Previous work Review and Assessment

With the increasing impact of software applications on today's businesses and activities, software attribute prediction such as effort estimation, maintainability, defect and quality classification are gaining growing interest from both academic and industry communities.

A research paper by[16] focuses on the increasing importance of software attribute prediction, such as effort estimation, maintainability, defect, and quality classification, due to a growing reliance on software applications. The paper highlights the limitations of conventional prediction algorithms such as decision trees, Bayesian methods, and artificial neural networks multilayer perceptron (ANN-MLP) when handling skewed and redundant defect datasets. The research introduces ensemble learning's voting mechanism as a solution, which assigns higher weights to successful individual classifiers, thereby mitigating the effects of feature irrelevance and redundancy. The paper's main objective is to demonstrate the positive effect of feature selection on the performance of defect classification. The findings suggest that for a dataset with 0.5% defective components, a classification accuracy of 99.5% can be achieved by classifying all components as non-defective, and an overall accuracy of 99% can be achieved by a binary classifier that classifies all data samples as the majority class. In the discussion section, the paper compares the results of the average probability ensemble (APE) model with two basic

classifiers (W-SVMs and random forests). The W-SVMs classifier assigns weights inversely proportional to the class occurrence in the dataset, allowing for potentially better model fitting in the case of imbalanced datasets.

Another paper by[17] discusses the process of software defect prediction, which involves identifying likely flawed sections of software. The paper introduces a model that uses a base layer of Linear Discriminant Analysis (LDA), K Nearest Neighbors (KNN), and Generalized Linear A model with Elastic Net Regularization (GLMNet) and a top layer of Random Forest (RF). The results demonstrate that this model is capable of effectively handling PROMISE datasets, known for their noisy attributes and high dimensions. This paper is anticipated to make a significant contribution to the field of software defect prediction. According to the results, the Ensemble machine learning technique offers superior prediction accuracy for software defect prediction, providing a significant insight from this study. The proposed model achieved an overall prediction accuracy of 88.56% across all experimental datasets. Furthermore, the Mean Squared Error (MSE) of the proposed model is significantly lower than other models, demonstrating that the The ensemble technique effectively handles errors in the learning model caused by noise, bias, and variance. Therefore, the research suggests that ensemble machine-learning models provide a robust solution for software defect prediction.

Methodology

Data Collection and Preprocessing

The NASA Metrics Data Program (MDP) software defect datasets were collected from various NASA software projects. These datasets encompass information on software modules each having their attributes and a binary label indicating whether the module is defective or not. The datasets were preprocessed to remove any missing values by imputing the missing values by the mean of the column and removing duplicates found in the datasets.

Brief Overview of the Task and Models Used

The task at hand involved a binary classification problem, a common type of task in the field of machine learning. Binary classification refers to the process of classifying elements into two distinct groups based on certain characteristics or features. In this context, the task was to predict whether a data point belongs to the False class (majority class) or the True class (minority class). To accomplish this task, four different machine learning models were employed. These models represent a variety of algorithmic approaches and were chosen for their distinctive strengths in handling classification tasks.

Support Vector Machine

The Support Vector Machine (SVM) is a widely utilized algorithm in the field of machine learning, particularly for classification and regression tasks. According to[4], who first introduced the concept, SVM is a classifier which performs its task by constructing a hyperplane in a multi-dimensional space that separates data points of different classes. The The SVM model operates on the principle of maximizing the margin, which is the distance between the hyperplane (decision boundary) and the nearest data points from different classes, known as support vectors.

The hyperplane is defined as: $w \cdot x + b = 0$ where...

w is the weight vector.

x is the input data vector.

and b is the bias term.

The decision function for SVM is given by:

$$f(x) = \text{sign}(w \cdot x + b)$$

where...

$f(x)$ predicts the class label of input vector x.

$sign(\cdot)$ returns the sign of its argument.

In cases where the data is not linearly separable from its original feature space, SVM can use the kernel trick to implicitly map the data to a higher-dimensional space where it becomes separable.

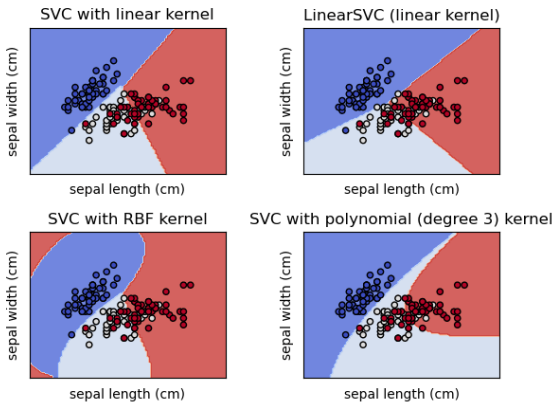

Figure 1: SVM Classifier

Linear Kernel

Equation: $K(x_i, x_j) = x_i^T x_j$

Explanation: The linear kernel represents the dot product between the input feature vectors x_i and x_j. This kernel is used when the data is linearly separable in the original feature space.

Polynomial Kernel

Equation: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$

Explanation: The polynomial kernel maps the input data into a higher-dimensional space using a polynomial function. The hyperparameters d and r are user-defined positive integers that control the degree of the polynomial and a constant term, respectively. The parameter γ is a scaling factor that influences the dot product.

