. 770–778.
19. Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
20. Zhou, Z.; Feng, J. Deep forest: Towards an alternative to deep neural networks. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), Melbourne, Australia, 19–25 August 2017.
21. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; pp. 448–456.
22. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), Sardinia, Italy, 13–15 May 2010; pp. 249–256.
23. Gupta, Swati and Saurabh, Sonal, Ensemble Approach for Titanic Survival Predictor. Available online: https://ssrn.com/abstract=4663312 (accessed on 13 December 2023).
24. Wu, T. Predicting Titanic Survival Rates: A Comparison of AdaBoost, XGBoost, and Random Forest. Ph.D. Thesis, University of California, Los Angeles, CA, USA, 2024.
25. Manjusha, D.; Pujith, P.; Sailusha, V. Comparative Analysis for Survival Prediction from Titanic Disaster Using Machine Learning. *i-Manag. J. Softw. Eng.* **2023**, *18*, 36.
26. Zhang, Y. Titanic Survival Prediction Based on Machine Learning Algorithms. *Highlights Sci. Eng. Technol.* **2024**, *107*, 189–195. [CrossRef]