Gaussian (RBF) Kernel

Equation: $K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$

Explanation: The Radial Basis Function (RBF) kernel measures the similarity between two data points using the Gaussian distribution. It implicitly maps the data into a high-dimensional space, making it suitable for handling nonlinear relationships. The hyperparameter γ controls the width of the Gaussian kernel.

Sigmoid Kernel

Equation: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Explanation: The sigmoid kernel uses the hyperbolic tangent function to map the data into a higher-dimensional space. This kernel can be useful in some cases, but it is generally less commonly used compared to linear, polynomial, and Gaussian kernels.

Laplacian Kernel

Equation: $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|}{\gamma}\right)$

Explanation: The Laplacian kernel measures the similarity between two data points using the Laplace distribution. It is similar to the Gaussian kernel but has a sharper peak, making it robust to outliers.

This characteristic makes SVM highly effective in high-dimensional spaces and in situations where the number of dimensions is greater than the number of samples[12]. In addition to performing linear classification, SVMs can also effectively handle non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces[24]. In terms of binary classification, the SVM model seeks a hyperplane that separates the data into two classes while maximizing the margin between the classes.

Random Forest

Random Forest is a versatile and popular machine-learning algorithm known for its simplicity and diversity. It was first introduced by[1] as an extension of the decision tree algorithm. A Random Forest operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

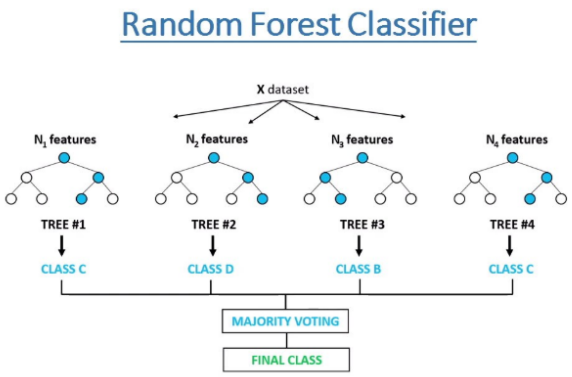

Figure 2: Random Forest

This idea of combining several models to improve the predictive performance is known as ensemble learning ([6]). Random Forest attempts to mitigate the high variance problem of individual decision trees, where small changes in the training set can result in significantly different tree structures. By averaging the results of a collection of de-correlated trees, it reduces the risk of overfitting and typically provides better predictive performance ([23]). Random Forest also has an inherent feature selection mechanism, as it ranks features (variables) based on their ability to improve the purity of the node, measured by the Gini impurity or entropy ([8]).

Logistic Regression

The Logistic Regression model, despite its name, is a powerful tool for binary classification tasks. This model calculates the probability that a given data point belongs to a certain class by applying the logistic function to a linear combination of features[13]. One of the main advantages of Logistic Regression is its interpretability. Each feature is assigned a coefficient that describes its relative importance and direction of association with the assumption that there is a linear relationship between the log odds of the dependent variable and the independent variables. It also requires that the observations be independent, and the absence of multicollinearity among the independent variables[20]. The equation of logistic function or logistic curve is a common "S" shaped curve defined by the below equation. The logistic curve is also known as the sigmoid curve[25]. the formula of logistic function:

$$P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

where..

$$-(\beta_0 + \beta_1 x)$$

is linear function

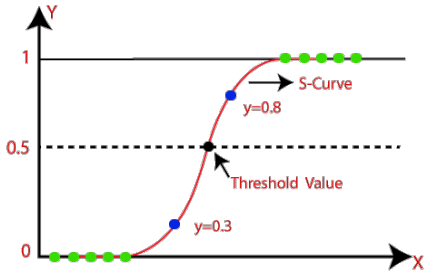

Figure 3: Logistic function

Ensemble Model

Ensemble Models are powerful machine learning tools that combine multiple base models to improve prediction accuracy and robustness over a single model. The central idea behind ensemble models often referred to as the "Wisdom of the Crowd", is that a group of weak learners can come together to form a strong learner[7].

There are several types of ensemble methods, including Bagging, Boosting, and Stacking. Bagging, or Bootstrap Aggregating, involves training each model in the ensemble using a randomly drawn subset of the training set. Boosting is a sequential process, where each subsequent model attempts to correct the mistakes of the previous models. Stacking involves training a model to combine the predictions of several other models[28]. Ensemble models have been shown to significantly increase accuracy on a variety of tasks, including both regression and classification problems[19].

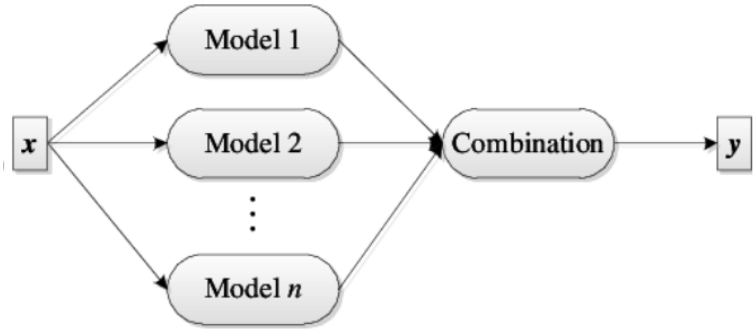

Figure 4: Ensemble Model

They tend to be more robust to noise, outliers, and overfitting, as they average out biases, reduce variance, and are not likely to model random noise in the output data[29]. However, ensemble models can be computationally expensive and may require more time to train and predict than individual models. Also, they may suffer from complexity, which may make them harder to interpret than individual models[22].

Performance Metrics

In machine learning, the performance of a model is evaluated based on certain metrics that depend on the nature of the task whether it is a classification, regression, or clustering task. This paper focuses on the metrics used to evaluate classification models, including accuracy, precision, recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUROC). Accuracy is the ratio of correctly predicted observations to the total observations is the most straightforward measure but can be misleading in the case of imbalanced classes[21].

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

Precision, the ratio of correctly predicted positive observations to the total predicted positives is an essential metric when the cost of false positives is high[26].

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

Recall measures the ratio of correctly predicted positive observations to all observations in the actual class, and is crucial when the cost of false negatives are high[26].

$$\text{Recall } = \frac{TP}{(TP + FN)}$$

The F1-score is the weighted average of precision and recall and is typically more useful than accuracy, particularly for uneven class distributions[27].

$$F1 \text{ Score } = \frac{2TP}{(2TP + FP + FNN)}$$

Cross validation

According to[14], Cross-validation is a powerful preventative measure against overfitting. The technique is used to assess how the results of a statistical analysis will generalize to an independent data set. It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split. In machine learning, we cannot fit the model on the training data and say the model will work well with the real-world data. We need some kind of assurance or approximation about how our model will behave in the future or unseen data. This is where Cross-validation plays a crucial role.

[14] explained that Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. It provides a more accurate method of measuring the performance of a model, which helps in improving the model's performance and optimizing the hyperparameters. The most common type of cross-validation, and the one often referred to by this name, is k-fold cross-validation. In k-fold cross-validation, you divide your data points into k subsets of roughly equal size. You then run k separate learning experiments:

- Randomly partition the data into k equal-sized subsets. One of the k subsets is used as the test set, and the other k-1 subsets are put together to form a training set.
- Train a machine learning model on the k-1 training sets and test the model on the test set, recording the test set error rate.
- Repeat the process k times, each time using a different subset as your test set.

Finally, compute the average of the k-test set error rates. This is the cross-validation error rate, an estimate of the model's prediction error. This process is illustrated in the figure below:

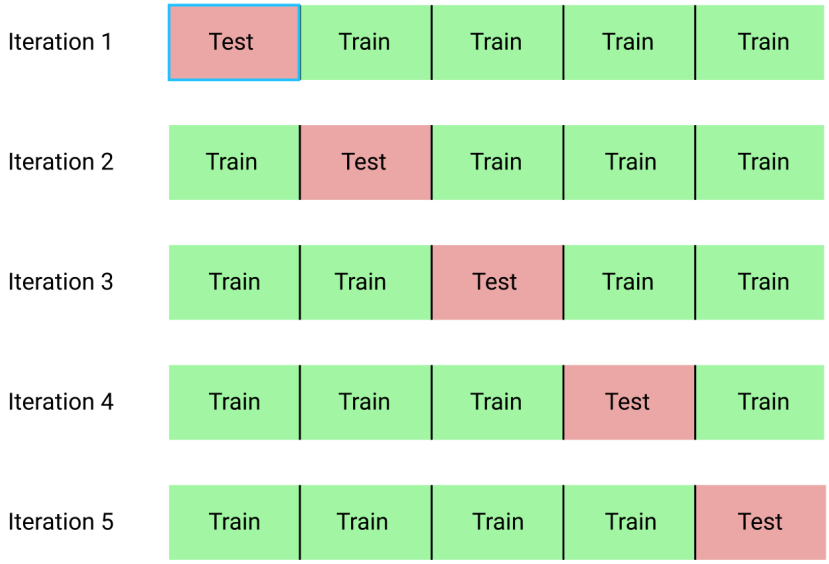

Figure 5: kfold

Cross-validation helps in making the model more robust by validating it on different subsets of data and then taking the average of all the performances which helps in improving the model stability. A model's effectiveness is determined by its ability to perform well on unseen data. Cross-validation, such as k-fold cross-validation, helps ensure that the model's performance is not just tied to the way the data was split.

Model performance analysis

Four different machine learning models were evaluated in this study, which included a Support Vector Machine (SVM), a Random Forest, a Logistic Regression, and an Ensemble model. The performance of these models was assessed based on a binary classification task.

Table 1: Accuracies for each model on all datasets

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM	0.9	0.8	0.8	0.8	0.7	1.0	0.7	0.8	0.9	1.0	0.9	0.9	0.8
Random Forest	0.8	0.8	0.7	0.9	0.8	0.9	0.8	0.8	0.9	1.0	0.9	0.9	0.8
Logistic Regression	0.8	0.8	0.7	0.8	0.7	1.0	0.7	0.9	0.9	1.0	0.9	0.9	0.8
Voting Classifier	0.9	0.8	0.7	0.8	0.7	1.0	0.7	0.9	0.9	1.0	0.9	0.9	0.8

Table 2: Recall for each model on all datasets

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM	0.0	0.0	0.1	0.1	0.6	0.0	0.6	0.1	0.2	0.0	0.0	0.4	0.2
Random Forest	0.0	0.2	0.3	0.0	0.9	0.0	0.4	0.0	0.1	0.0	0.1	0.4	0.4
Logistic Regression	0.0	0.1	0.2	0.1	0.6	0.1	0.6	0.3	0.3	0.0	0.3	0.5	0.3
Voting Classifier	0.0	0.1	0.2	0.1	0.6	0.0	0.6	0.1	0.2	0.0	0.0	0.4	0.3

Table 3: Precision for each model on all datasets

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM	0.0	0.7	0.8	0.2	0.7	0.0	0.6	0.5	0.7	0.0	0.0	0.9	0.9
Random Forest	0.0	0.5	0.6	0.0	0.8	0.0	1.0	0.0	0.5	0.0	0.3	0.9	0.6
Logistic Regression	0.0	0.6	0.5	0.2	0.8	0.5	0.6	0.8	0.5	0.0	0.5	0.9	0.7
Voting Classifier	0.0	0.6	0.6	0.3	0.7	0.0	0.7	1.0	0.7	0.0	0.5	0.9	0.8

In conclusion, although high accuracy was demonstrated by all models, the other metrics were low. These results indicated the presence of a class imbalance in the dataset, a common problem in machine learning tasks. Despite the high accuracy demonstrated by all models, difficulties were encountered in effectively classifying the minority class.

Table 4: F1-scores for each model on all datasets

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM	0.0	0.1	0.2	0.2	0.7	0.0	0.6	0.1	0.3	0.0	0.0	0.5	0.3
Random Forest	0.0	0.3	0.4	0.0	0.8	0.0	0.6	0.0	0.2	0.0	0.1	0.5	0.5
Logistic Regression	0.0	0.2	0.3	0.2	0.7	0.2	0.6	0.5	0.4	0.0	0.4	0.6	0.4
Voting Classifier	0.0	0.1	0.3	0.2	0.7	0.0	0.7	0.2	0.3	0.0	0.1	0.5	0.4

Strategies such as resampling techniques are suggested to address this issue. The model's performance should not only be assessed on overall accuracy but also on its ability to identify each class accurately, especially when dealing with imbalanced datasets.

Proposed Enhancement to Defect Prediction Model

The Problem with Imbalanced Data Sets

Class imbalance is a common problem in machine learning classification where there is a disproportionate ratio of observations in each class. Class imbalance can be found in many different areas including medical diagnosis, spam filtering, and fraud detection. The main issue with class imbalance is that most machine learning algorithms work best when the number of samples in each class is about equal. This is because most algorithms are designed to maximize accuracy and reduce error.

Synthetic Minority Over-sampling Technique (SMOTE)

Synthetic Minority Over-sampling Technique, or SMOTE, is a popular algorithm to create synthetic observations of the minority class. It was presented by[2].

SMOTE works by selecting examples that are close to the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

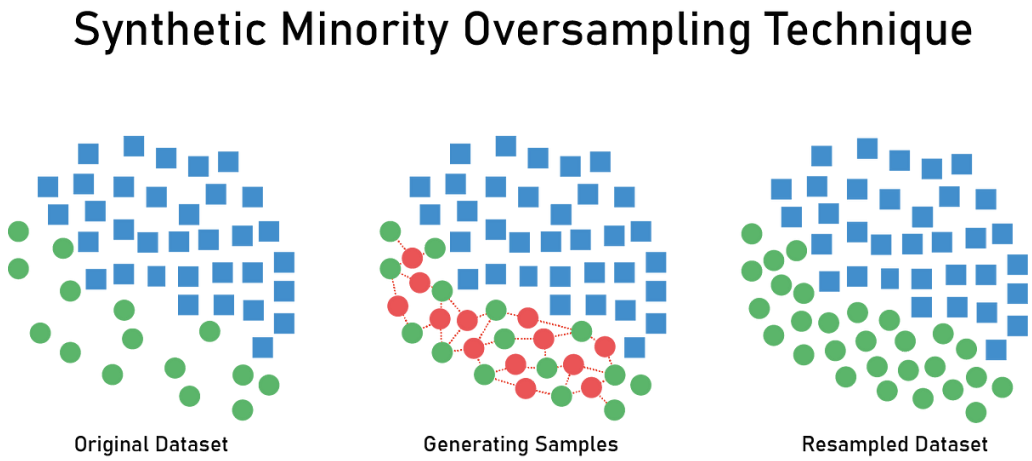

Figure 6: Schematic of the SMOTE algorithm.

The algorithm can be summarized as follows:

- For each sample in the minority class, calculate the k nearest neighbors.
- From the k neighbors, a neighbor is randomly selected.
- A synthetic sample is created by choosing a point between the two instances in feature space.

This process is repeated until the data is balanced.

Efficacy and Limitations of SMOTE

According to[9], the main advantage of SMOTE is that it can improve the performance of the minority class by generating synthetic examples that are quite similar to the existing observations in the minority class. However, one potential drawback of SMOTE is that it can increase the likelihood of overfitting since it generates synthetic examples without considering the majority class. New synthetic examples could be generated that are quite similar, or even identical, to existing examples. Furthermore, synthetic examples are generated without considering the majority class, possibly resulting in ambiguous examples if there is a strong overlap for the classes.

Implementation

We utilized the SMOTE algorithm implemented in the imbalanced-learn library in python, and setting the number of neighbors to 5 which is the default and it's sufficient for most cases.

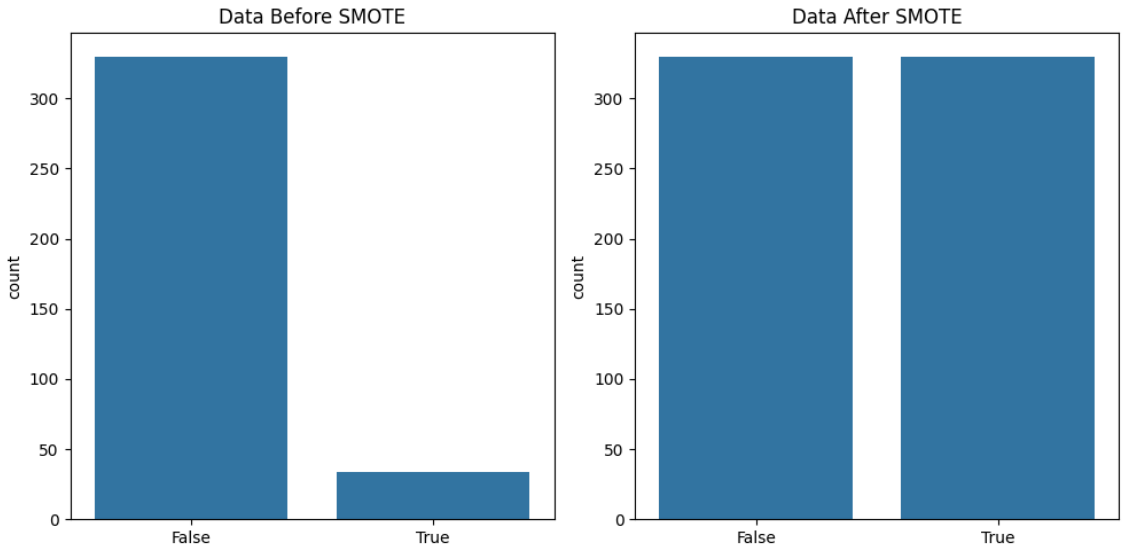

Figure 7: Data distribution before and after using SMOTE

Results

After Training SVM, Random Forest, Logistic Regression, and Ensemble models on the balanced dataset and evaluated their performance. The results showed a significant improvement in the performance of all models in precision, recall and consequently the f1-score. The best performance was observed with the Random Forest model.

Table 5: Accuracies for each model on all datasets after using SMOTE

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM (SMOTE)	0.8	0.7	0.7	0.7	0.7	0.8	0.6	0.7	0.8	0.8	0.8	0.8	0.7
Random Forest (SMOTE)	0.8	0.8	0.7	0.8	0.8	1.0	0.8	0.9	0.9	1.0	0.8	0.9	0.8
Logistic Regression (SMOTE)	0.8	0.7	0.6	0.7	0.7	0.8	0.6	0.7	0.8	0.8	0.8	0.8	0.7
Voting Classifier (SMOTE)	0.8	0.7	0.7	0.7	0.7	0.8	0.6	0.8	0.8	0.9	0.8	0.8	0.7

Table 6: Recall for each model on all datasets after using SMOTE

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM (SMOTE)	0.6	0.4	0.5	0.2	0.6	0.6	0.7	0.6	0.8	0.2	0.7	0.9	0.7
Random Forest (SMOTE)	0.2	0.3	0.4	0.1	0.8	0.1	0.7	0.4	0.3	0.0	0.3	0.7	0.6
Logistic Regression (SMOTE)	0.5	0.5	0.6	0.2	0.6	0.4	0.7	0.6	0.8	0.2	0.7	0.9	0.7
Voting Classifier (SMOTE)	0.5	0.5	0.5	0.2	0.6	0.5	0.7	0.6	0.8	0.2	0.7	0.9	0.7

Table 7: Precision for each model on all datasets after using SMOTE

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM (SMOTE)	0.4	0.4	0.4	0.1	0.7	0.1	0.6	0.3	0.2	0.0	0.2	0.5	0.5
Random Forest (SMOTE)	0.4	0.4	0.5	0.1	0.8	0.1	0.8	1.0	0.3	0.0	0.2	0.7	0.6
Logistic Regression (SMOTE)	0.4	0.4	0.4	0.1	0.8	0.1	0.5	0.3	0.2	0.0	0.2	0.5	0.4
Voting Classifier (SMOTE)	0.4	0.4	0.4	0.1	0.7	0.1	0.6	0.4	0.2	0.0	0.2	0.5	0.5

Table 8: F1-scores for each model on all datasets after using SMOTE

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM (SMOTE)	0.5	0.4	0.4	0.1	0.7	0.2	0.6	0.4	0.3	0.0	0.3	0.6	0.6
Random Forest (SMOTE)	0.2	0.4	0.5	0.1	0.8	0.1	0.7	0.6	0.3	0.0	0.3	0.7	0.6
Logistic Regression (SMOTE)	0.4	0.4	0.4	0.1	0.8	0.1	0.5	0.3	0.2	0.0	0.4	0.6	0.5
Voting Classifier (SMOTE)	0.5	0.4	0.4	0.2	0.7	0.1	0.6	0.5	0.3	0.1	0.4	0.6	0.6

Feature Selection

Feature selection is a crucial step in the machine learning pipeline. It is the process of selecting the most relevant features from the original dataset to use in model training. The aim is to enhance the performance of the machine learning model By using feature selection, we can simplify data entry, reduce size, and improve model performance in several ways according to[11]:

- Reduce overfitting: Feature selection helps prevent overfitting by focusing on the most important features that contain the most predictive information.
- Faster training: Reducing the number of features shortens the training time of a machine learning as there is less data to process in the model.
- Enhanced interpretation: We can improve the interpretation of models by selecting the most important features. This means we can better understand which features are relevant to the model's predictions and extract insights from the model's behavior.

There are two main categories of feature selection techniques: filter methods and wrapper methods.

Wrapper Methods

According to[15], The wrapper method is a type of feature selection technique that relies on the performance of a machine learning model to evaluate the importance of features. It involves training a model on a subset of features, evaluating its performance, and using that information to decide whether to add or remove features. This process repeats until it reaches an optimal set of features. The wrapper method is computationally expensive as it involves evaluating the performance of a machine-learning model for every

possible combination of features. However, it often provides a better-performing feature set compared to other methods, like filter methods, because it takes into account the interaction of features. There are different types of wrapper methods, including forward selection, backward elimination, and recursive feature elimination. Forward selection starts with an empty set and adds one feature at a time. Backward elimination starts with all features and removes one feature at a time. Recursive feature elimination is a greedy optimization algorithm that aims to find the best-performing feature subset.

Recursive Feature Elimination with Cross-Validation (RFECV)

RFECV is a powerful technique for feature selection in machine learning that fits a model and removes the weakest feature (or features) until the specified number of features are reached as mentioned by[10]. Features are ranked and by recursively eliminating a small number of features per loop, RFECV attempts to eliminate dependencies and collinearity that may exist in the model. It combines the strengths of recursive feature elimination (RFE) and cross-validation to provide a robust and efficient method for selecting the most important features and optimizing the performance of a machine learning model.

Then select the optimal number of features based on the cross-validation score. The cross-validation score generally increases with the number of features. However, the score may also increase due to overfitting. RFECV solves this problem by selecting the number of features for which the cross-validation the score is maximum. RFECV works by iteratively eliminating the least important features in a dataset while using cross-validation to evaluate the performance of the model. The process can be broken down into the following steps:

- Initialize the model with all the features: The first step is to initialize the model with all the features in the dataset.
- Perform cross-validation: The next step is to perform cross-validation on the dataset, using the initialized model. This involves splitting the dataset into training and testing sets and evaluating the model's performance on the testing sets.
- Eliminate the least important features: Based on the performance of the model in the cross-validation step, the least important features are eliminated from the dataset.
- Repeat steps 2–3: Steps 2–3 are repeated until a stopping criterion is met, such as a maximum number of iterations or a minimum number of features.
- Evaluate the final model: The final model is evaluated on the entire dataset to determine its performance.

according to[3], RFECV has several advantages over other feature selections. RFECV can improve the accuracy of a machine learning model by selecting the most important features. It is computationally efficient, as it only requires a single pass through the dataset to eliminate the least important features. Also, provides interpretable results, as it eliminates features one at a time, allowing for feature importance to be easily understood. However, RFECV needs a specified or calculable measure of importance provided by the estimator to perform its operations. Therefore, it doesn't work with all kinds of models. For instance, it won't work directly with models like KNN, and SVM with non-linear kernels as these models do not provide a straightforward measure of feature importance.

Filter Methods

Filter methods operate independently of any specific machine learning model. They rely on statistical measures to evaluate individual features and assign them a score based on their presumed relevance to the target variable[5]. These measures don't involve training a model, making filter methods computationally efficient and scalable for large datasets.

Advantages of Filter Methods:

- Computational efficiency: Filter methods are much faster than wrapper methods as they don't involve training models on multiple feature subsets.
- Scalability: They can be applied to large datasets without significant computational burden.
- Model independence: The selection process is not biased towards a specific machine learning model.

Drawbacks of Filter Methods:

- Overlooking feature interactions: Filter methods don't consider how features might interact with each other, potentially missing important relationships.
- Suboptimal for specific models: The chosen features might not be optimal for the performance of a particular machine learning model.

Information Gain

Information Gain, commonly referred to as Mutual Information (MI), quantifies the decrease in ambiguity regarding the objective variable subsequent to the evaluation of a specific characteristic. This metric capitalizes on the principle of entropy, a measure that gauges the level of randomness or indeterminacy connected to a stochastic variable[18].

The core idea:

- Entropy of the Target Variable $(H(Y))$: This represents the initial uncertainty about the target variable.
- Entropy after Considering a Feature $(H(Y|X))$: This represents the remaining uncertainty about the target variable after observing a specific feature.
- Information Gain $(IG(X, Y))$: This is the difference between the initial entropy and the conditional entropy. It signifies the information gained about the target variable by considering the feature. Features with higher IG are deemed more relevant for predicting the target variable.

Benefits of Information Gain:

- Versatility: Applicable to both categorical and continuous features, making it a flexible choice for mixed datasets.
- Interpretability: Provides a clear measure of the information a feature contributes to predicting the target variable.
- Scalability: Can be efficiently applied to large datasets.

Limitations of Information Gain:

- Computational Cost: While generally efficient, it can be more computationally expensive compared to simpler methods like chi-square, especially for large datasets.
- Independence Assumption: Assumes independence between features, which might not always be true in real-world data.

Comparing Results

The two feature selection techniques, Recursive Feature Elimination Cross-validation (RFECV) and Information Gain (IG), were compared to see how they affect the performance of various machine learning models. The tables show the accuracy, recall, precision, and F1-score of different models on several datasets after applying SMOTE (Synthetic Minority Oversampling Technique) and the two feature selection techniques. Overall, the random forest appears to perform the best based on the highest accuracy across all datasets in both feature selection techniques9. However, for recall10 and F1-score12, random forest only achieves the best results on some datasets. It's important to consider all these metrics together to choose the best model and feature selection technique depending on the specific task.

Here's a more detailed comparison:

- Accuracy: RFECV resulted in higher accuracy for most models compared to IG.
- Recall: The results are mixed, with both techniques achieving similar recall scores for most models.
- Precision: RFECV resulted in higher precision for most models compared to IG.
- F1-score: Similar to recall, the results are mixed with both techniques achieving similar F1-scores for most models.

Table 9: Accuracies for each model on all datasets after using SMOTE and Feature Selection

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM (SMOTE RFECV)	0.8	0.7	0.7	0.6	0.8	0.8	0.6	0.8	0.8	0.8	0.8	0.8	0.7
Random Forest (SMOTE RFECV)	0.8	0.8	0.7	0.8	0.9	1.0	0.8	0.9	0.9	1.0	0.8	0.9	0.8
Logistic Regression (SMOTE RFECV)	0.8	0.7	0.6	0.6	0.8	0.8	0.6	0.8	0.8	0.8	0.8	0.8	0.7
Voting Classifier (SMOTE RFECV)	0.8	0.7	0.7	0.7	0.9	0.8	0.6	0.8	0.8	0.9	0.8	0.8	0.7
SVM (SMOTE IG)	0.7	0.8	0.7	0.7	0.7	0.8	0.7	0.8	0.7	0.7	0.8	0.8	0.6
Random Forest (SMOTE IG)	0.8	0.8	0.7	0.7	0.8	1.0	0.7	0.9	0.9	1.0	0.8	0.9	0.7
Logistic Regression (SMOTE IG)	0.8	0.7	0.6	0.7	0.7	0.8	0.7	0.8	0.7	0.7	0.8	0.8	0.7
Voting Classifier (SMOTE IG)	0.8	0.8	0.7	0.7	0.7	0.8	0.7	0.8	0.7	0.7	0.8	0.9	0.7

Table 10: Recall for each model on all datasets

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM (SMOTE RFECV)	0.6	0.4	0.5	0.2	0.8	0.6	0.7	0.6	0.8	0.2	0.8	0.9	0.6
Random Forest (SMOTE RFECV)	0.2	0.3	0.4	0.1	0.9	0.1	0.7	0.4	0.4	0.0	0.4	0.7	0.6
Logistic Regression (SMOTE RFECV)	0.5	0.5	0.6	0.2	0.7	0.5	0.7	0.8	0.8	0.2	0.7	0.8	0.6
Voting Classifier (SMOTE RFECV)	0.5	0.5	0.5	0.2	0.8	0.5	0.7	0.7	0.8	0.2	0.7	0.9	0.6
SVM (SMOTE IG)	0.6	0.4	0.4	0.5	0.6	0.5	0.5	0.8	0.8	0.5	0.6	0.8	0.7
Random Forest (SMOTE IG)	0.2	0.4	0.5	0.0	0.8	0.1	0.6	0.3	0.5	0.0	0.5	0.7	0.5
Logistic Regression (SMOTE IG)	0.6	0.4	0.6	0.5	0.5	0.5	0.5	0.8	0.9	0.3	0.6	0.7	0.7
Voting Classifier (SMOTE IG)	0.6	0.4	0.5	0.5	0.6	0.5	0.5	0.8	0.8	0.3	0.6	0.7	0.7

Table 11: Precision for each model on all datasets

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM (SMOTE RFECV)	0.3	0.4	0.4	0.1	0.8	0.1	0.5	0.4	0.2	0.0	0.3	0.5	0.5
Random Forest (SMOTE RFECV)	0.4	0.4	0.5	0.1	0.8	0.1	0.8	1.0	0.4	0.0	0.3	0.7	0.6
Logistic Regression (SMOTE RFECV)	0.4	0.4	0.4	0.1	0.8	0.1	0.6	0.5	0.2	0.0	0.2	0.5	0.5
Voting Classifier (SMOTE RFECV)	0.4	0.4	0.4	0.1	0.9	0.1	0.6	0.4	0.2	0.0	0.3	0.5	0.5
SVM (SMOTE IG)	0.3	0.5	0.4	0.2	0.6	0.1	0.6	0.5	0.1	0.0	0.2	0.4	0.4
Random Forest (SMOTE IG)	0.3	0.5	0.5	0.0	0.8	0.1	0.7	0.8	0.3	0.0	0.3	0.7	0.5
Logistic Regression (SMOTE IG)	0.4	0.4	0.4	0.2	0.8	0.1	0.6	0.5	0.1	0.0	0.2	0.5	0.4
Voting Classifier (SMOTE IG)	0.4	0.5	0.4	0.2	0.6	0.1	0.6	0.5	0.1	0.0	0.2	0.6	0.4

Table 12: F1-scores for each model on all datasets

Model	CM1	JM1	KC1	KC3	KC4	MC1	MC2	MW1	PC1	PC2	PC3	PC4	PC5
SVM (SMOTE RFECV)	0.4	0.4	0.4	0.1	0.8	0.1	0.6	0.5	0.3	0.0	0.4	0.6	0.5
Random Forest (SMOTE RFECV)	0.2	0.4	0.5	0.1	0.8	0.1	0.8	1.0	0.4	0.0	0.3	0.7	0.6
Logistic Regression (SMOTE RFECV)	0.4	0.4	0.4	0.1	0.8	0.1	0.6	0.5	0.2	0.0	0.4	0.6	0.5
Voting Classifier (SMOTE RFECV)	0.4	0.4	0.4	0.2	0.9	0.1	0.6	0.4	0.2	0.1	0.4	0.6	0.5
SVM (SMOTE IG)	0.3	0.4	0.4	0.3	0.6	0.1	0.6	0.6	0.2	0.1	0.3	0.5	0.5
Random Forest (SMOTE IG)	0.2	0.4	0.5	0.0	0.8	0.1	0.7	0.8	0.3	0.0	0.3	0.7	0.5
Logistic Regression (SMOTE IG)	0.5	0.4	0.5	0.3	0.8	0.1	0.6	0.5	0.1	0.0	0.2	0.5	0.4
Voting Classifier (SMOTE IG)	0.5	0.4	0.4	0.3	0.6	0.1	0.6	0.5	0.1	0.1	0.4	0.6	0.4

Conclusion

The results indicate that SMOTE when applied in conjunction with certain models like SVM, Random Forest, Logistic Regression, and Voting Classifiers, can improve the performance of these models, especially in terms of recall and F1-score. However, the impact of SMOTE on precision and accuracy varied depending on the dataset and the model used.

Feature selection techniques, such as Recursive Feature Elimination with Cross-Validation (RFECV) and Information Gain (IG), also played a crucial role in improving model performance. These techniques helped reduce overfitting and enhance the generalization ability of the models, particularly in scenarios with a high number of features.

Overall, the best-performing model varied across datasets, but in general, Random Forest and Voting Classifier, especially when combined with SMOTE and feature selection, showed promising results. These models achieved competitive performance across all metrics, making them suitable choices for handling imbalanced multivariate time series regression datasets. Further research could explore other techniques or models to further improve performance in these challenging scenarios.

Bibliography

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] Qi Chen, Zhaopeng Meng, and Ran Su. Werfe: A gene selection algorithm based on recursive feature elimination and ensemble strategy. *Frontiers in Bioengineering and Biotechnology*, 8, 2020.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[5] S. Delany, M. Buckley, and D. Greene. Sms spam filtering: methods and data. *Expert Systems With Applications*, 39:9899–9908, 2012.

[6] Thomas G. Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.

[7] Thomas G Dietterich. Ensemble methods in machine learning. *Multiple classifier systems*, 1857:1–15, 2000.

[8] Ramón Díaz-Uriarte and S. Amelia De Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):1–13, 2006.

[9] Alberto Fernández, Salvador García, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905, 2018.

[10] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[11] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[12] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[13] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. John Wiley & Sons, 2013.

[14] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, 1995.

[15] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[16] Issam H. Laradji, Mohammad Alshayeb, and Lahouari Ghouti. Software defect prediction using ensemble learning on selected features. *Information and Software Technology*, 58:388–402, 2015.

[17] Adv Lear, Emmanuel Dada, David Oyewola, Stephen Joseph, Ali Dauda, Stephen Bassi, and Ali Baba. Ensemble machine learning model for software defect prediction. *Advances in Machine Learning and Artificial Intelligence*, 2:11–21, 07 2021.

[18] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[19] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[20] Chao-Ying Joanne Peng, Kristin L Lee, and Gary M Ingersoll. An introduction to logistic regression analysis and reporting. *The journal of educational research*, 96(1):3–14, 2002.

[21] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2011.

[22] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[23] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[24] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks — ICANN'97*, pages 583–588, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[25] H. Sharma. Logistic regression - python implementation from scratch without using sklearn, July 2022.

[26] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[27] Cornelis Joost Van Rijsbergen. *Information retrieval*. Butterworth-Heinemann, 1979.

[28] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[29] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.